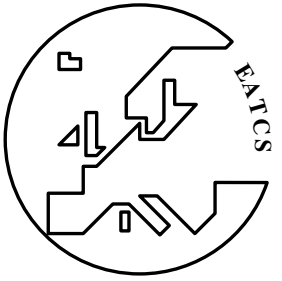
ISSN 0252–9742

Bulletin

of the

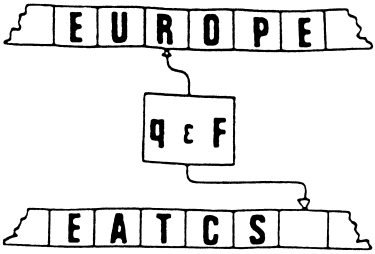
European Association for Theoretical Computer Science

EATCS



Number 106 February 2012

Council of the European Association for Theoretical Computer Science



President:	Burkhard Monien	Germany
Vice Presidents:	Giorgio Ausiello	Italy
	Don Sannella	United Kingdom
	Paul Spirakis	Greece
Treasurer:	Dirk Janssens	Belgium
Bulletin Editor:	Maria Serna	Spain

Luca Aceto	Iceland	Antonin Kucera	Czech Republic
Susanne Albers	Germany	Jan van Leeuwen	The Netherlands
Jos Baeten	The Netherlands	Elvira Mayordomo	Spain
Josep Díaz	Spain	Catuscia Palamidessi	France
Zoltán Ésik	Hungary	David Peleg	Israel
Fedor Fomin	Norway	Giuseppe Persiano	Italy
Lance Fortnow	USA	Jean-Eric Pin	France
Leslie Ann Goldberg	United Kingdom	Vladimiro Sassone	United Kingdom
Monika Henzinger	Austria	Roger Wattenhofer	Switzerland
Giuseppe F. Italiano	Italy	Thomas Wilke	Germany
Christos Kaklamanis	Greece	Gerhard Wöeginger	The Netherlands
Juhani Karhumäki	Finland		

Past Presidents:

Maurice Nivat	(1972–1977)	Mike Paterson	(1977–1979)
Arto Salomaa	(1979–1985)	Grzegorz Rozenberg	(1985–1994)
Wilfred Brauer	(1994–1997)	Josep Díaz	(1997–2002)
Mogens Nielsen	(2002–2006)	Giorgio Ausiello	(2006–2009)

EATCS Council Members

EMAIL ADDRESSES

Luca Aceto . . . . . luca@ru.is
Susanne Albers . . . . . albers@informatik.hu-berlin.de
Giorgio Ausiello . . . . . ausiello@dis.uniroma1.it
Jos Baeten . . . . . josb@win.tue.nl
Josep Díaz . . . . . diaz@lsi.upc.es
Zoltán Ésik . . . . . ze@inf.u-szeged.hu
Fedor Fomin . . . . . fomin@ii.uib.no
Lance Fortnow . . . . . fortnow@eecs.northwestern.edu
Leslie Ann Goldberg . . . . . L.A.Goldberg@liverpool.ac.uk
Monika Henzinger . . . . . monika.henzinger@univie.ac.at
Giuseppe F. Italiano . . . . . italiano@disp.uniroma2.it
Dirk Janssens . . . . . Dirk.Janssens@ua.ac.be
Christos Kaklamanis . . . . . kakl@ceid.upatras.gr
Juhani Karhumäki . . . . . karhumak@cs.utu.fi
Antonin Kucera . . . . . tony@fi.muni.cz
Jan van Leeuwen . . . . . jan@cs.uu.nl
Elvira Mayordomo . . . . . elvira@unizar.es
Burkhard Monien . . . . . bm@upb.de
Catuscia Palamidessi . . . . . catuscia@lix.polytechnique.fr
David Peleg . . . . . peleg@wisdom.weizmann.ac.il
Giuseppe Persiano . . . . . giuper@dia.unisa.it
Jean-Eric Pin . . . . . Jean-Eric.Pin@liafa.jussieu.fr
Don Sannella . . . . . dts@dcs.ed.ac.uk
Vladimiro Sassone . . . . . vs@ecs.soton.ac.uk
Maria Serna . . . . . mjserna@lsi.upc.edu
Paul Spirakis . . . . . spirakis@cti.gr
Roger Wattenhofer . . . . . wattenhofer@tik.ee.ethz
Thomas Wilke . . . . . wilke@ti.informatik.uni-kiel.de
Gerhard Wöeginger . . . . . g.j.woeginger@math.utwente.nl

Bulletin Editor: Maria Serna, Barcelona, Spain
Cartoons: DADARA, Amsterdam, The Netherlands

The bulletin is entirely typeset by pdfTeX and ConTeXt in TXfonts. The Editor is grateful to Ivan Couto for his support.

All contributions are to be submitted electronically through the Bulletin's web site and must be prepared in LaTeX 2_ε using the class `beatcs.cls` (a version of the standard LaTeX 2_ε `article` class). All sources, including figures, and a reference PDF version must be included in the submission.

Pictures are accepted in EPS, JPG, PNG, TIFF, MOV or, preferably, in PDF. Photographic reports from conferences must be arranged in ZIP files layed out according to the format described at the Bulletin's web site. Those reports will appear only in the electronic edition of the Bulletin.

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.

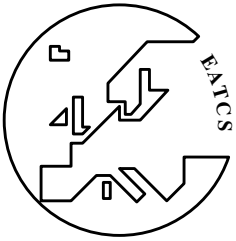
The details can be found at `http://www.eatcs.org/bulletin`, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email `bulletin@eatcs.org`.

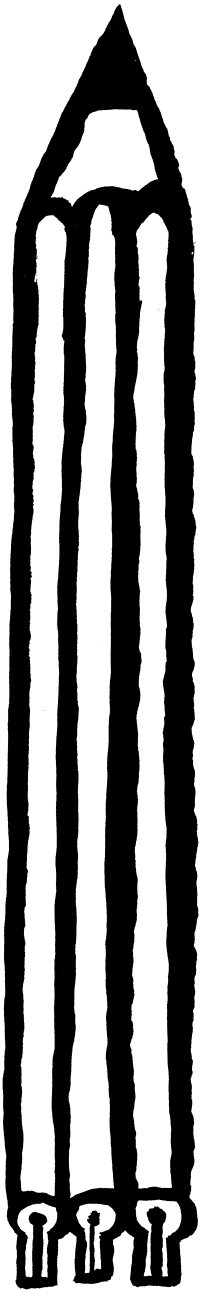
Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 10/15 weeks. Accepted papers will appear in print as soon as possible thereafter.

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

Table of Contents

EATCS Matters





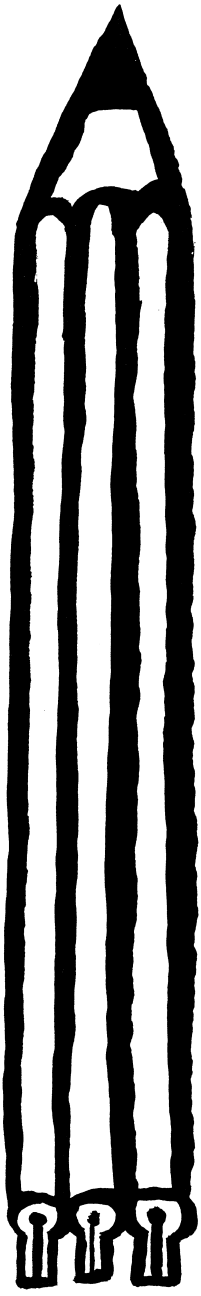
Letter from the President

Dear EATCS members,

I hope that you all have had an excellent start into the year 2012 and I take the opportunity to wish you all the best and much success for your work.

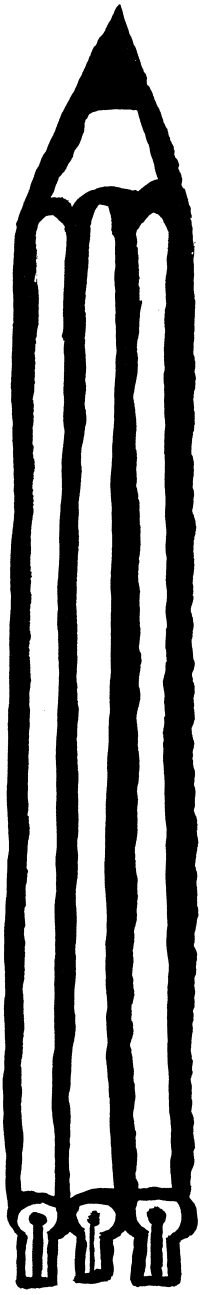
As usual, I would like to inform you with this letter about recent developments regarding our association. One of the news concerns the establishment of the so called European Forum for Information and Communication Sciences and Technologies (ICST) which was established on November 7th, 2011 in Milan by a joint action of seven leading organizations and societies in ICT in Europe (ACM Europe, CEPIS, EAPLS, EASST, EATCS, ERCIM, and Informatics Europe). The Forum is intended to be an open platform for cooperation among the scientific ICT societies in Europe and the mission statement is formulated as follows: "The development of common viewpoints and strategies for ICST in Europe and, whenever appropriate or needed, a common representation of these viewpoints and strategies at the international level." I had reported about the preceding meetings in Brussels (March 17th, 2011) and Prague (October 13th, 2010) during the General Assembly at ICALP'2011. The executive board of the forum consists of the President Jan van Leeuwen (ACM Europe, EATCS, Informatics Europe) and the two Vice-Presidents Keith Jeffery (ERCIM) and Paul Spirakis (EATCS). For more information I refer to the corresponding webpage http://www.cs.uu.nl/groups/AD/forum.html.

The organization of the next ICALP in



Warwick is proceeding well. We are very pleased that we have again an increase according to the submission numbers: Overall we have 433 submissions for ICALP 2012, 249 for track A, 105 for track B, and 79 for track C. This is a big step forward for track C and a slight increase at a high level for track A and B compared to the numbers of the last three ICALP's, whereas the ICALP in Reykjavik in 2008 has set an absolute high in recent years which seems to be difficult to beat. A particular highlight of ICALP 2012 promises to be the Turing Talk *which will be given by David Harel from the Weizmann Institute and which will be part of the celebration of the centennial of the birth of A.M. Turing. Moreover, at ICALP 2012 we will have again a special award session where the EATCS community will have the chance to listen to the talks of the 2012 winners of the Gödel-Prize, the EATCS-Award, and the Presburger-Award. In this context, let me remind you that EATCS, in cooperation with Springer provides sponsorship for ten 500-Euro student scholarships. The scholarships will be used to support participation of students in ICALP 2012 by covering early registration and possibly some of the local expenses. The applications will be reviewed by the ICALP 2012 conference and PC chairs. Preference will be given to PhD students from countries where access to funds is limited and who will present papers at the conference.*

The organization of ICALP 2013 has already started. ICALP will be organized next year in Riga by Rusins Freivalds and his team. The conference will have the same tracks as



in 2012, and in the meantime we have successfully completed the search for the Program Chairs for ICALP 2013. We are very pleased that we could win our first choice candidates Fedor Fomin (track A), Marta Kwiatkowska (track B) and David Peleg (track C) to act as PC chairs of the scientific program of ICALP 2013.

Finally, all that remains for me is to encourage you to participate at next ICALP and I hope we meet in Warwick!

Burkhard Monien, Paderborn
February 2012

Letter from the Bulletin Editor

Dear Reader,

First of all, I wish all of you a happy and successful 2012. Let it be a fruitful year for each of you, for theoretical computer science, and for EATCS.

*Our columns deliver the usual richness and variety of interesting contents. We start with a discussion on the recent progress in the connections between SAT algorithms and circuit lower bounds by Rahul Santhanam ("*Computational Complexity Column*"). A survey on recent results in distributed systems under non-uniform failure models by Petr Kuznetsov ("*Concurrency Column*"),and an excursion on some of the mechanisms for type inference used by the "Mathematical Componnets" project by Jeremy Avigad ("*Logic in Computer Science Column*").*

Let me draw your attention to the reports on activities and conferences included in this issue, and express my thanks to the authors in keeping track of the main ideas discussed at them.

I have to end this letter with a sad message. On January 23nd, 2011, our friend and colleague Sheng Yu passed away. I wish to express his family and all his colleagues the deepest mourning of our community. You can find an in memoriam note by Arto Salomaa in this bulletin issue.

I hope you'll enjoy the contents of this Bulletin issue,

Maria Serna, Barcelona
February 2012

In Memoriam

Sheng Yu
(1950-2012)

A wonderful scientist, most diligent collaborator and close friend of mine, *Sheng Yu*, passed away on January 23nd, the New Year of the Dragon, just before his 62nd birthday. His death was very unexpected: only a few days earlier we were still corresponding about referee reports concerning our joint work, and Sheng's letters contained no mention of a possible illness. Sheng is missed by his wife *Lizhen*, his family in China, notably his old mother, as well as by numerous friends and colleagues.

Sheng worked day after night after day and wanted to check all the details of a proof. He taught the same rigorous approach and work ethic to his students and postdocs. After a late dinner he still used to go to the university to continue his work. Nobody can tell how much the resulting deprivation of sleep contributed to his untimely death.

This writing contains mostly my personal thoughts and memories. However, I am convinced that everybody who got to know Sheng more closely, be it as a collaborator, teacher or in some other role, has similar experiences and recollections of his warm and helpful personality.

Sheng's studies were delayed by the cultural revolution and he was not very young anymore when he began graduate studies in Waterloo. That's where I met him the first time. He was a student in my course on recursive functions in 1982, obtaining the grade 105%. Karel Culik and the late Derick Wood were in those days in Waterloo. Sheng was in contact with both of them, and became a Ph.D. student of Karel Culik. Karel writes about him:"Sheng was not only my best Ph.D. student and valuable collaborator, he was also our family friend."

I lived in the same building with Sheng, so he became acquainted also with my wife. He wanted to cook for us. I remember him bringing all the supplies needed on his bicycle. Sheng was very helpful in our move back to Finland.

After finishing his Ph.D., Sheng spent half a year in 1986 as a postdoc in Turku. Our earliest joint work, on the equivalence of derivation (Szilard) languages and on a special public-key cryptosystem, dates back to his visit. Sheng was also one of the first users of email, if not the very first, at the university of

Bulletin of the EATCS no 106, pp. 7–9, February 2012

Turku. Also my family and many friends got to know him. We enjoyed his cooking many times. His skill and expertise in many areas, including sauna heating, became apparent to us. Sheng was even interviewed by the local newspaper Turun Sanomat because he was regularly playing table tennis in a local club. The interview had a big picture of Sheng and the coach of the club discussing “the philosophy of table tennis".

Sheng was teaching at Kent State University for some years at the end of the 80’s. Then he moved to the University of Western Ontario in London, where he has stayed afterwards. UWO is a school very familiar to me, already in 1966-68 I was visiting there. My cooperation with Sheng got a new beginning in the early 90’s. From 1991 to 2011 I have visited Sheng at UWO every year, with only two exceptions. Sheng also visited Turku frequently and, apart from scientific collaboration, served as the opponent and external examiner of several Ph.D. candidates.

Sheng was a wonderful person to work with. Both insightful and diligent, he was also willing to do most of the writing of papers and the correspondence in submitting them. In discussions he often had a crucial idea from which the solution could be deduced. Altogether I had 26 papers with Sheng. Sometimes we had coauthors: Yo-Sub Han, Tao Jiang, Efim Kinber, Alexandru Mateescu, Kai Salomaa and Derick Wood. Of the topics covered, the following come to mind: undecidability of the inclusion problem for pattern languages, codes with a finite delay and the P=NP problem, definition and study of Parikh matrices and the resulting subword histories and subword conditions, primality types of PCP solutions, commutativity conditions for languages, prime decomposition of languages versus length codes, state complexity of reversal and of combined operations.

The matters described above represent only a small part of Sheng’s scientific activity. He had 74 coauthors, and the topics covered extend far beyond the limits of theoretical computer science. His interests were unusually broad, and his work ranged from object-oriented programming and parallel processing to computer architecture. He was widely quoted, notably because of his seminal work in state complexity. In fact, Sheng was for several years planning a *Handbook of State Complexity*, with several coauthors. The book was already in the program of the publisher Springer-Verlag but Sheng always had to postpone the project because of other duties.

Sheng’s chapter on regular languages in the *Handbook of Formal Languages* is one of the basic references in the field. A special issue of the journal TCS was published for Sheng’s 60th birthday, as a token of appreciation in the scientific community.

Many of Sheng’s numerous Ph.D. students have become well-known scientists or have leading positions in industry. Sheng took very good care of each of them.

Our discussions were often interrupted because Sheng met a Ph.D. student. For such meetings Sheng had a weekly schedule. Sheng helped students in every possible way, including the details of writing papers, as well as, various intricacies of everyday life.

The courses given by Sheng were by no means restricted to theory. When one looks at the list of the courses taught by him, the title of a recent book, *Rainbow of Computer Science*, comes to mind. Although never the Department Head, Sheng had all the time numerous administrative duties. Sheng was a member or the chair in roughly half of the some 30 departmental committees in 2011. He was always busy with a heavy work load. Because he was so conscientious, the result was long work days. The best time to reach him by phone was midnight in his office.

Sheng was an invited speaker, program committee member or chair in many leading conferences. He was the originator and steering committee chair of the international CIAA conference series on *Implementation and Application of Automata*. Under Sheng's guidance the CIAA conferences have become well-established as the premier venue for research on new types of applications of automata theory. Sheng organized the big DLT (Developments in Language Theory) conference in London in 2010. By his initiative, the conference *Fifty Years of Automata Theory* was organized in London in 2000. Sheng was also an editor of several journals and took on editorial and refereeing tasks with the same dedication that characterized his own research.

Sheng was amazingly knowledgeable in classical music. We often went to concerts together, both in London and in Turku. Recently our music discussions concerned mostly Bruckner and Mahler. Apart from music, Sheng had many other interests. When discussing religion, sports, politics or personal relations, Sheng always expressed original ideas, sometimes very strongly.

One could write a book about Sheng's hospitality and willingness to help. I am not the only person who has experienced this; there are many others, for instance my good friend Grzegorz Rozenberg. Sheng and Lizhen organized big dinner parties, often in their home or in the Springbank Park. Sheng was always looking after me, driving me around, helping me in stairs, carrying my bag. The *Sheng number* indicates how many times Sheng took me to or from my hotel during my stay in London. In 2009 and 2010 the Sheng number was 30 and 32, respectively. In 2011 it was only 22 because my stay was shorter.

Sit tibi terra levis. Ollos iäti muistettu.

Turku, January 2012, *Arto Salomaa*

Institutional Sponsors

BiCi, Bertinoro international Center for informatics
Bertinoro, Italy

CTI, Computer Technology Institute
Greece

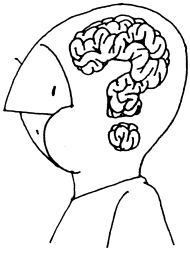
Elsevier
Amsterdam, The Netherlands

MADALGO, Center for Massive Data Algorithmics
Aarhus, Denmark

Microsoft Research
Cambridge, United Kingdom

Springer-Verlag
Heidelberg, Germany

EATCS News



News from Latin America

BY

Alfredo Viola

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

In this issue I present the 1st Latin American Theoretical Informatics School and the call for papers of Latincrypt 2012. 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.

1st Latin American Theoretical Informatics School

This will be a first of, what will hopefully be, a series of Schools on topics related to Theoretical Computer Science which are expected to take place in conjunction with the Latin American Theoretical Computer Science Symposium (LATIN). In particular, the first version of the School will be co-located and run in parallel to LATIN2012 on April 16 - 20 at Arequipa, Perú.

The School's objectives are to encourage attendance of Latin American students to the LATIN Symposium, to expose them to recent research developments, to give them the opportunity of learning recently developed advanced topics, and to facilitate their interaction with researchers working in and outside Latin America. The target audience are graduate students and advanced undergraduate students, in particular those studying in Latin America.

The scheduled courses are

- "Information Spreading in Distributed Systems" by Keren Censor-Hillel (MIT, USA).
- "Applications of Analytic Combinatorics to the Analysis of Algorithms: An introduction" by Conrado Martínez (UPC, Spain).
- "Discrete and Computational Geometry (with Applications to Routing, Clustering, and others)", by Jorge Urrutia (UNAM, México).

The Scientific Committee members are David Fernández-Baca (USA), Marcos Kiwi (Chile), Gonzalo Navarro (Chile), and Sergio Rajsbaum (México), while the local organizers are María Pilar Rondón and Ernesto Cuadros-Vargas. For more information you may consult at `http://latinschool2012.dim.uchile.cl/`.

Call for papers: Latincrypt 2012

Latincrypt 2012 is the Second International Conference on Cryptology and Information Security in Latin America, and will take place from October 7th to October 10th 2012 in Santiago, Chile. Latincrypt 2012 is being organized by CLCERT at the Univ. of Chile, in cooperation with The International Association for Cryptologic Research (IACR). Original papers on all technical aspects of cryptology are solicited for submission to Latincrypt 2012. The conference seeks original contributions on new cryptographic primitive proposals, cryptanalysis, security models, hardware and software implementation aspects, cryptographic protocols and applications, as well as submissions about cryptographic aspects of network security, complexity-theoretic cryptography, information theory, coding theory, number theory, and quantum computing.

The submission deadline is May 4, 2012, and the web page of the conference is `http://2012.latincrypt.org/`.

Regional Events

- April 16 - 20, 2012, Arequipa, Perú: Latin American Symposium on Theoretical Informatics (LATIN2012). `http://latin2012.cs.iastate.edu/`.
- April 16 - 20, 2012, Arequipa, Perú: 1st Latin American Theoretical Informatics School. `http://latinschool2012.dim.uchile.cl/`.
- October 7 - 10, 2012, Santiago, Chile: Second International Conference on Cryptology and Information Security in Latin America (LATINCRYPT 2012). `http://2012.latincrypt.org/`.

News from New Zealand

BY

C. S. Calude

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

1 Scientific and Community News

0. The meeting *Analysis and Randomness*, http://www.cs.auckland.ac.nz/~nies/ARAhome.html, organised by A. Nies, was held in Auckland on 12–13 December 2011. Speakers: Laurent Bienvenu, Willem Fouche, Cameron Freer, Rupert Hölzl, A. Melnikov, Kenshi Miyabe, Jason Rute, Tom ter Elst, and Dan Turetsky.

1. The *12th Asian Logic Conference*, http://msor.victoria.ac.nz/Events/ALC2011/WebHome, was held 15–20 December 2011 in Wellington. Plenary Speakers: Hiroakira Ono, Mic Detlefsen, Akito Tsuboi, Noam Greenberg, Simon Thomas, Isaac Goldbring, Grigor Sargsyan and Wu Guohua. Several special sessions have been organised, including *Algorithmic Randomness* (by R. Downey and A. Nies), and *Computability and Algebraic Structures* (by R. Downey).

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

407. K. Svozil. Neutrino Dispersion Relation Changes Due to Radiative Corrections as the Origin of Faster-than-Light-in-Vacuum Propagation in a Medium. 09/2011

408. A.A. Abbott, C.S. Calude and K. Svozil. On Demons and Oracles. 11/2011

409. C.S Calude and E. Calude. The Complexity of Euler's Integer Partition Theorem. 11/2011

410. C.S Calude and E. Calude. The Complexity of Mathematical Problems: An Overview of Results and Open Problems. 11/2011

411. L. Staiger. On Oscillation-free Chaitin h-random Sequences. 11/2011

412. L. Staiger. Asymptotic Subword Complexity. 11/2011

413. D.H. Bailey, J.M. Borwein, C.S. Calude, M.J. Dinneen, M. Dumitrescu and A. Yee. An Empirical Approach to the Normality of π. 11/2011

414. S. Datt and M.J. Dinneen. Towards Practical P Systems: Discovery Algorithms. 12/2011

415. R. Nicolescu. Parallel and Distributed Algorithms in P Systems. 12/2011

416. M. Burgin, C.S. Calude and E. Calude. Inductive Complexity Measures for Mathematical Problems. 12/2011

2 A Dialogue with Reinhard Wilhelm about Compiler Construction and Dagstuhl

Reinhard Wilhelm is professor and leader of the chair for programming languages and compiler construction at Saarland University and the scientific director of the Leibniz Center for Informatics at Schloss Dagstuhl since its inception in 1990.

Professor Wilhelm has obtained numerous results in compiler construction, static program analysis, embedded real time systems, animation and visualization of algorithms and data structures. He is one of the co-developers of the MUG1, MUG2 and OPTRAN compiler generators, which are based on attribute grammars. He is a co-founder of the European Symposium on Programming, ESOP, and the European Joint Conferences on Theory and Practice of Software, ETAPS, a member of the ACM SIGBED Executive Committee and a member of the Scientific Advisory Board of CWI.

Professor Wilhelm is a fellow of the ACM (2000) and a member of Academia Europaea (2008); he was awarded the Alwin-Walther medal (2006), the Prix Gay-Lussac-Humboldt (2007), the Konrad-Zuse medal (2009), the Cross of the Order of Merit of the Federal Republic of Germany and the ACM Distinguished Service Award (2010); he has honorary doctorates from RWTH Aachen and Tartu University (2008).

Cristian Calude: You studied mathematics, physics and mathematical logic at University of Münster, computer science at Technical University Munich and Stanford University and obtained your PhD at TU Munich, quite a broad background. Please reminiscence about this period.

Reinhard Wilhelm: I studied at a time when the first curricula in computer science were being established. As a native of Westphalia, Westfälische Wilhelms Universität Münster with its strong tradition in Mathematics and Mathematical Logic was a natural starting point. Josef Stör, a numerical analyst from my home town, on the faculty of USC San Diego, recommended to switch to computer science, an advice I followed after passing the Vordiplom exam in Münster. At TH, later TU Munich, I was among the first students of the new curriculum in computer science. I finished this obtaining a Diploma degree, already oriented towards compiler construction. The German Academic Exchange Service (DAAD) offered one-year fellowships to study computer science in the US as they felt that the CS curricula did not yet have the same quality as the American curricula. I obtained such a fellowship and studied at Stanford University for one year. It was an exciting year with courses taught by Robert Floyd, Donald Knuth, Zohar Manna, John McCarthy, Robin Milner, and Niklaus Wirth. Looking back, the semantics people, Floyd, Manna, and McCarthy, seemed to have had the strongest influence on me. I gathered practical experience in compiler construction with my MS project, part of the port of the Zurich Pascal compiler to the IBM 360 machine.

CC: You discovered connections between code selection and regular tree automata, which are relevant for code generation.

RW: My group at Saarland University developed a formally-based approach to compiler optimizations expressed as transformations of attributed trees. The necessary tree pattern-matching algorithm—identifying places where transformations could be applied—used a subset construction on non-deterministic tree automata as I learned later from Helmut Seidl. I found some informal proposals in the literature proposing to express code selection by tree parsing. This led to a beautiful and efficient approach using deterministic bottom-up tree automata, which could be nicely combined with dynamic programming to identify least-cost code sequences. However, reality, i.e. processor-architecture design, made this beautiful approach obsolete as real processor architectures did not offer the required regularity.

CC: Although your research is quite practical, the theoretical component is strong. How do you manage this?

RW: Well, the colleagues in the CS department at Saarland University have a strong conviction, that nothing is as practical as a good theory. This conviction has been a recipe for success. Our curriculum has always had a strong theoretical

foundation on which one could build solid practical work.

CC: Reinhard Wilhelm and Dieter Maurer's book *Compiler Design*—written in German and translated into English and French—is a good illustration of the interplay between theory and applications: it offers a solid theoretical foundation for compilers for imperative, object oriented, functional and logic-based languages.

RW: I was not content with the Dragon Book, the dominant compiler textbook, which was and is by and large void of the theoretical foundations for compiler design. The underlying theory, however, is quite beautiful. So, I decided to write a book that I would like to teach from. I was fortunate to have Dieter Maurer in my group, who coauthored the first two editions. Currently, I cooperate with Helmut Seidl and Sebastian Hack on a rather complete rewrite for the third edition. The virtual machines in this third edition are made more uniform. The code-optimization part introducing static program analysis and program transformations has been largely extended. The code-generation chapters will be completely restructured and rewritten due to new insights into the code-generation process obtained in Sebastian Hack's dissertation.

CC: Please explain the shape analysis based on three-valued logic you designed.

RW: Static program analysis, which received most of its theoretical foundations by Patrick and Radhia Cousot in the 70s, computes invariant properties of all behaviors of a program. Abstract interpretation, as the Cousots formulated it, uses an abstraction of the semantics of the programming language to determine these invariants at all program points. Due to the impossibility to be sound and complete at the same time, sound static analysis approximate these properties; they give up completeness, but maintain soundness.

A largely unexplored area was the static analysis of heap-manipulating programs. These offer particular challenges, namely dynamically created anonymous objects and linked data structures of unbounded size. During a sabbatical I spent in Israel I was fortunate to meet Mooly Sagiv, then a student at the Technion. He asked me for a good thesis topic, and I proposed to develop a specification language for static program analyses. Mooly and I cooperated on this topic for something like 16 years, joined by Tom Reps, whom I knew from our attribute-grammar times.

The breakthrough in our research came with the discovery that predicate logic was a good basis to express program semantics, and that a reinterpretation of the same semantics over a 3-valued logical domain—the third value expressing *don't know*—could be used as an abstract interpretation. Our approach was parametric in the abstraction properties, i.e., different sets of predicates could be used to obtain different abstractions, which would (approximately) different properties of the program.

The *shapes* occurring in the name *Shape Analysis*were somethin like generalized *types* of data structures in the heap. Example are *singly-linked list without shared nodes*, *balanced binary tree* etc.

CC: Your ACM fellowship citation refers to your research on compiler construction and program analysis. Can you discuss one or two important results in this area?

RW: A result of my group that had quite some impact is the development of an approach to derive run-time guarantees for real-time embedded systems, that is, to show that such systems satisfy their timing constraints. These are often quite tight; in the automotive domain, they range down to microseconds. At the same time, the execution platforms used to realize these systems have a huge variability of execution times: the execution time of an instruction depends on the state of the platform and may vary by a factor of 100 or more.

The engineers at Airbus in Toulouse called us to help them because they knew that their traditional methods, based on measurement, were not sound for the new architectures they were deploying in their planes. We were able to solve this problem and provide tools through a spinoff company, AbsInt, that Airbus could use. The meanwhile long cooperation between Airbus, AbsInt, and my group at the University was so successful that several time-critical subsystems of the Airbus A380, the *big Airbus*, were certified with the AbsInt tool, which thereby became the only tool worldwide to be *validated* for the certification of these avionics applications. This work is considered as one of the major success stories of formal methods.

CC: How do you see your book *Informatics: 10 Years Back. 10 Years Ahead* (Springer 2001), 10 years after its publication?

RW: That is hard to answer! I would have to reread the prognoses contained in it. In the domain of verification, I have recently coauthored a manifesto, *Formal Methods—Just a Euroscience?* attempting to describe the state of the art. This could be compared with the articles in the monograph you refer to.

CC: Please summarise your manifesto.

RW: The manifesto gives an overview of how far different formal methods, in particular the verification methods, have been taken up by industry. There are notable differences between hardware and software industries and also some between Europe and America. The acceptance of verification methods is related to the costs of potential failures. The chip manufacturers have broadly adopted verification methods after the Pentium bug cost Intel a lot of money. The Ariane 5 disaster due to a software bug was very helpful to raise problem awareness in some parts of the embedded-systems industry. There is a somewhat surprising distribution of

strongholds for the different verification methods; model checking is stronger in the US, abstract interpretation stronger in Europe, deductive verification initially stronger in the US, but now strong in Europe.

One particular insight I gained in my work with industry and which is described in the manifesto is that the different verification techniques have a different distribution of roles, researcher, tool developer, user. In academia, typically the researcher also develops the tools based on his findings, and, of course, he is an enthusiastic user of his own tools. Some of the biggest disappointments resulted from the expectations raised by enthusiastic researchers/tool developers when the tools were deployed in industry and engineers could not use them.

CC: Since 1990 you have been the scientific director of the Leibniz Center for Informatics at Schloss Dagstuhl. What was the initial motivation of starting this center? How did it evolve in the last twenty years?

RW: The Leibniz Center for Informatics was formed after the famous Mathematics Research Institute in Oberwolfach, in the Black Forest. Theoretical computer scientists had been guests there for a number of years and felt the desire to have an Oberwolfach for Informatics. The German Informatics Society (GI) set up a search committee to identify an appropriate place for it. Several offers were made by the states Baden-Württemberg, Rheinland-Pfalz, and Saarland. The search committee selected Schloss Dagstuhl, a late-baroque mansion, at that time a retirement home run by a nuns order. The Saarland government agreed to buy the ensemble for the center and the German National Science Council supported the decision to set the center up in Dagstuhl.

Apparently, the Informatics community had waited for this center. Against my expectations it filled up rather quickly. An extension building was opened in 1995 together with a new kitchen and a restaurant. The greater capacity also filled up quickly so that lead times of far more than a year became common. You must know that meetings in Dagstuhl, the so-called Dagstuhl Seminars, result from successful applications to a Scientific Directorate, which meets twice a year to decide about the submitted proposals.

CC: Yes, I indeed know as I was privileged to be invited to a few seminars. As a participant to both Oberwolfach and Dagstuhl, I noted similarities but also differences...

RW: Definitely, Oberwolfach was our role model when we set up Dagstuhl. When I had been convinced to run Dagstuhl, I went to Oberwolfach together with my colleague on the administrative side, Wolfgang Lorenz, to get advice from Martin Barner, the long-time director of the Mathematical Research Institute, on what to do and, even more importantly, what not to do. Among the latter was his recommendation not to establish entailed estates, that is, long running series of

meetings, which ran too long to be ever stopped. We, therefore, established an iron rule that the organizing team of a series had to, at least incrementally, change from instance to instance. This was not always well received by organizing teams, but proved fruitful in the long run.

Another notable difference to Oberwolfach was that we charged participation fees right from the beginning. Computer scientists usually are better funded than mathematicians, and our fees were more symbolic than covering real costs.

Let me report an anecdote about where Dagstuhl profited from Oberwolfach. I was amazed by the fantastic music room on Oberwolfach. Great instruments and an extensive musical library! Actually, I had met Don Knuth and told him about our plans for Dagstuhl, and he had sent me a letter saying that he had always enjoyed playing the grand piano in Oberwolfach. The White Hall in Dagstuhl, a beautiful baroque hall, offered itself for our music room. I set out to buy instruments, a grand piano—not as grand as the one in Oberwolfach—, a decent violin, a cello. The executive in the ministry in charge of supervising our efforts complained about us acquiring a grand piano. I sent him a copy of Knuth's letter to prove that luminaries like him would find their way to Dagstuhl because of the grand piano. This stopped the complaints.

Another anecdote on setting up the music library in Dagstuhl. Musical scores are very expensive. So I thought about how to save on buying a basic library. I knew that the German publishers had licensed editions to the Eastern countries not meant to be reimported, at least not large scale. At that time I was playing with a Hungarian pianist. I told him my problem and asked him to see how he could import Eastern editions of scores for Dagstuhl. Next time, a friend of his came to visit him, he had the trunk of his car full with scores, somewhat biased towards Southeast Europe, all that for just 1000 DEM. I was nervous about what would happen to the fellow and the smuggled scores at the Austro-Hungarian border, and, in fact, Austrian customs asked the fellow to open the trunk of his car. On top of all the scores, there was a twelve-pack of cigarettes. They made him pay a fine for smuggling cigarettes.

CC: In addition to the music, the dedication to the fine arts is visible in Dagstuhl. What is the origin for this?

RW: Although I have a sister who is an artist my connection to the fine arts was not very strong. That changed when the extension building in Dagstuhl was finished. It is, I think, a beautiful modern architecture based on a traditional concept, a monasterial building. The architects saw Dagstuhl as a scientific monastery. Our monastery has a cloister, a very nice opportunity for arts exhibitions. But what got me really involved with fine art and not so fine artists was the procedure for equipping the new building with artistic objects. Germany has a law requiring that public buildings should be furnished with pieces of art. A certain percentage of

the construction money should go to the arts. A jury was set up, some groups of artists were asked to submit proposals. The architect and I were made members of the jury. When the submissions were discussed, I felt that something fishy was going on. I didn't know what. The jury, against my vote, selected some proposal that would deal with computer science in a pubertal way. I was quite upset and told the jury that this work would never see the center. I was declared a philistine, ignorant of modern trends in the fine arts. A four month battle behind the scenes led to the rejection of the jury's proposal by the minister in charge. As a revenge, the jury decided to let the money in the arts budget fall back to the construction budget. We were left with empty walls! I then invented an arts donation scheme, see `http://www.dagstuhl.de/en/about-dagstuhl/kunst/`, which, with a little help by our friends, has helped us to acquire quite a few nice pieces mostly from exhibitions we have had in the cloisters.

CC: What is the "job description" of the scientific director of the Leibniz Center for Informatics?

RW: The Scientific Director is responsible for the scientific program in Schloss Dagstuhl. That is the primary duty. Unlike a conference hotel, the Scientific Directorate, and the whole scientific staff at Schloss Dagstuhl feel responsible to guarantee high-quality meetings. The participants, who spend considerable effort to travel to this remote place, expect a high return for this travel investment. A disappointed participant will most likely not accept another invitation.

The Scientific Director chairs the Scientific Directorate at its meetings, moderates the discussion, and executes the decisions taken.

He also develops or takes up new directions and functions of the center. The Leibniz Center has extended its activities beyond the original function in several directions. It has become an open-access publisher. The high-quality conference series, LIPIcs, provides a low-cost, open-access alternative to established publishers, who, under financial pressure of their owners, were forced to change their publication policy to increase their revenue.

Another new direction is the cooperation with DBLP, the renowned bibliographic database established by Michael Ley at the University of Trier. The Leibniz Center has agreed to secure the long-term existence of this important source of information for computer science. With support from the Leibniz Association and the Klaus Tschira Foundation, DBLP has strongly increased the coverage of computer science publications.

CC: Over the years you have witnessed many interesting events in Schloss Dagstuhl. Are there any such memories which you would like to share with us?

RW: Let me report about two events, one rather sad, one positive. We scheduled a meeting on Computer Science and Astronomy at the time of last total solar eclipse

covering central Europe. This meeting included computer scientists, astronomers, and historians. As it brought together different communities that would hardly meet anywhere else it was a quite typical event for Dagstuhl.

One particular talk attempted to refute the then popular claim of some pseudo-historians that three centuries, around 700–1000 ad, had been invented. A historian had collected recordings about solar and lunar eclipses from that time. These were checked against an exciting software reproducing the planetary constellations at any time and any location. And indeed, all recorded eclipses were properly reproduced by this software. Another exciting experience at this event was that we selected exactly the right place to watch the eclipse. More or less all others in Britain, in France, and in Germany did not see anything due to rain and clouds while we had a 20 minutes hole in the clouds through which we could perfectly watch the eclipse.

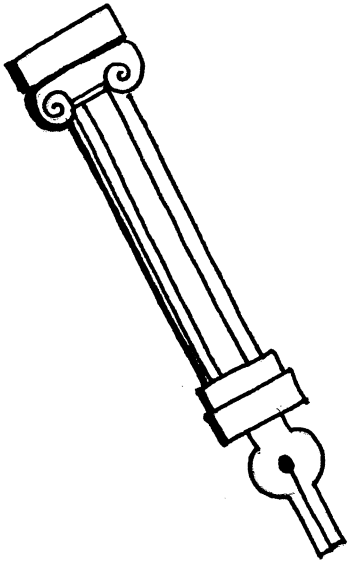
Now to the sad side. As we know from history, total solar eclipses were always seen as bringing with them mischief, catastrophes, and plagues. To support this old superstition, one participant had an accident coming to the meeting, one fell ill during the meeting, and one had to leave early because his father died.

As mentioned above, it is very common that Dagstuhl meetings bring together different communities that don't have any conference where they would meet. Dagstuhl thus often establishes absolutely necessary communication. Let me report about a meeting about Scheduling. Scheduling is an important topic, which occurs in manufacturing and in logistics—this is typically dealt with in the Operations Research community—, but also in computer science, and in computer science again in different subdomains, e.g. real-time scheduling, compilation, and algorithms. A meeting in 2010 brought together the algorithms community, the real-time scheduling community, and the operations-research community. Some real-time scheduling participants were asked to list their most interesting open problems, which were unknown to the algorithms community. They wrote up a report about their most urgent open problems, and in the proposal to the successor meeting the proposers proudly presented 10 publications that had resulted from this meeting solving at least 5 of the listed open problems of the real-time scheduling community.

CC: Many thanks.

The EATCS

Columns



The Computational Complexity Column

BY

V. Arvind

Institute of Mathematical Sciences, CIT Campus, Taramani
Chennai 600113, India
arvind@imsc.res.in
http://www.imsc.res.in/~arvind

The 1980's was a golden period for Boolean circuit complexity lower bounds. There were major breakthroughs. For example, Razborov's exponential size lower bound for monotone Boolean circuits computing the Clique function and the Razborov-Smolensky superpolynomial size lower bounds for constant-depth circuits with MOD_p gates for prime p. These results made researchers optimistic of progress on big lower bound questions and complexity class separations. However, in the last two decades, this optimism gradually turned into despair. We still do not know how to prove superpolynomial lower bounds for constant-depth circuits with MOD_6 gates for a function computable in exponential time.

Ryan Williams' exciting lower bound result of 2011, that nondeterministic exponential time does not have polynomial-size unbounded fanin constant-depth circuits with MOD_m gates for any composite m, has renewed optimism in the area. The best part is that his approach is potentially applicable to other lower bound questions.

In this wonderful article, Rahul Santhanam explores this theme of connections between improved SAT algorithms and circuit lower bounds.

Bulletin of the EATCS no 106, pp. 31–52, February 2012

Ironic Complicity: Satisfiability Algorithms and Circuit Lower Bounds

Rahul Santhanam
University of Edinburgh
rsanthan@inf.ed.ac.uk

Abstract

I discuss recent progress in developing and exploiting connections between SAT algorithms and circuit lower bounds. The centrepiece of the article is Williams' proof that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$, which proceeds via a new algorithm for ACC^0-SAT beating brute-force search. His result exploits a formal connection *from* non-trivial SAT algorithms *to* circuit lower bounds. I also discuss various connections in the reverse direction, which have led to improved algorithms for k-SAT, Formula-SAT and AC^0-SAT, among other problems.

1 Introduction

Theoretical computer science suffers from a dichotomy between the *algorithmic* endeavour and the *complexity-theoretic* endeavour. Algorithmists strive to design the most efficient algorithms for problems of interest, while complexity theorists investigate which problems are hard to solve, and why. Algorithmists focus on concrete problems, while complexity theorists often work in a more abstract framework, proving general theorems about computation. Algorithmists use constructive methods, while the enterprise of proving complexity lower bounds seems an inherently non-constructive one.

But is this dichotomy fundamental? At some level, algorithmists and complexity theorists are studying two sides of the same question: which is the most efficient solution for a problem? A priori, one would imagine that a deep understanding of the structure of a computational problem would assist both in designing the most efficient solution possible, as well as proving that no more efficient solution exists. In part because the theory of computation is still at a fairly early stage in its development, and in part because the basic questions seem to be very difficult, this has not often been the case so far. The algorithms community and

the complexity theory community have pursued their research programs more or less independently.

Recent developments have the potential to change this, opening the possibility of greater interaction and accelerated progress in both areas. These developments hint at a *complicity* between algorithms and lower bounds, which is ironic in that these endeavours seem superficially to be in opposition.

The most significant such development is the recent work of Williams [37, 38] proving that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$. This work has attracted a great deal of interest, since lower bound breakthroughs are rare. Though the result is interesting in itself, what is more interesting is the conceptual message of Williams' work, which is that algorithms for Satisfiability (SAT) can be used to prove lower bounds, and that there are strong connections between the two endeavours.

In this article, I give a sampler of work in the past couple of decades which shares this message. I make no claim that this is an exhaustive survey of the connections between SAT algorithms and lower bounds. Rather, I aim to give illustrations of the various connections that exist, and an indication of what the most promising research directions might be. This is a very actively growing area, and my hope is that this article could serve as a rough "road-map" for researchers wishing to work in this area, or else as a quick digest for those who are curious about the recent developments.

1.1 Historical Context

The connection between lower bounds and algorithms can be traced back to the pioneering work of Yao [39] and Blum & Micali [8] on pseudo-random generators. They showed how to construct cryptographic pseudo-random generators based on strong average-case circuit lower bounds. Cryptographic pseudo-random generators can be used to define sub-exponential time algorithms for problems in BPP, beating the trivial brute-force bound. Indeed, this implication was explicitly noted in Yao's paper [39].

Yao's connection is in a sense a byproduct of a conceptual machinery designed for cryptographic problems. In an influential paper, Nisan & Wigderson [28] adapted the notion of a pseudo-random generator to the context of complexity theory, and gave tighter implications from circuit lower bounds to pseudo-random generators, and vice versa. Since then, a sequence of papers [23, 26, 20], have established progressively tighter and more refined versions of these implications, and it is now known that circuit lower bounds for E (linear exponential time) against a class C of circuits are more or less equivalent to pseudo-random generators which are resilient to statistical tests from C, for essentially any natural class C of circuits. While pseudo-random generators imply improved deterministic simulations for problems in BPP, the converse is not the case. However,

Kabanets & Impagliazzo [24] have shown that sub-exponential time algorithms for the Polynomial Identity Testing (PIT) problem actually imply circuit lower bounds against arithmetic circuits. A weak converse of this result is known as well, showing a deep connection between algorithms and circuit lower bounds in this setting.

Though these results in the theory of pseudo-randomness are fairly strong, the connections haven't led to much progress either on lower bounds or on algorithms. The reason is that the known algorithmic ideas for solving PIT fall well short of having implications for pseudo-random generators, and hence for lower bounds. We won't discuss the pseudo-randomness literature further in this survey, but we note that it heavily influenced the formation of the connections we *will* discuss both historically, as well as methodologically.

There are other areas of theoretical computer science where progress on hardness results has gone hand-in-hand with new algorithms. This is the case, for example, with the recent work on semi-definite programming algorithms and the Unique Games conjecture [31], with the caveat that the notion of hardness there is conditional, i.e., based on reductions from presumed hard problems rather than on proven lower bounds. There is also the sophisticated and ambitious Geometric Complexity Theory (GCT) approach of Mulmuley & Sohoni [27] towards proving complexity lower bounds, which relies ultimately on algorithmic conjectures. We do not discuss these other examples of complicity between algorithms and lower bounds, but they do add to the evidence that there is something fundamental about this phenomenon.

1.2 Plan of the Article

Following on a short section establishing relevant notation, there are three main sections to this article discussing recent work, and a final section speculating on future research directions. The first section discusses a series of papers by Paturi, Zane and others proving structural theorems about CNF formulas which were then exploited both in an algorithmic context and to prove lower bounds. These were the earliest papers showing connections between exact algorithms for Satisfiability and circuit lower bounds. The middle section discusses the breakthroughs of Williams, which demonstrate and use a formal connection *from* SAT algorithms to lower bounds. The final section discusses various subsequent works which exploit connections in the reverse direction to give new and improved algorithms for variants of SAT such as Formula-SAT and AC^0-SAT.

Throughout this article, I will favour heuristic arguments over precise ones in cases where the former are more helpful in establishing intuition.

2 Preliminaries

I assume knowledge of the basic concepts of complexity theory. The book by Arora and Barak [1] and the Complexity Zoo (which can be found at the address `http://qwiki.caltech.edu/wiki/ComplexityZoo`) are good references.

I will be dealing with several variants of Satisfiability. For a positive integer k, k-SAT is the satisfiability problem for k-CNFs. CNF-SAT is the satisfiability problem for CNFs without any restriction on clause size. Formula-SAT is the satisfiability problem for formulas over the De Morgan basis. Circuit-SAT is the satisfiability problem for Boolean circuits. In general, given a class C of circuits, C-SAT is the satisfiability problem for circuits in C. I will refer simply to "SAT" when I wish to speak of the Satisfiability problem generally rather than of a specific variant.

Definition 1. *A parametric problem p-L consists of a language $L \subseteq \{0,1\}^*$ together with a parameter function $n : \{0,1\}^* \to \mathbb{N}$. Given a function $t : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$, we say that p-$L$ is solvable (resp. probabilistically solvable) in time t if there is a deterministic (resp. probabilistic) algorithm which decides L correctly and runs in time $t(|x|, n(x))$ on all inputs x.*

I will only be considering parametric versions of SAT variants, and for these problems there is a very natural notion of parameter: the number of variables in the formula. For any SAT variant L, p-L is the parametric problem corresponding to L.

The notion of "non-trivial" solvability of SAT can now be defined.

Definition 2. *A SAT variant L is said to have a non-trivial algorithm if p-L is solvable in time t, where $t(m,n) = O(\mathsf{poly}(m)2^{n-\omega(\log(n))})$.*

There is a natural notion of the "savings" an algorithm for SAT achieves over brute-force search. Note that the brute-force search algorithm operates in time $2^n\mathsf{poly}(m)$.

Definition 3. *Given a function $c : \mathbb{N}\times\mathbb{N} \to \mathbb{N}$, a SAT variant L is said to have savings (resp. probabilistic savings) c if p-L is solvable (resp. probabilistically solvable) in time t, where $t(m,n) = O(\mathsf{poly}(m)2^{n-c(m,n)})$.*

Thus a non-trivial algorithm achieves savings $\omega(\log(n))$, and $\mathsf{NP} = \mathsf{P}$ iff 3-SAT has savings $n - O(\log(n))$.

For information on the best known upper bounds for variants of SAT, refer to the survey by Dantsin and Hirsch [11]. Here I only discuss upper bound techniques and results which connect in some way to lower bounds.

However, it might be useful to say something about the common algorithmic paradigms for SAT. There are essentially two commonly used paradigms: the

DLL paradigm and the *local search* paradigm. Algorithms belonging to the DLL paradigm operate as follows. At each stage in the algorithm, a fixed rule is used to select a variable in the formula and a value to assign to it. With the variable set accordingly, the formula is simplified according to standard simplification rules, and the algorithm proceeds to the next stage. If at any stage, the formula simplifies to "true", the algorithm halts, since a satisfying assignment has been found. If it simplifies to "false", the algorithm "backtracks" by re-setting the most recently set variable to the other possible value and recursing. Intuitively, a DLL procedure explores a tree of candidate satisfying assignments, where nodes correspond to variables and edges to values which can be assigned to a given variable, with leaves being labelled "true" or "false". The procedure aims to construct and explore this tree in the most efficient possible manner, and the number of leaves of the tree gives a bound on the running time.

Algorithms belonging to the local search paradigm operate as follows. An initial assignment is chosen, and if this assignment is not already satisfying, the algorithm explores the space of assignments by changing the value of one variable at a time, with the variable whose value is to be changed determined by using some local measure of "progress". This exploration continues for a fixed number of steps, unless a satisfying assignment is found in the process. "Re-starts" are also allowed, with the algorithm choosing a new assignment and starting its exploration from scratch.

It seems as though other kinds of algorithmic ideas could potentially be useful as well, but there has been little rigorous analysis of alternatives to DLL and local search. One exception, jumping ahead, is Williams' algorithm for ACC^0-SAT [38], which uses dynamic programming.

3 Algorithms for k-SAT and Lower Bounds for Depth-3 Circuits

To the best of my knowledge, the first instance in the literature where a connection is explicitly drawn between upper bounds for SAT and circuit lower bounds is a paper by Paturi, Pudlak and Zane [30] giving probabilistic savings n/k for k-SAT. They also derandomize their algorithm to achieve savings $n/2k$. The inspiration for their algorithm and analysis is a lemma which they call the "Satisfiability Coding Lemma". This lemma is then used by them to give tight bounds for the circuit size of unbounded fan-in depth-3 circuits computing Parity.

Before describing their ideas, it might be good to step back a bit and give some general intuition for why there are connections between non-trivial SAT algorithms and circuit lower bounds. Suppose we wish to design a non-trivial al-

gorithm for C-SAT, where C is some natural class of circuits. For example, k-SAT corresponds to C being the class of depth-2 circuits with bottom fan-in bounded by k, and CNF-SAT corresponds to C being the class of depth-2 circuits. Intuitively, in order to design and analyze a non-trivial algorithm, we require some understanding of the structure of instances. Suppose we are able to isolate some special property that the instances to our problem share, eg., some property common to all k-CNFs, then we might be able to exploit this to achieve savings over brute-force search. The point is that the same property also indicates some *limitation* of the circuit class C under consideration, and by identifying a Boolean function f which does not have this property, we can prove a lower bound against C. Thus, it is fundamental to this connection between upper bounds and lower bounds that SAT is a *meta-algorithmic* problem - the instances to the problem are themselves computational objects, such as formulas or circuits.

Of course, the key to achieving good upper bounds as well as tight lower bounds is identifying the right property. The Satisfiability Coding Lemma shows that *isolated* solutions to k-CNFs have short descriptions on average, and hence that there can't be too many of them. Here an isolated solution is a satisfying assignment such that none of its neighbours in the Hamming cube are satisfying assignments to the same formula. Note that the property identified in the Satisfiability Coding Lemma is rather specialized. Parity, for example, has 2^{n-1} isolated solutions. Indeed Parity is in a sense the function that violates the property in the Satisfiability Coding Lemma most drastically, and intuitively this is why the Lemma is also useful in proving tight circuit size lower bounds for Parity.

To describe the Lemma more precisely, we need some notation. Given a formula ϕ on n variables and an integer $j, 0 \leqslant j \leqslant n$, call a satisfying assignment w to the variables of ϕ j-isolated if exactly j neighbours of y in the Hamming cube are not satisfying assignments to ϕ. An isolated solution is one that is n-isolated.

Lemma 4. *[30] There are polynomial-time computable functions Enc and Dec such that the following holds. Let ϕ be a k-CNF formula on n variables, and w be a j-isolated solution to the variables, where $0 \leqslant j \leqslant n$. Then $Dec(Enc(\phi, \pi, w)) = w$ for any permutation π on $[n]$, and moreover, on average over uniformly random choice of π, $|Enc(\phi, \pi, w)| \leqslant n - n/k$.*

The intuition behind the proof of Lemma 4 is that isolated solutions lead to many *critical* clauses. Given a solution w, a critical clause is one for which exactly one of the literals is true. An isolated solution w has at least n critical clauses, one for each assignment to a variable in w. If there were a variable without a critical clause corresponding to it, then flipping the value of that variable would result in a satisfying assignment, contradicting the fact that w is isolated.

Critical clauses can be used to save on variables when searching the space of solutions. Let w be an isolated solution. Imagine a process where variables are

chosen in a random order and set in ϕ to their value in w, excepting when there's a unit clause containing that variable. If there's a unit clause, the variable is set to satisfy that clause. The point is that if variables are chosen in random order, then for a critical clause of length k, there is a probability at least $1/k$ that the variable (say x) corresponding to the true literal in that clause is chosen last. In this case, the clause has already been reduced to a unit clause by the time x is set, and therefore x is *forced* rather than having to be set by w. So we don't need to store the value of x in w - in some sense, it can be recovered from the formula itself. Since there at least n critical clauses, on average at least n/k variables are forced in this process, and hence an isolated solution can be compressed to only store values of variables that are not forced, which saves n/k bits. In general, for a j-isolated solution, j/k bits are saved, using the same argument. This essentially gives the proof of Lemma 4.

It is easy to imagine how Lemma 4 can be used to achieve savings for Unique-k-SAT, the version of k-SAT where there's a promise that the input formula has either zero or one satisfying assignments. Clearly, any satisfying assignment in such a case is isolated, and hence it can be compressed on average. Intuitively, one just needs to search the compressed representations to find a solution if one exists, and this reduces the size of the search space to $2^{n-n/k}$ from 2^n.

A variation of this argument actually gives the same upper bound for k-SAT without any restriction on number of satisfying assignments. Consider a k-CNF ϕ. If there is a solution w which is j-isolated for large j, then it can be compressed by Lemma 4 and hence can be found much more quickly than brute-force search. If on the other hand, if all solutions are only j-isolated for small j, then intuitively there are *many* solutions, which means that a random solution is likely to work. In the paper by Paturi, Pudlak and Zane, this tradeoff idea is exploited nicely to prove the following result.

Theorem 5. *[30] k-SAT has probabilistic savings n/k.*

This was a huge improvement over the previous best known result for general k, which only gave savings $n/g(k)$ for some exponential function g. Because I wished to highlight how the algorithmic result takes advantage of the Satisfiability Coding Lemma, I focussed on the ideas in the analysis, and wasn't specific about the actual algorithm used. In fact, the algorithm designed by Paturi, Pudlak and Zane is a very natural and simple DLL algorithm. The algorithm repeatedly does the following: set the variables in ϕ in a random order to random values, except when there is a unit clause and the current variable is forced. It is no coincidence that this algorithm is similar to the encoding process used to prove Lemma 4!

Lemma 4 implies that there are at most $2^{n-n/k}$ isolated solutions to a k-CNF, and this can be used to give depth-3 circuit size lower bounds for Parity, where the circuits have bottom fan-in bounded by k. The argument is very simple: a

depth-3 circuit with bottom fan-in bounded by k is an OR of k-CNFs (the circuit can be assumed to have top gate OR without loss of generality). Since Parity has 2^{n-1} isolated solutions but each k-CNF can only have $2^{n-n/k}$ isolated solutions, the circuit needs to have at least $2^{n/k-1}$ gates. This bound is tight up to a constant factor. By a slightly more involved argument, Paturi, Pudlak and Zane show the following for general depth-3 circuits computing Parity.

Theorem 6. *[30] The depth-3 circuit size of Parity is* $\theta(n^{1/4}2^{\sqrt{n}})$.

The upper bound in Theorem 6 is given by a very natural divide-and-conquer strategy: break the variables up into blocks of size $\sqrt{n} - \log(n)/4$, compute the parity within each block, and then compute the parity of the resulting values.

Paturi, Pudlak, Saks and Zane [29] showed an improvement to Theorem 5 by using Resolution in a pre-processing step before applying the Paturi-Pudlak-Zane algorithm. Essentially, they try to increase the number of critical clauses in a formula. Note that if some variable in an isolated solution occurs in more than one critical clause, then in a random permutation of variables, the probability that it occurs last in *some* critical clause is larger than $1/k$, and so better compression of isolated solutions can be achieved than in Lemma 4. They prove that the repeated use of Resolution to derive all possible clauses of some bounded width (where the bound is $o(\log(n))$) from the original formula actually does yield benefits.

Theorem 7. *[29] For each* $k \geqslant 3$*, there is a constant* $\mu_k > 1$ *such that k-SAT has probabilistic savings* $\mu_k n/(k-1)$.

As with the Paturi-Pudlak-Zane result, the proof of this theorem gives a structural characterization of k-CNFs in terms of the maximum possible number of sufficiently isolated solutions. Here a sufficiently isolated solution is one such that there is no other solution within a given distance of it. This characterization was used to give the first depth-3 circuit size lower bound of the form $2^{c\sqrt{n}}$ for an explicit function, where $c > 1$.

Theorem 8. *[29] There is an explicit Boolean function* f *in* P *such that* f *does not have depth-3 circuits of size* $2^{\pi\sqrt{n}/\sqrt{6}-\sqrt{n}/\log(\log(n))}$.

A further example of a structural property of CNFs which is relevant both to algorithmic questions and to lower bounds is the Sparsification Lemma of Impagliazzo, Paturi and Zane [22] which says that every k-CNF can be written as the disjunction of $2^{\epsilon n}$ linear-sized k-CNFs, for arbitrarily small $\epsilon > 0$. I do not discuss this further here because the Sparsification Lemma does not directly give an improved algorithm for a natural variant of SAT. However, it has been quite influential in the structural theory of SAT, specifically with regard to the robustness of the Exponential Time Hypothesis (ETH), which states that 3-SAT is not solvable in time $2^{o(n)}$. It is also useful in proving certain kinds of depth-3 circuit lower bounds.

4 From Algorithms for Circuit-SAT to Circuit Lower Bounds

In the previous section, I described an informal connection between SAT algorithms and lower bounds - the Satisfiability Coding Lemma can be used both to analyze a natural algorithm for k-SAT and to prove tight lower bounds on the size of depth-3 circuits solving Parity. In this section, the spotlight is on the recent breakthroughs of Ryan Williams [37, 38]. Williams made two major contributions. First, he proved that non-trivial algorithms for C-SAT imply that $\mathsf{NEXP} \not\subseteq \mathsf{C}$ for a wide range of natural circuit classes C. This makes the connection between algorithms and circuit lower bounds formal, and also generic, in the sense that it opens up the possibility of using the algorithmic approach to prove a variety of new circuit lower bounds. Second, he gave a "proof-of-concept" for this novel approach by using it to show that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$, a brand-new circuit lower bound. This involved designing and analyzing a non-trivial algorithm for ACC^0-SAT.

To give intuition for the formal connection from SAT algorithms to circuit lower bounds, I first describe a simpler version of the result, which has an easy proof. Williams' connection is best understood as a refinement of this simpler result.

Suppose we have a polynomial-time algorithm for SAT. Then it is easy to see that EXP does not have polynomial-size circuits. If $\mathsf{EXP} \subseteq \mathsf{SIZE}(\mathit{poly})$, then by the classical Karp-Lipton-Meyer theorem [25] relating non-uniform inclusions of EXP to uniform collapses, $\mathsf{EXP} \subseteq \Sigma_2^p$. Now, by our assumption that SAT is in P, we have that $\mathsf{NP} = \mathsf{P}$, and hence that $\Sigma_2^p = \mathsf{P}$. But these collapses together imply that $\mathsf{EXP} = \mathsf{P}$, which is a contradiction to the deterministic time hierarchy theorem [18, 19]. Hence the assumption that $\mathsf{EXP} \subseteq \mathsf{SIZE}(\mathit{poly})$ must be false.

This is an example of an *indirect diagonalization* argument. An implication is proved by showing that its negation implies a contradiction to a hierarchy theorem. Such arguments have proven very useful in various contexts in structural complexity theory, including uniform lower bounds for the permanent [2], time-space tradeoffs [13, 12], a Karp-Lipton style result for NEXP [20] and separations against advice [6].

How far can this argument be stretched? If we try and use it to show that EXP does not have subexpontial-size circuits, we run into the issue that subexponential functions are not closed under composition. Indeed, if SAT is in SUBEXP, we have that $\mathsf{NP} \subseteq \mathsf{SUBEXP}$, but this does not imply that $\Sigma_2^p \subseteq \mathsf{SUBEXP}$. The best we can say is that $\Sigma_2^p \subseteq \mathsf{NSUBEXP}$, by replacing the inner co-nondeterministic polynomial-time part of a Σ_2^p computation with a deterministic subexponential-time computation. But this is not enough to derive a contradiction to a hierarchy theorem, as all we get using the additional assumption that $\mathsf{EXP} \subseteq \mathsf{SIZE}(\mathit{poly})$ is

that EXP $\subseteq$ NSUBEXP.

Perhaps we can salvage a superpolynomial size circuit lower bound for NEXP instead? Indeed this is the case. As hinted before, the analogue of the Karp-Lipton-Meyer theorem for NEXP is known - it was proved by Impagliazzo, Kabanets and Wigderson [20]. Their argument is a clever indirect one using pseudo-randomness in a critical way (though the statement of the result itself does not mention randomness!). At this point, we just need the result, not the proof technique. However, as we shall see, the Impagliazzo-Kabanets-Wigderson proof technique plays an important role in the derivation of Williams' connection.

Let us now re-do the old argument to establish a circuit lower bound from the weaker assumption that there is an algorithm for SAT running in time $2^{n^{o(1)}}$. The circuit lower bound we get from this assumption is that NEXP $\not\subseteq$ SIZE($poly$). Assume, to the contrary, that NEXP $\subseteq$ SIZE($poly$). Then, by the Impagliazzo-Kabanets-Wigderson result, we have that NEXP $= \Sigma_2^p$. Now, SAT in time $2^{n^{o(1)}}$ implies that NP $\subseteq$ SUBEXP, and therefore that $\Sigma_2^p \subseteq$ NSUBEXP. Combining this with the collapse for NEXP, we have that NEXP $\subseteq$ NSUBEXP, which is a contradiction to the non-deterministic time hierarchy theorem [10, 35, 41, 14].

The implication we have just proved is folklore. It wasn't given much significance because it does not represent a viable route to proving circuit lower bounds - few believe that SAT can be solved in sub-exponential time. Indeed, the Exponential-Time Hypothesis of Impagliazzo, Paturi and Zane [22] stating that 3-SAT cannot be solved in time $2^{o(n)}$ is widely believed.

On the surface, it doesn't look like there is much hope for getting an implication for circuit lower bounds from a much weaker algorithmic assumption for SAT, such as solvability in time $2^{n/2}$. Such a simulation seems "fragile" in that it doesn't compose with polynomial-time reductions to give a non-trivial simulation for all of NP, so it seems unlikely that the method of indirect diagonalization can be used.

However, it turns out that is is still possible to use the method, and a key factor in getting things to work is the parametric view of SAT, i.e., making a distinction between the size of the instance and the number of variables. Williams [37] proved the following theorem.

Theorem 9. *[37] If there is a non-trivial algorithm for Circuit-SAT, then* NEXP $\not\subseteq$ SIZE($poly$).

It is somewhat surprising that such a weak algorithmic assumption already yields lower bounds, and just the implication is interesting in itself. But what makes it more interesting is the possibility of actually proving circuit lower bounds this way. As per the current state of knowledge, there is no indication that Circuit-SAT is unlikely to have a non-trivial algorithm. After all, we are only asking to save over brute-force search by a superpolynomial factor in the running time.

Indeed, as it later turned out, a more general version of Theorem 9 yielded new lower bounds against ACC^0.

The proof of Theorem 9 combines several known facts and ideas in a clever way, including the completeness of the Succinct-3SAT problem for NEXP, local checkability and the easy witness method [20].

The high-level idea is still to use indirect diagonalization. Consider an arbitrary language $L \in \mathsf{NTIME}(2^n)$, and assume that $\mathsf{NEXP} \subseteq \mathsf{SIZE}(poly)$. We use the presumed non-trivial algorithm for Circuit-SAT to solve L non-deterministically in time $2^n/\omega(1)$. This contradicts the non-deterministic time hierarchy theorem, which has as a consequence the existence of a language L in $\mathsf{NTIME}(2^n)$ but not in $\mathsf{NTIME}(2^n/\omega(1))$.

Let x be an instance for the language L such that $|x| = n$. We first use the NEXP-completeness of the Succinct-3SAT problem to reduce x in polynomial time to a circuit C of size $\mathsf{poly}(n)$ with $n + O(\log(n))$ input bits. C implicitly encodes a 3CNF formula ϕ_C of size $2^n\mathsf{poly}(n)$ such that ϕ_C is satisfiable iff $x \in L$. By an implicit encoding here, we mean that given an index i into the binary representation of the formula ϕ_C, C outputs the i'th bit of the representation of ϕ_C.

We can't apply the presumed Circuit-SAT algorithm directly to ϕ_C since it is too large. Instead, we will work with the implicit encoding. The easy witness method [20] shows that if $\mathsf{NEXP} \subseteq \mathsf{SIZE}(poly)$, then every positive Succinct-SAT instance has a succinct witness, meaning that there is a circuit C' of size $\mathsf{poly}(n)$ and with $n + O(\log(n))$ inputs such that C' is the implicit encoding of a satisfying assignment to the formula encoded by the instance. Applying this to our context, we have that there is a circuit C' of size $\mathsf{poly}(n)$ which implicitly encodes a satisfying assignment to ϕ_C.

Now we can apply the guess-and-check paradigm: guess a circuit C' and check that the assignment encoded by C' indeed satisfies ϕ_C. The check that the assignment satisfies the formula can be done naturally in co-non-deterministic polynomial time: Universally guess a clause of ϕ_C and check using three calls to the circuit C' (each call recovering one bit of the succinct witness) that the clause is indeed satisfied by the assignment encoded by C'. The key point here is that this is a co-non-deterministic computation with only $n + O(\log(n))$ guess bits, since that many guess bits suffice to identify a clause of ϕ_C.

At this point, we use our algorithmic assumption and replace the co-non-deterministic computation by a deterministic one. Using the non-trivial algorithm for Circuit-SAT, we can implement the co-non-deterministic computation in time $2^n/\omega(1)$, since the co-non-deterministic computation is equivalent to solving a Circuit-SAT instance of size $\mathsf{poly}(n)$ with parameter $n + O(\log(n))$. By putting together the guess of the circuit C' with this computation, we get a non-deterministic algorithm which decides correctly whether $x \in L$ in time $2^n/\omega(1)$ as desired, yielding a contradiction to the non-deterministic hierarchy theorem.

Hopefully, this description clarifies how this argument is a much more refined version of the arguments giving the simpler implications. The Karp-Lipton-Meyer collapse appears here implicitly in our use of local checkability, and we use a much tighter version of the non-deterministic time hierarchy than is required for the simpler implications. The explicit use of the easy witness method is a new ingredient, though it appeared indirectly in our earlier argument since it underlies the Karp-Lipton-Meyer style collapse for NEXP [20].

Though Theorem 9 is interesting, it hasn't yielded any lower bounds yet as we do not know any non-trivial algorithms for Circuit-SAT. In the follow-up paper [38] which showed $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$, Williams significantly generalized Theorem 9 to apply to any circuit class satisfying some natural conditions.

Theorem 10. *[38] Let* C *be any circuit class which is closed under composition, contains* AC^0 *and is contained in the class of general Boolean circuits. If* C*-SAT has a non-trivial algorithm, then* NEXP *does not have polynomial-size circuits from* C*.*

Examples of classes C to which Theorem 10 applies include AC^0, ACC^0 and NC^1. Thus it gives a generic approach towards proving circuit lower bounds of interest.

Why doesn't the proof technique of Theorem 9 suffice to establish Theorem 10? The reason is that the reduction from $x \in L$ to a circuit C doesn't yield circuits that are structured enough. It is unclear whether the variant of Succinct-SAT where the circuits encoding the exponential-length formula are constant-depth circuits is still NEXP-complete. Williams gets around this by using the assumptions that C-SAT has a non-trivial algorithm and that NEXP has polynomial-size circuits from C a second time in a clever way.

More specifically, assume for the purpose of contradiction that NEXP has polynomial-size circuits from C, and that C-SAT has a non-trivial algorithm. Since C is a sub-class of Boolean circuits, we have that $\mathsf{NEXP} \subseteq \mathsf{SIZE}(poly)$. As before, we consider an arbitrary language $L \in \mathsf{NTIME}(2^n)$ and reduce a given instance x of L to a circuit C encoding an exponential-length CNF such that the CNF is satisfiable iff $x \in L$. The circuit C is not in general an ACC^0 circuit, and this is where the new idea comes in: we guess an *equivalent* polynomial-size ACC^0 circuit D and check during the co-non-deterministic computation that D is in fact equivalent to C by using local checkability together with the non-trivial algorithm for ACC^0-SAT. We also guess a polynomial-size ACC^0 circuit D' representing an "easy witness". The point is that since by assumption $\mathsf{NEXP} \subseteq \mathsf{ACC}^0$, we also have that $\mathsf{P} \subseteq \mathsf{ACC}^0$ and this implies that the circuits C and C' in the old proof have equivalent ACC^0 circuits D and D'. In the case of D, we actually need to check that it is equivalent to C, but as mentioned, this can be done using the algorithmic assumption. The rest of the argument is the same as before - once

we have D and D' which are ACC^0 circuits, the co-non-deterministic computation checking if the easy witness satisfies the formula encoded by D can be simulated deterministically in time $2^n/\omega(1)$ using the assumption of a non-trivial algorithm for ACC^0-SAT. Note that D and D' are guessed together, and the check of whether D is equivalent to C is performed before the check of whether the assignment encoded by D' satisfies the 3CNF encoded by D. What we get in the end is a non-deterministic algorithm for deciding x which runs in time $2^n/\omega(1)$, yielding a contradiction to the non-deterministic time hierarchy as before.

While Theorem 10, it could have been the case that for some fundamental reason, this approach to new lower bounds was not viable. Williams' greatest contribution was to give a "proof of concept" by using his approach to show that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$. The biggest circuit class for which super-polynomial size lower bounds were known for NEXP previously was $\mathsf{AC}^0[p]$ - the class of constant-depth circuits with modular counting gates where the modulus is a prime. In fact, the lower bounds against $\mathsf{AC}^0[p]$ are for explicit Boolean functions in P [32, 36], however the full power of NEXP seems necessary to achieve Williams' lower bound.

Williams' algorithm for ACC^0-SAT is innovative even from the algorithmic viewpoint, as it uses algorithmic ideas which hadn't been explored before in the context of algorithms for SAT. The first algorithm he came up with was a rather involved one using a result of Coppersmith about matrix multiplication. Following on a suggestion of Bjorklund, he later came up with a much simpler algorithm which uses dynamic programming, and this is the one I discuss. The algorithm relies on a well-known structural property of polynomial-size ACC^0 circuits [40, 9, 2] - the fact that they can be simulated by quasi-polynomial-size depth-2 SYM+ circuits. A SYM+ circuit is a circuit where the bottom layer is composed only of ANDs of small fan-in and the top gate is a symmetric gate. An additional property that is required is that these depth-2 SYM+ circuits can be constructed efficiently from the original ACC^0 circuits, and the top symmetric gate can be efficiently evaluated.

The algorithm is not non-trivial in the sense we defined before, but using the proof of Theorem 10, it does imply that $\mathsf{NEXP} \subseteq \mathsf{ACC}^0$ since it runs in time $2^{n-\omega(\log(n))}$ on circuits of size $\mathsf{poly}(n)$.

Theorem 11. *[38] There is an algorithm for p-ACC^0-SAT running in time $O(2^{n-n^{\Omega(1)}})$ when $m = \mathsf{poly}(n)$.*

I now sketch the proof. Let C be an ACC^0 circuit of size $m \leqslant n^c$ with n inputs, where c is a constant. Let $l < n$ be a parameter which will be fixed later. First, convert C to an equivalent circuit C' of size $m2^l$ on $t = n - l$ variables by enumerating all possible assignments on the first l variables and taking a big OR of the resulting 2^l copies of C. Note that C' is still an ACC^0 circuit. Let $s = m2^l$.

Next, convert C' to an equivalent depth-2 circuit C'' of size $s' = s^{\log^k(s)}$, where k is a constant. This can be done in time $O(s^{\log^{O(1)}(s)})$ using a result of Allender and Gore [2].

The key lemma is that a SYM+ circuit of size s' on t variables can be evaluated on all possible truth assignments to the variables in time $O((s' + 2^t)\mathsf{poly}(t))$. Note that this is superior to brute-force search in that the circuit size and the 2^t term are related additively rather than multiplicatively. This gives a significant advantage when the circuit size s' is large, as it is in our case.

Given the key lemma, we are done by choosing $l = n^\epsilon$ for ϵ sufficiently small. This is because, by the lemma, the SYM+ circuit can be evaluated on all possible truth assignments in time $O((2^{n^{\epsilon+k\epsilon+o(1)}} + 2^{n-n^\epsilon})\mathsf{poly}(n))$, which is $O(2^{n-n^\epsilon})$ when $\epsilon = 1/(k+2)$.

To prove the key lemma, we use dynamic programming. Essentially, we need to keep track of which AND gates evaluate to 1, in order to evaluate the symmetric function. We initialize a look-up table which states for every subset S of the input variables, the number $f(S)$ of AND gates which have precisely this subset as input. This initialization can be done in time $O((s' + 2^t)\mathsf{poly}(t))$. We then compute the *zeta transform* g of f using a standard dynamic programming algorithm, where for any subset T, $g(T)$ is the sum over all subsets $S \subseteq T$ of $f(T)$. For each T, $g(T)$ is the number of AND gates evaluating to 1 on the input which is 1 for precisely those input bits in T. This gives all the information required to evaluate the symmetric gate on that input. Thus we simultaneously obtain the answers of the circuit for all candidate assignments in time $O((s' + 2^t)\mathsf{poly}(t))$, proving the key lemma.

The Williams results raise the intriguing question of whether there are inherent barriers to proving lower bounds in this fashion. Progress on lower bounds using more traditional techniques has been halted by several barriers, including the relativization barrier [7], the natural proofs barrier [33] and the algebrization barrier [4]. None of these barriers seem to apply directly to the approach via algorithms. This is not necessarily cause for hope, but it is cause not to be pessimistic!

Of course, the viability of the approach depends on the existence of non-trivial algorithms (or algorithms at least good enough to be able to apply Theorem 10 for C-SAT, where C is a broader class of circuits than ACC^0. The jury is still out on this, but there's certainly a strong motivation now to develop the structural theory of the exact complexity of SAT variants, with the goal of understanding in which situations non-trivial algorithms are likely to exist.

5 Improved SAT Algorithms using Lower Bound Techniques

The results of Williams discussed in the previous section take advantage of a formal connection from algorithms to lower bounds. It is natural to ask whether there is a connection in the reverse direction - can lower bound techniques be used to design and analyze SAT algorithms?

In Section 4, I described structural properties of CNFs which were useful both in designing algorithms and proving lower bounds. The results in this section will have a slightly different flavour. Standard lower bound techniques will be used as inspiration to design SAT algorithms improving on brute-force search. No formal connection will be established, but using lower bounds as inspiration will have significant payoffs nevertheless.

While k-SAT and CNF-SAT have been widely studied, and improvements over brute-force search are known, until recently nothing non-trivial was known for Formula-SAT, where there is no restriction on the depth of the input formula. A year and a half ago, Santhanam [34] gave a simple deterministic algorithm which achieved savings $\Omega(n^3/m^2)$ for Boolean formulae over the de Morgan basis. Note that the savings is $\Omega(n)$ for linear-size formulae. Santhanam also gave a different algorithm which achieved savings $\Omega(n^2/(m\log(n)))$ on formulae over an arbitrary basis.

Theorem 12. *[34] Formula-SAT has savings $\Omega(n^3/m^2)$.*

The same savings applies to the problem of *counting* the number of satisfying assignments of a Boolean formula, using the same analysis.

The proof technique of Theorem 12 also yields a new lower bound consequence.

Corollary 13. *[34] Any linear-size sequence of formulae fails to compute Parity correctly on at least a $1/2-1/2^{\Omega(n)}$ fraction of inputs of length n, for all but finitely many n.*

The algorithm underlying the proof of Theorem 12 is very simple indeed. It is a DLL algorithm where the variable to be set is chosen as the most frequently occurring variable in the current formula, and the value to which it is set is chosen arbitrarily. This is a purely deterministic algorithm, however the analysis is probabilistic and uses the popular *random restriction* lower bound method as inspiration.

The random restriction method has been used to prove lower bounds in various settings, including for constant-depth circuits and Boolean formulae [3, 15, 16, 5, 17]. The basic idea is as follows. Suppose we are trying to prove a lower bound

against a class C of circuits. We look at what happens when a circuit from the class is "hit" with a random restriction, meaning that some of the variables are set in a specific way. For the present, we deal with *pure* random restrictions. A pure random restriction with parameter p is a probability distribution on partial assignments to inputs which sets each variable independently to 1 with probability $(1-p)/2$, to 0 with probability $(1-p)/2$ and leaves it unset with probability p. We try to argue that when a pure random restriction is applied to the inputs of a circuit from C, the circuit "simplifies" drastically. For constant-depth circuits, this is done using the Switching Lemma [16], which says that the induced function is constant with high probability, where the meaning of "high" depends on the choice of p. For Boolean formulae over the de Morgan basis, this is done by analyze the shrinkage exponent, which is the largest constant γ so that a formula of size L shrinks to a formula of size $O(p^\gamma L)$ under a restriction with parameter p. Subbotovskaya [5] proved that the shrinkage exponent is at least 1.5, and there was a sequence of papers obtaining improvements until Hastad proved that the shrinkage exponent is exactly 2 [17]. Indeed, the current best formula size lower bound of $n^{3-O(1)}$ for an explicit function is based on Hastad's result.

How do random restrictions connect to DLL algorithms? There is a superficial similarity in that processes involve variables being set incrementally, but in fact the connection goes deeper. In both processes, the notion of "simplification" is important. A DLL algorithm stops when the formula simplifies to "true" and backtracks when it simplifies to "false". The hope is that not too much backtracking is required before finding a satisfying assignment, if one exists. In the case of random restrictions, simplification of the formula is key to the technique being usable to prove lower bounds. The more drastic the simplification, the more limited the circuit class is, in some sense, and hence the better the lower bounds that can be shown. Quick simplification is also useful for DLL algorithms, as it means less backtracking and hence better savings over brute-force search.

This intuition can be made precise in the analysis of the DLL algorithm described above for FormulaSAT. We analyze a slightly different kind of random restriction - an *adaptive* restriction. In a pure restriction, the choice of which variables to set is made uniformly at random, and so too which values to set variables to. In an adaptive restriction, while the choice of values remains uniform, the choice of which variables to set is done adaptively depending on which variables are already set and how this setting has simplified the formula. It makes sense to study adaptive restrictions where the variables are set in the same order as they are set in the algorithm for FormulaSAT, as this gives a natural correspondence between properties of the restriction and efficiency of the algorithm. Subbotovskaya's analysis of pure random restrictions can be refined to show a *concentration bound* for simplification of formulae under such adaptive restrictions, and this concentration bound can then be used to bound the running time of

the DLL algorithm. Details can be found in the paper [34].

As with the results in Section 4, the analytical technique exposes a structural property of small formulae - they have *decision trees* that are not too large. This property can be exploited to prove Corollary 13, as it is easy to see that Parity requires decision trees of size 2^n. Indeed, any leaf of a decision tree that is not at depth n is uncorrelated with Parity, which is why this argument gives a strong correlation lower bound.

The random restriction method and the DLL algorithmic paradigm have both been the subject of much interest, so it is natural to wonder whether the connection between them can be exploited further. Santhanam conjectured that an analogous argument to his could yield an improved algorithm for AC^0-SAT, as well as new correlation bounds against AC^0 circuits. There has been a spate of recent work on this. Beame, Impagliazzo and Srinivasan (manuscript) have considerably improved an old correlation bound of Ajtai [3], and designed the current best deterministic algorithm for AC^0-SAT. Independently, Impagliazzo, Matthews and Paturi [21] came up with a probabilistic DLL algorithm for AC^0-SAT achieving savings close to linear.

Theorem 14. *[21]* AC^0*-SAT has probabilistic savings* $\Omega(n/(\log(m/n))^{d-1})$.

The analysis of the Impagliazzo-Matthews-Paturi algorithm extends and refines the Hastad switching lemma, and gives a new structural characterization of AC^0 functions in terms of partitions of the Hamming cube into subcubes where the function is constant. An optimal correlation bound for Parity against constant-depth circuits follows from this characterization, in a similar way to how Corollary 13 follows from Theorem 12.

Corollary 15. *[21]* AC^0 *circuits of size s fail to compute Parity correctly on at least a* $1/2 - 1/2^{\Omega(n/(\log(m/n))^{d-1})}$ *fraction of inputs, for n large enough.*

A similar correlation bound was obtained independently by Hastad (manuscript). The above results exploit a connection between DLL algorithms and random restrictions. Are there other lower bound techniques that can be harnessed algorithmically? This is an intriguing question about which little is known. Santhanam's algorithm for formulae over an arbitrary basis can be interpreted as utilizing a connection between the algorithmic paradigm of *memoization* and the Neciporuk lower bound technique in complexity theory, but I do not know of any other results along this direction.

6 Speculation

The recent papers on SAT algorithms and lower bounds have opened up what promises to be a very fruitful area of research. There are many research directions

that look interesting, and in this section I will give a personal selection.

Perhaps the most exciting questions arise from the work of Williams. His lower bound against ACC^0 circuits is for a Boolean function in NEXP. The lower bounds we know against weaker classes are all for functions in P. This is a major discrepancy - can we prove a similar lower bound for a much more explicit function? It seems that techniques somewhat different from Williams' will be required. Perhaps the limitations of the circuit class ACC^0 which are exposed by his algorithm for ACC^0-SAT could be exploited in a more direct fashion, giving a more explicit bound.

Another very natural question is to derive lower bounds against larger classes of circuits. This motivates the exploration of new algorithmic paradigms for SAT, such as dynamic programming and graph sparsification.

In terms of the reverse connection from lower bounds to algorithms, it would be interesting to identify if there is any "algorithmic content" in other common lower bound techniques such as the polynomial method and the Khrapchenko method. New analyses for DLL algorithms have been found by constructivizing the proofs that random restrictions simplify formulae, and perhaps other lower bound proofs could be constructivized in a similar way. In an optimistic scenario, this would lead to new algorithmic methods that could be used elsewhere.

In the Boolean complexity world, the connections between algorithms and lower bounds have only been studied so far in the context of the Satisfiability problem. There are various other NP-hard problems, such as Clique, Colouring, Subset Sum etc. for which improved algorithms beating brute-force search are an active topic of study. Could any of the lower bound connections help in analyzing these problems? An immediate obstacle to doing this is that none of these problems are inherently meta-algorithmic, unlike SAT. But maybe the use of alternative notions of complexity, such as graph complexity, could provide some insight here.

Connections analogous to those in the Boolean complexity setting could exist in the arithmetic complexity setting as well. Specifically, it is quite conceivable that algorithms for the Polynomial Identity Testing problem marginally beating brute force search could lead to new arithmetic complexity lower bounds, and this possibility ought to be explored further.

To reiterate, the complicity between lower bounds and algorithms could provide a way around the obstacles to which complexity theorist, and to a lesser extent algorithmists, are so accustomed. But the maps we can draw at this stage are of necessity rough, unformed. All we can do is to believe that the deep mysteries mask a deeper sense.

References

[1] S. Arora and B. Barak. *Complexity Theory: A Modern Approach*. Cambridge University Press, Cambridge, 2009.

[2] Eric Allender and Vivek Gore. A uniform circuit lower bound for the permanent. *SIAM Journal on Computing*, 23(5):1026–1049, 1994.

[3] Miklos Ajtai. Σ_1^1-formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.

[4] Scott Aaronson and Avi Wigderson. Algebrization: A new barrier in complexity theory. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing (STOC '08)* 731–740, 2008.

[5] B.A.Subbotovskaya. Realizations of linear functions by formulas using and, or, not. *Soviet Mathematics Doklady*, (2):110–112, 1961.

[6] Harry Buhrman, Lance Fortnow, and Rahul Santhanam. Unconditional lower bounds against advice. In *Proceedings of 36th International Colloquium on Automata, Languages and Programming*, pages 195–209, 2009.

[7] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. *SIAM Journal on Computing*, 4(4):431–442, 1975.

[8] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequence of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[9] Richard Beigel and Jun Tarui. On ACC. *Computational Complexity*, 4:350–366, 1994.

[10] Stephen Cook. A hierarchy for nondeterministic time complexity. In *Conference Record, Fourth Annual ACM Symposium on Theory of Computing*, pages 187–192, Denver, Colorado, 1–3 May 1972.

[11] Evgeny Dantsin and Edward Hirsch. Worst-case upper bounds. In H.van Maaren A.Biere, M.Heule and T.Walsh, editors, *Handbook of Satisfiability*. 2008.

[12] Lance Fortnow, Richard Lipton, Dieter van Melkebeek, and Anastasios Viglas. Time-space lower bounds for satisfiability. *Journal of the ACM*, 52(6):833–865, 2005.

[13] L. Fortnow. Time-space tradeoffs for satisfiability. *Journal of Computer and System Sciences*, 60(2):337–353, April 2000.

[14] Lance Fortnow and Rahul Santhanam. Robust simulations and significant separations. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming*, pages 569–580, 2011.

[15] Merrick Furst, James Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, April 1984.

[16] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing*, pages 6–20, 1986.

[17] Johan Hastad. The shrinkage exponent of de morgan formulas is 2. *SIAM Journal on Computing*, 27(1):48–64, 1998.

[18] Juris Hartmanis and Richard Stearns. On the computational complexity of algorithms. *Trans. Amer. Math. Soc. (AMS)*, 117:285–306, 1965.

[19] Frederick Hennie and Richard Stearns. Two-tape simulation of multitape Turing machines. *Journal of the ACM*, 13(4):533–546, October 1966.

[20] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *Journal of Computer and System Sciences*, 65(4):672–694, 2002.

[21] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC0. In *Proceedings of Symposium on Discrete Algorithms*, page To appear, 2012.

[22] Russell Impagliazzo, Ramamohan Paturi, and Francis Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 62(4):512–530, 2001.

[23] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the 29th Annual ACM Symposium on the Theory of Computing*, pages 220–229, 1997.

[24] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the 35th Annual ACM Symposium on the Theory of Computing*, pages 355–364, 2003.

[25] Richard Karp and Richard Lipton. Turing machines that take advice. *L'Enseignement Mathématique*, 28(2):191–209, 1982.

[26] Adam Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial hierarchy collapses. *SIAM Journal of Computing*, 31(5):1501–1526, 2002.

[27] Ketan Mulmuley. On p vs np and geometric complexity theory: dedicated to sri ramakrishna. *Journal of the Association of Computing Machinery*, 58(2), 2011.

[28] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[29] Ramamohan Paturi, Pavel Pudlak, Mike Saks, and Francis Zane. An improved exponential-time algorithm for k-sat. In *Proceedings of 39th International Symposium on Foundations of Computer Sciece (FOCS)*, pages 628–637, 1998.

[30] Ramamohan Paturi, Pavel Pudlak, and Francis Zane. Satisfiability coding lemma. In *Proceedings of 38th International Symposium on Foundations of Computer Science (FOCS)*, pages 566–574, 1997.

[31] Prasad Raghavendra. Optimal algorithms and inapproximability results for every csp? In *ACM Symposium on Theory of Computing (STOC)*, pages 245–254, 2008.

[32] Alexander Razborov. Lower bounds on the size of bounded-depth networks over the complete basis with logical addition. *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987.

[33] Alexander Razborov and Steven Rudich. Natural proofs. *Journal of Computer and System Sciences*, 55(1):24–35, 1997.

[34] Rahul Santhanam. Fighting perebor: New and improved algorithms for formula and QBF satisfiability. In *Proceedings of 51st Annual IEEE Symposium on Foundations of Computer Science*, pages 183–192, 2010.

[35] Joel Seiferas, Michael Fischer, and Albert Meyer. Separating nondeterministic time complexity classes. *Journal of the ACM*, 25(1):146–167, January 1978.

[36] Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual Symposium on Theory of Computing*, pages 77–82, 1987.

[37] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proceedings of the 42nd Annual ACM Symposium on Theory of Computing*, pages 231–240, 2010.

[38] Ryan Williams. Non-uniform ACC circuit lower bounds. In *Proceedings of 26th Annual IEEE Conference on Computational Complexity*, pages 115–125, 2011.

[39] Andrew Yao. Theory and application of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*, pages 80–91, 1982.

[40] Andrew Yao. On ACC and threshold circuits. In *Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, pages 619–627, 1990.

[41] Stanislav Žák. A Turing machine time hierarchy. *Theoretical Computer Science*, 26(3):327–333, October 1983.

T D C C

P F

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

Bulletin of the EATCS no 106, pp. 53–77, February 2012

UNDERSTANDING NON-UNIFORM FAILURE MODELS

Petr Kuznetsov
TU Berlin/Deutsche Telekom Laboratories
petr.kuznetsov@tu-berlin.de

Abstract

Traditionally, models of fault-tolerant distributed computing assume that failures are "uniform": processes are equally probable to fail and a failure of one process does not affect reliability of the others. In real systems, however, processes may not be equally reliable. Moreover, failures may be correlated because of software or hardware features shared by subsets of processes. In this paper, we survey recent results addressing the question of what can and what cannot be computed in systems with non-identical and non-independent failures.

L'égalité sera peut-être un droit, mais aucune puissance humaine ne saura le convertir en fait.[1]

Honoré de Balzac

1 Introduction

A distributed system is a collection of computing units, called *processes*. The principal challenge of distributed computing is to devise protocols that correctly operate in the presence of failures of processes and asynchrony. A *failure model* describes the assumptions on where and when failures might occur. The classical "uniform" failure model assumes that processes fail with equal probabilities, independently of each other. This enables reasoning about the maximal number of processes that may, with a non-negligible probability, fail in any given execution of the system. It is natural to ask questions of the kind: what problems can be solved *t-resiliently*, i.e., assuming that at most *t* processes may fail. In particular,

[1] Equality may perhaps be a right, but no human power can ever turn it into a fact.

the *wait-free* ($(n-1)$-resilient, where n is the number of processes) model assumes that any subset of processes may fail.

However, in real systems, processes do not always fail in the uniform manner. Processes may be unequally reliable and prone to correlated failures. A software bug makes all processes using the same build vulnerable, a router's failure may makes all processes behind it unavailable, a successful malicious attack on a given process increases the chances to compromise processes running the same software, etc. Thus, understanding how to deal with non-uniform failures is crucial.

Adversaries. Consider a system of three processes, p, q, and r. Suppose that p is very unlikely to fail, and otherwise, all failure patterns are allowed. Since we only exclude executions in which p fails, the set of correct processes in any given execution must belong to $\{p, pq, pr, pqr\}$[2].

Now we give an example of correlated failures. Suppose that p and q share a software component x, p and r share a software component y, and q and r are built atop the same hardware platform z (Figure 1). Further, let x, y, and z be prone to failures, but suppose that it is very unlikely that two failures occur in the same execution. Hence, the possible sets of correct processes in our system are $\{pqr, p, q, r\}$.

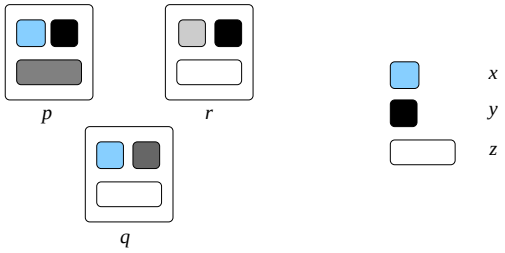


Figure 1: A system modeled by the adversary $\{pqr, p, q, r\}$: p and q share component x, p and r share component y, and q and r run atop the same hardware platform z.

The notion of a generic *adversary* introduced by Delporte et al. [9] intends to model such scenarios. An adversary $\mathcal{A}$ is defined as a set of possible correct process subsets. E.g., the *t-resilient* adversary $\mathcal{A}_{t\text{-}res}$ in a system of n processes consists of all sets of $n - t$ or more processes. We say that an execution is $\mathcal{A}$-*compliant* if the set of processes that are correct in that execution belongs to $\mathcal{A}$. Thus, an adversary $\mathcal{A}$ describes a model consisting of $\mathcal{A}$-compliant executions.

[2]For brevity, we simply write pqr when referring to the set $\{p, q, r\}$.

The formalism of adversaries [9] assumes that processes fail only by crashing, and adversaries only specify the *sets* of processes that may be correct in an execution, regardless of the timing of failures. Of course, this sorts out many kinds of possible adversarial behavior, such as malicious attacks or timing failures. However, it is probably the simplest model that still captures important features of non-uniform failures.

Distributed tasks. In this paper, we focus on a class of distributed-computing problems called *tasks*. A task can be seen as a distributed variant of a function from classical (centralized) computing: given a distributed input (an *input vector*, specifying one input value for every process) the processes are required to produce a distributed output (an *output vector*, specifying one output value for every process), such that the input and output vectors satisfy the given *task specification*.

The classical theory of computational complexity theory categorizes functions based on their inherent difficulty (e.g., with respect to solving them on a Turing machine). In the distributed setting, the difficulty in solving a task also depends on the adversary we are willing to consider. There are tasks that can be trivially solved on a Turing machine, but are not solvable in the presence of some distributed adversaries. For example, the fundamental task of *consensus*, in which the processes must agree on one of the input values, cannot be solved assuming the 1-resilient adversary $\mathcal{A}_{1\text{-}res}$ [11, 28]. More generally, the task of k-set consensus [8], where every correct process is required to output an input value so that at most k different values are output, cannot be solved in the presence of $\mathcal{A}_{k\text{-}res}$ [21, 30, 4].

Most of this paper deals with *colorless* tasks (also called convergence tasks [5]). Informally, colorless tasks allow every process to adopt an input or output value from any other participating process. Colorless tasks include consensus [11], k-set consensus [8] and simplex agreement [22].

The relative power of an adversary. This paper primarily addresses the following question. Given a task T and an adversary $\mathcal{A}$, is T solvable in the presence of $\mathcal{A}$?

Intuitively, the more sets an adversary comprises, the more executions our system may expose, and, thus, the more powerful is the adversary in "disorienting" the processes. In this sense, the *wait-free* adversary $\mathcal{A}_{wf} = \mathcal{A}_{n-1\text{-}res}$ is the most powerful adversary, since it describes the set of *all* possible executions.

In contrast, a "singleton" adversary $\mathcal{A} = \{S\}$ that consists of only one set $S \subseteq \mathcal{P}$ is very weak. For example, we can use any process in S as the "leader" that never fail. This allows us to solve consensus or implement any sequential data type [18].

But in general, there are exponentially many adversaries defined for n processes that are not related by containment. Therefore, it is difficult to say a priori which of two given adversaries is stronger.

Superset-closed adversaries. We start with recalling the model of *dependent failures* proposed by Junqueira and Marzullo [25], defined in terms of *cores* and *survivor sets*. In brief, a survivor set is a minimal subset of processes that can be the set of correct processes in some execution, and a core is a minimal set of processes that do not all fail in any execution.

We show that, in fact, the formalism of [25] describes a special class of *superset-closed* adversaries: every superset of an element of such an adversary $\mathcal{A}$ is also an element of $\mathcal{A}$. The minimal elements of $\mathcal{A}$ (no subset of which are in $\mathcal{A}$) are the survivor sets of the resulting model.

It turns out that the power of a superset-closed adversary $\mathcal{A}$ in solving colorless tasks is precisely characterized by the size of its minimal core, i.e., the minimal-cardinality set of processes that cannot all fail in any $\mathcal{A}$-compliant execution. A superset-closed adversary with minimal core size c allows for solving a colorless task T if and only if T can be solved $(c-1)$-resiliently. In particular, if $c = 1$, then any task can be solved in the presence of $\mathcal{A}$, and if $c = n$, then $\mathcal{A}$ only allows for solving wait-free solvable tasks. Thus, all superset-closed adversaries can be categorized in n classes, based on their minimal core sizes.

We present two ways of deriving this result: first, using the elements of modern topology (proposed by Herlihy and Rajsbaum [20]) and second, through shared-memory simulations (proposed by Gafni and Kuznetsov [16]).

Characterizing generic adversaries. The dependent-failure formalism of [25] is however not expressive enough to capture the task solvability in generic non-uniform failure models. It is easy to construct an adversary that has the minimal core size n but allows for solving tasks that can cannot be wait-free solved. One example is the "bimodal" adversary $\{pqr, p, q, r\}$ (Figure 1) that allows for solving 2-set consensus.

Therefore, to characterize the power of a generic adversary, we need a more sophisticated criterion than the minimal core size. Surprisingly, such a criterion, that we call *set consensus power*, is not difficult to find. Suppose that we can partition an adversary $\mathcal{A}$ into k sub-adversaries, each powerful enough to solve consensus. We conclude that $\mathcal{A}$ allows for solving k-set consensus: simply run k consensus algorithms in parallel, each assuming a distinct sub-adversary. Moreover, we show that the set consensus power of $\mathcal{A}$, defined as the minimal such number of sub-adversaries, precisely characterizes the power of $\mathcal{A}$ in solving colorless tasks.

Therefore, generic adversaries defined on n processes can still be split into n equivalence classes. Each class j consists of adversaries of set consensus power j that agree on the set of colorless tasks they allow for solving: namely, tasks that can be solved $(j-1)$-resiliently and not j-resiliently. In particular, class n contains adversaries that only allow for solving tasks that can be solved wait-free, and class 1 allows for solving consensus and, thus, any task.

Roadmap. We begin with a background section that states recalls the basics of our model and the notion of a distributed task. Then we discuss several approaches to model non-uniform failures: dependent failure model of Junqueira and Marzullo [25], adversaries of Delporte et alii [9], and asymmetric progress conditions by Imbs et alii [24].

Then we present a complete characterization of superset-closed adversaries. The result is first shown using elements of combinatorial topology [20] and then through simple shared-memory simulations [16].

We then characterize generic (not necessarily superset-closed) adversaries using the notion of set consensus power and relate it with the *disagreement power* proposed by Delporte et alii [9].

We conclude with a brief overview of open questions, primarily related to solving generic (not necessarily colorless) tasks in the presence of generic (not necessarily superset-closed) adversaries.

The results described in this paper originally appeared in [9, 14, 20, 16, 24, 31].

2 Background

In this section, we briefly state our system model and recall the notion of a distributed task and two important constructs used in this paper: Commit-Adopt and BG-simulation.

2.1 Model

We consider a system Π of n processes, $p_1, \ldots, p_n$, that communicate via reading and writing in the shared memory. We assume that the system is *asynchronous*, i.e., relative speeds of the processes are unbounded. Without loss of generality, we assume that processes share an *atomic snapshot* memory [1], where every process may update its dedicated element and take atomic snapshot of the whole memory.

A process may only fail by crashing, and otherwise it must respect the algorithm it is given. A *correct* process never crashes.

2.2 Tasks

In this paper, we focus on a specific class of distributed computing problems, called *tasks* [22]. In a distributed task [22], every participating process starts with a unique input value and, after the computation, is expected to return a unique output value, so that the inputs and the outputs across the processes satisfy certain properties. More precisely, a *task* is defined through a set $\mathcal{I}$ of input vectors (one input value for each process), a set O of output vectors (one output value for each process), and a total relation $\Delta : \mathcal{I} \mapsto 2^O$ that associates each input vector with a set of possible output vectors. An input $\perp$ denotes a *not participating* process and an output value $\perp$ denotes an *undecided* process.

For example, in the task of *k-set consensus*, input values are in $\{\perp, 0, \ldots, k\}$, output values are in $\{\perp, 0, \ldots, k\}$, and for each input vector I and output vector O, $(I, O) \in \Delta$ if the set of non-$\perp$ values in O is a subset of values in I of size at most k. The special case of 1-set consensus is called *consensus* [11].

We assume that every process runs a *full-information* protocol: initially it writes its input value and then alternates between taking snapshots of the memory and writing back the result of its latest snapshots. After a certain number of such asynchronous rounds, a process may gather enough state to *decide*, i.e., i.e., to produce an irrevocable non-$\perp$ output value.

In *colorless* task (also called *convergence* tasks [5]) processes are free to use each others' input and output values, so the task can be defined in terms of input and output *sets* instead of vectors.[3] The k-set consensus task is colorless.

Note that to solve a colorless task, it is sufficient to find a protocol (a decision function) that allows just one process to decide. Indeed, if such a protocol exists, we can simply convert it into a protocol that allows every correct process to decide: every process simply applies the decision function to the observed state of any other process and adopts the decision.

2.3 The Commit-Adopt protocol

One tool extensively used in this paper is the *commit-adopt* abstraction (CA) [12]. CA exports one operation *propose*(v) that returns $(commit, v')$ or $(adopt, v')$, for $v', v \in V$, and guarantees that

(a) every returned value is a proposed value,

(b) if only one value is proposed then this value must be committed,

[3] Formally, let $val(U)$ denote the set of non-$\perp$ values in a vector U. In a colorless task, for all input vectors I and I' and all output vectors O and O', such that $(I, O) \in \Delta$, $val(I) \subseteq val(I')$, $val(O') \subseteq val(O)$, we have $(I', O) \in \Delta$ and $(I, O') \in \Delta$.

(c) if a process commits on a value v, then every process that returns adopts v or commits v, and

(d) every correct process returns.

The CA abstraction can be implemented wait-free [12]. Moreover, CA can be viewed as a way to establish *safety* in shared-memory computations.

For example, consider a protocol where every processes goes through a series of instances of commit-adopt protocols, $CA_1, CA_2, \ldots$, one by one, where each instance receives a value adopted in the previous instance as an input (the initial input value for CA_1). One can easily see that once a value v is committed in some CA instance, no value other than v can ever be committed (properties (a) and (c) above). One the other hand, if at most one value is proposed to some CA instance, then this value must be committed by every process that takes enough steps (property (b) above).

This algorithm can be viewed as a *safe* version of consensus: every committed value is a proposed value and no two processes commit on different values (properties (a), (b) and (c) above). Given that every correct process goes from one CA instance to the other as long as it does not commit (property (d) above), we can boost the liveness guarantees of this protocol using external oracles.

In fact, the algorithm *per se* guarantees termination in every *obstruction-free* execution, i.e., assuming that eventually at most one process is taking steps. Moreover, we can build a consensus algorithm that terminates *almost always* if we allow processes to toss coins when choosing an input value for the next CA instance [2]. Also, if we allow a process to access an *oracle* (e.g., the Ω failure detector of [6]) that eventually elects a correct leader process, we get a live consensus algorithm.

2.4 The BG-simulation technique.

Another important tool used in this paper is *BG-simulation* [4, 5]. BG-simulation is a technique by which $k+1$ processes $s_1, \ldots, s_{k+1}$, called *simulators*, can wait-free simulate a *k-resilient* ($\mathcal{A}_{k\text{-}res}$-compliant) execution of any protocol *Alg* on m processes $p_1, \ldots, p_m$ ($m > k$). The simulation guarantees that each simulated step of every process p_j is either agreed upon by all simulators, or one less simulator participates further in the simulation for each step which is not agreed on.

The central building block of the simulation is the *BG-agreement* protocol. BG-agreement reminds consensus: processes propose values and agree one of the proposed values at the end. Indeed, the BG-agreement protocol ensures safety of consensus—every decided value was previously proposed, and no two different values are decided— but not liveness. If one of the simulators slows down while

executing BG-agreement, the protocol's execution at other correct simulators may "block" until the slow simulator finishes the protocol. If the slow simulator is faulty, no other simulator is guaranteed to decide.

Suppose the simulation tries to promote $m > k$ simulated processes in a fair (e.g., round-robin) way. As long there is a live simulator, at least $m - k$ simulated processes performs infinitely many steps of *Alg* in the simulated execution.

Recently the technique of BG-simulation was extended to show that any colorless task that can be solved assuming the $(k-1)$-resilient adversary can also be solved using read-write registers and k-set consensus objects [13].

3 Non-uniform failures in shared-memory systems

In this section, we overview several approaches to model non-uniform failures: dependent failure model of Junqueira and Marzullo [25], adversaries of Delporte et alii [9], and asymmetric progress conditions by Imbs et alii [24] and Taubenfeld [31].

3.1 Survivor Sets and Cores

Junqueira and Marzullo [26, 25] proposed to model non-uniform failures using the language of *survivor sets* and *cores*. A survivor set $S \subseteq \Pi$ if a set of processes such that:

(a) in some execution, S is the set of correct processes, and

(b) S is minimal: for every proper subset S' of S, there is no execution in which S' is the set of correct processes.

A collection $\mathcal{S}$ of survivor sets describes a system such that the set of correct processes in every execution contains a set in $\mathcal{S}$.

Respectively, a *core* C is a set of processes such that:

(a) in every execution, some process in C is correct, and

(b) C is minimal: for every proper subset C' of C, there is an execution in which every process in C' fails.

Thus, a core is a minimal set of processes that cannot be all faulty in any execution of our system. Note that the set of cores is unambiguously determined by the set of survivor sets.

A core is actually a *minimal hitting set* of the set system built of survivor sets, and a core of smallest size is a corresponding minimum hitting set. Determining minimum hitting set of a set system is known to be NP-complete [27].

The language of cores [26, 25] proved to be convenient in understanding the ability of a system with non-uniform failures to solve consensus or build a fault-tolerant replicated storage.

3.2 Adversaries

A more general way to model non-uniform failures was proposed by Delporte et al. [9]. Formally, an *adversary* defined for a set of processes Π is a non-empty set of process subsets $\mathcal{A} \subseteq 2^{\Pi}$. We say that an execution is $\mathcal{A}$-*compliant* if the *correct set*, i.e., the set of correct processes, in that execution belongs to $\mathcal{A}$. Thus, assuming an adversary $\mathcal{A}$, we only consider the set of $\mathcal{A}$-*compliant* executions. [4] By convention, we assume that in every execution, at least one process is correct, i.e., no adversary contains $\emptyset$.

Given a task T and an adversary $\mathcal{A}$, we say that T is $\mathcal{A}$-*resiliently solvable* if there is a protocol such that in every execution, the outputs match the inputs with respect to the specification of T, and in every $\mathcal{A}$-compliant execution, each correct process eventually produces an output.

It is easy to see that the language of survivor sets of [25] describes a special class of *superset-closed* adversaries. Formally, the set $\mathcal{SC}$ of superset-closed adversaries consists of all $\mathcal{A}$ such that for all $S \in \mathcal{A}$ and $S \subseteq S' \subseteq \Pi$, we have $S' \in \mathcal{A}$.

For example, consider the t-resilient adversary $\mathcal{A}_{t\text{-}res} = \{S \subseteq \Pi, |S| \geq n - t\}$. By definition, $\mathcal{A}_{t\text{-}res} \in \mathcal{SC}$. The survivor sets of $\mathcal{A}_{t\text{-}res}$ are all sets of $n-t$ processes, and the cores are all sets of $t+1$ processes. The $(n-1)$-resilient adversary $\mathcal{A}_{WF} = \mathcal{A}_{n-1\text{-}res}$ is also called *wait-free*. An $\mathcal{A}_{WF}$-resilient task solution must ensure that every process obtains an output in a finite number of its own steps, regardless of the behavior of the rest of the system.

Another example $\mathcal{A}_{L_p} = \{S \subseteq \Pi | p \in S\} \in \mathcal{SC}$ describing a system in which p never fails. $\mathcal{A}_{L_p}$ has one survivor set $\{p\}$ and one core $\{p\}$. Intuitively, p may then act as a correct leader in a consensus protocol. Thus, every task can be solved in the presence of $\mathcal{A}_{L_p}$ [18].

The *k-obstruction-free* adversary $\mathcal{A}_{k\text{-}OF}$ is defined as $\{S \subseteq \Pi \mid 1 \leq |S| \leq k\}$. In particular, $\mathcal{A}_{OF} = \mathcal{A}_{1\text{-}OF}$ allows for solving consensus [10]. Clearly, $\mathcal{A}_{k\text{-}OF}$ for $1 \leq k < n$ is not in $\mathcal{SC}$.

The "bimodal" adversary $\{pqr, p, q, r\}$ (Figure 1) is not in $\mathcal{SC}$ either: it contains the singleton p but not its supersets pq and pr.

[4] Note that in the original definition [9], an adversary is defined as a collection of *faulty sets*, i.e., the sets of processes that can fail in an execution. For convenience, we chose here an equivalent definition based on *correct sets*.

3.3 Failure patterns and environments

An adversary is in fact a special case of a *failure environment* introduced by Chandra et alii [6]. An environment $\mathcal{E}$ is a set of *failure patterns*. For a given run, a failure pattern F is a map that associates each time value $t \in \mathbb{T}$ with a set of processes crashed by time t. The set of correct processes, denoted $correct(F)$ is thus defined as $\Pi - \cup_{t\in\mathbb{T}} F(t)$.

Since an adversary $\mathcal{A}$ only defines sets of correct processes and does not specify the timing of failures, it can be viewed as a specific environment $\mathcal{E}_\mathcal{A}$ that is closed under changing the timing of failures. More precisely, $\mathcal{E}_\mathcal{A} = \{F \mid correct(F) \in \mathcal{A}\}$. Clearly, if $F \in \mathcal{E}_\mathcal{A}$ and $correct(F) = correct(F')$, then $F' \in \mathcal{E}_\mathcal{A}$.

Thus, we can rephrase the statement "task T can be solved $\mathcal{A}$-resiliently" as "task T can be solved in environment $\mathcal{E}_\mathcal{A}$". It is shown in [15] that, with respect to colorless tasks, all environments can be split into n equivalence classes, and each class j agrees on the set of tasks it can solve: namely, tasks that can be solved $(j-1)$-resiliently and not j-resiliently. Therefore, by applying [15], we conclude that each adversary belongs to one of such equivalence class. However, this characterization does not give us an explicit algorithm to compute the class to which a given adversary belongs.

3.4 Asymmetric progress conditions

Imbs et alii [24] introduced *asymmetric progress conditions* that allow us to specify different progress guarantees for different processes. Informally, for sets of processes X and Y, $X \subseteq Y \subseteq \Pi$, (X, Y)-liveness guarantees that every process in X makes progress regardless of other processes (wait-freedom for processes in X) and every process in $Y - X$ makes progress if it is eventually the only process in $Y - X$ taking steps (obstruction-freedom for processes in $Y - X$).

With respect to solving colorless tasks, it is easy to represent (X, Y)-liveness using the formalism of adversaries. The equivalent adversary $\mathcal{A}_{X,Y}$ consists of all subsets of Π that intersect with X and all sets $\{p_i\} \cup S$ such that $p_i \in Y - X$ and $S \subseteq \Pi - Y$. It is easy to see that a colorless task is (read-write) solvable assuming (X, Y)-liveness if and only if it is solvable in the presence of $\mathcal{A}_{X,Y}$.

Taubenfeld [31] introduced a refined condition that associates each process p_i with a set $\mathcal{P}_i$ of process subsets (each containing p_i). Then p_i is expected to make progress (e.g., output a value in a task solution) only if the current set of correct processes is in $\mathcal{P}_i$. Similarly, with respect to the question of solvability of colorless tasks, every such progress condition can be modeled as an adversary, defined simply as the union $\cup_i \mathcal{P}_i$.

4 Characterizing superset-closed adversaries

Intuitively, the size of a smallest-cardinality core of an adversary $\mathcal{A}$, denoted $csize(\mathcal{A})$, is related to its ability to "confuse" the processes (preventing them from agreement). Indeed, since in every execution, at least one process in a minimal core C is correct, we can treat C as a collection of leaders. But for a superset-closed adversary, every non-empty subset of C can be *the* set of correct processes in C in some execution. Therefore, intuitively, the system behaves like a wait-free system on $c = |C|$ processes, where c quantifies the "degree of disagreement" that we can observe among all the processes in the system.

In this section, we show that $csize(\mathcal{A})$ precisely captures the power of $\mathcal{A}$ with respect to colorless tasks. We overview two approaches to address this question, each interesting in its own right: using combinatorial topology and using shared-memory simulations.

4.1 A topological approach

Herlihy and Rajsbaum [20] derived a characterization of superset-closed adversaries using the Nerve Theorem of modern combinatorial topology [3]. A set of finite executions is modeled as a *simplicial complex*, a geometric (or combinatorial) structure where each simplex models a set of local states (*views*) of the processes resulting after some execution. This allows for reasoning about the power of a model using topological properties (e.g., connectivity) of simplicial complexes it generates.[5]

The model of [20] is based on *iterated* computations: each process p_i proceeds in (asynchronous) rounds, where every round r is associated with a shared array of registers $M[r,1],\ldots,M[r,n]$. When p_i reaches round r, it updates $M[r,i]$ with its current view and takes an atomic snapshot of $M[r,.]$. In the presence of a superset-closed adversary $\mathcal{A}$, the set of processes appearing in a snapshot should be an element of $\mathcal{A}$. We call the resulting set of executions the *$\mathcal{A}$-compliant iterated model*.

Naturally, given an adversary $\mathcal{A}$, it is easy to implement an iterated model with desired properties in the classical (non-iterated) shared memory model. To implement a round of the iterated model, every process writes its value in the memory and takes atomic snapshots until all processes in some survivor set (minimal element in $\mathcal{A}$) are observed to have written their values. The result of this snapshot is then returned. In an $\mathcal{A}$-compliant execution, this allows for simulating infinitely many iterated rounds.

[5] For more information on the applications of algebraic and combinatorial topology in distributed computing, check Maurice Herlihy's lectures at Technion [19].

Surprisingly, we can also use the $\mathcal{A}$-compliant iterated model to simulate an $\mathcal{A}$-compliant execution in the read-write model where *some* participating set of processes in $\mathcal{A}$ takes infinitely many steps (please check the wonderful simulation algorithm proposed recently by Gafni and Rajsbaum [17]). In particular, for the wait-free adversary $\mathcal{A}_{WF}$, the simulation is *non-blocking*: at least one participating process accepts infinitely many steps in the simulated execution.

Note that if the simulated $\mathcal{A}$-compliant execution is used for an $\mathcal{A}$-resilient protocol solving a given task, then we are guaranteed that at least one process obtains an output. But to solve a colorless task it is sufficient to produce an output for one participating process (all other participants may adopt this output). Thus:

Theorem 1. *[17] hosted Let $\mathcal{A}$ be a superset-closed adversary. A colorless task can be solved in the $\mathcal{A}$-compliant iterated model if and only if it can be solved in the $\mathcal{A}$-compliant model.*

This result allows us to apply the topological formalism as follows. The set of r-round executions of the $\mathcal{A}$-compliant iterated model applied to an initial simplex σ generates a *protocol complex* $\mathcal{K}_r(\sigma)$. By a careful reduction to the Nerve Theorem [3], $\mathcal{K}_r(\sigma)$ can be shown to be $(c-2)$-*connected*, i.e., $\mathcal{K}_r(\sigma)$ contains no "holes" in dimensions $c-2$ or less (any $(c-2)$-dimensional sphere can be continuously contracted to a point). The Nerve theorem establishes the connectivity of a complex from the connectivity of its components.

Roughly, the argument of [20] is built by induction on n, the number of processes. For a given adversary $\mathcal{A}$ on n processes with the minimal core size c, the $\mathcal{A}$-compliant protocol complex $\mathcal{K}_r(\sigma)$ can be represented as a union of protocol complexes, each corresponding to a sub-adversary of $\mathcal{A}$ on $n-1$ processes with core size $c-1$. By induction, each of these sub-adversaries is at least $(c-3)$-connected. Applying the Nerve theorem, we derive that $\mathcal{K}_r(\sigma)$ is $(c-2)$-connected. The base case $n=1$ and $c=1$ is trivial, since every non-empty complex is, by definition, (-1)-connected.

Thus, $\mathcal{K}_r(\sigma)$ is $(c-2)$-connected. Hence, no task that cannot be solved $(c-1)$-resiliently, in particular $(c-1)$-set consensus, allows for an $\mathcal{A}$-resilient solution [22].

Using the characterization of [22], we can reduce the question of $\mathcal{A}$-resilient solvability of a colorless task $T=(\mathcal{I},\mathcal{O},\Delta)$ to the existence of a continuous map f from $|skel^{c-1}(\mathcal{I})|$, the Euclidean embedding of the $(c-1)$-*skeleton* (the complex of all simplexes of dimension $c-1$ and less) of the input complex $\mathcal{I}$, to $|\mathcal{O}|$, the Euclidean embedding of the output complex $\mathcal{O}$, such that f is *carried by* Δ, i.e., $f(\sigma)\subseteq\Delta(\sigma)$. Indeed, the fact that of $\mathcal{K}_r(\sigma)$ is $(c-2)$-connected (and thus d-connected for all $0\le d\le c-2$) implies that every continuous map from d-sphere of $\mathcal{K}_r(\sigma)$ extends to the $(d+1)$-disk, for $0\le d\le c-2$. Intuitively, we can thus

inductively construct a continuous map from $|skel^{c-1}(\mathcal{I})|$ to $|\mathcal{O}|$, starting from any map sending a vertex of $\mathcal{I}$ to a vertex of $\mathcal{O}$ (for $d = 0$).

On the other hand, it is straightforward to construct an $\mathcal{A}$-resilient protocol solving a colorless task T, given a continuous map from the $(c-1)$-skeleton of the input complex of T to the output complex of T. Thus:

Theorem 2. *[20] An adversary $\mathcal{A} \in \mathcal{SC}$ with the minimal core size c allows for solving a colorless task $T = (\mathcal{I}, \mathcal{O}, \Delta)$ if and only if there is a continuous map from $|skel^{c-1}(\mathcal{I})|$ to $|\mathcal{O}|$ carried by Δ.*

Therefore, two adversaries in $\mathcal{A}, \mathcal{B} \in \mathcal{SC}$ with the same minimal core size c agree on the set of tasks they allow for solving, which is exactly the set of tasks that can be solved $(c-1)$-resiliently (since $csize(\mathcal{A}_{(c-1)\text{-}res}) = c$).

4.2 A simulation-based approach

It is comparatively straightforward to characterize superset-closed adversaries using classical BG-simulation [4, 5], and we present a complete proof below.

Theorem 3. *[14] Let $\mathcal{A}$ be a superset-closed adversary. A colorless task T is $\mathcal{A}$-resiliently solvable if and only if T is $(c-1)$-resiliently solvable, where c is the minimal core size of $\mathcal{A}$.*

Proof. Let a colorless task T be $(c-1)$-resiliently solvable, and let P_c be the corresponding algorithm. Let $C = \{q_1, \ldots, q_c\}$ be a minimal-cardinality core of $\mathcal{A}$ ($|C| = c$).

Let the processes in C BG-simulate the algorithm P_c running on all processes in Π. Here each simulator q_i tries to use its input value of task T as an input value of every simulated process [4, 5]. Since C is a core of $\mathcal{A}$, in every $\mathcal{A}$-compliant execution, at most $c-1$ simulators may fail. Since a faulty simulator results in at most one faulty simulated process, the produced simulated execution is $(c-1)$-resilient. Since P_c gives a $(c-1)$-resilient solution of T, at least one simulated process must eventually decide in the simulated execution. The output value is then adopted by every correct process. Moreover, the decided value is based on the "real" inputs of some processes. Since T is colorless, the decided values are correct with respect to the input values and, thus, we obtain an $\mathcal{A}$-resilient protocol to solve T.

For the other direction, suppose, by contradiction that there exists an $\mathcal{A}$-resilient protocol $P_{\mathcal{A}}$ to solve a colorless task T, but T is not possible to solve $(c-1)$-resiliently.

We claim that $\mathcal{A}_{(c-1)\text{-}res} \subseteq \mathcal{A}$, i.e., each $(c-1)$-resilient execution is $\mathcal{A}$-compliant. Suppose otherwise, i.e., some set S of $n-c+1$ processes is not in

$\mathcal{A}$. Since $\mathcal{A}$ is superset-closed, no subset of S is in $\mathcal{A}$ (otherwise, S would be in $\mathcal{A}$). No process in S belongs to any set in $\mathcal{A}$, thus, the smallest core of $\mathcal{A}$ must be a subset of $\Pi - S$. But $|\Pi - S| = c - 1$—a contradiction with the assumption that the size of a minimal cardinality core of $\mathcal{A}$ is c.

Thus, every $(c-1)$-resilient execution is also $\mathcal{A}$-compliant, which implies that $P_{\mathcal{A}}$ is in fact a $(c-1)$-resilient solution to T—a contradiction with the assumption that T is not $(c-1)$-resiliently solvable. □

Theorem 3 implies that adversaries in $\mathcal{SC}$ can be categorized into n equivalence classes, $\mathcal{SC}_1, \ldots, \mathcal{SC}_n$, where class $\mathcal{SC}_k$ corresponds to cores of size k. Two adversaries that belong to the same class $\mathcal{SC}_k$ agree on the set of colorless tasks they are able to solve, and it is exactly the set of all colorless task that can be solved $(k-1)$-resiliently.

5 Measuring the Power of Generic Adversaries

Let us come back to the "bimodal" adversary $\mathcal{A}_{BM} = \{pqr, p, q, r\}$ (Figure 1). Its only core is $\{p, q, r\}$. Does it mean that $\mathcal{A}_{BM}$ only allows for solving trivial (wait-free solvable) tasks? Not really: by splitting $\mathcal{A}_{BM}$ in two sub-adversaries $\mathcal{A}_{FF} = \{pqr\}$ and $\mathcal{A}_{OF} = \{p, q, r\}$ and running two consensus algorithms in parallel, one assuming no failures ($\mathcal{A}_{FF}$) and one assuming that exactly one process is correct ($\mathcal{A}_{OF}$), gives us a solution to 2-set consensus.

5.1 Solving consensus with $\mathcal{A}_{BM}$

But can we solve more in the presence of $\mathcal{A}_{BM}$? E.g., is there a protocol *Alg* that solves consensus $\mathcal{A}_{BM}$-resiliently? We derive that the answer is no by showing how processes, s_0 and s_1, can wait-free solve consensus through simulating an $\mathcal{A}_{BM}$-compliant execution of *Alg*. Initially, the two processes act as BG simulators [4, 5] trying to simulate an execution of *Alg* on *all* three processes p, q, and r. When a simulator s_i $(i = 0, 1)$ finds out that the simulation of some step is blocked (which means that the other simulator s_{1-i} started but has not yet completed the corresponding instance of BG-agreement), s_i switches to simulating a *solo execution* of the next process (in the round-robin order) in $\{p, q, r\}$. If the blocked simulation eventually resolves (s_{1-i} finally completes the instance of BG-agreement), then s_i switches back to simulating all p, q and r.

If no simulator blocks a simulated step forever, the simulated execution contains infinitely many steps of every process, i.e., the set of correct processes in it is $\{p, q, r\}$. Otherwise, eventually some simulated process forever runs in isolation and the set of correct processes in the simulated execution is $\{p\}$, $\{q\}$, or $\{r\}$. In

both cases, the simulated execution of *Alg* is $\mathcal{A}_{BM}$-compliant, and the algorithm must output a value, contradicting [11, 28]. This argument can be easily extended to show that $\mathcal{A}_{BM}$ cannot allow for solving any colorless task that cannot be solved 1-resiliently.

5.2 Disagreement power of an adversary

Thus, we need a more sophisticated criterion to evaluate the power of a generic adversary $\mathcal{A}$. Delporte et alii [9] proposed to evaluate the "disorienting strength" of an adversary $\mathcal{A}$ via its *disagreement power*.

Definition 1. *[9] The disagreement power of an adversary $\mathcal{A}$ is the largest k such that k-set consensus cannot be solved in the presence of $\mathcal{A}$.*

It is shown in [9] that adversaries of the same disagreement power agree on the sets of colorless task they allow for solving. The result is derived via a three-stage simulation. First, it is shown how an adversary can simulate any *dominating* adversary, where the domination is defined through an involved recursive inclusion property. Second, it is shown that every adversary $\mathcal{A}$ that does not dominate the k-resilient adversary[6] is strong enough to implement the anti-Ω_k failure detector that, in turn, can be used to solve k-set consensus [34]. Finally, it is shown that vector-Ω_k (a failure detector equivalent to anti-Ω_k) can be used to solve any colorless task that can be solved k-resiliently. Thus, the largest k such that k-set consensus cannot be solved $\mathcal{A}$-resiliently indeed captures the power of $\mathcal{A}$.

The characterization of adversaries proposed in [9] does not give a direct way of computing the disagreement power of an adversary $\mathcal{A}$ and it does not provide a direct $\mathcal{A}$-resilient algorithm to solve a colorless task T, when T is $\mathcal{A}$-resiliently solvable.

In the rest of this section, we give a simple algorithm to compute the disagreement power of an adversary. For convenience, we introduce notion of *set consensus power*, i.e., the smallest k such that k-set consensus can be solved in the presence of $\mathcal{A}$. Clearly, the disagreement power of $\mathcal{A}$ is the set consensus power of $\mathcal{A}$ minus 1.

5.3 Defining *setcon*

Let $\mathcal{A}$ be an adversary and let $S \subseteq P$ be any subset of processes. Then $\mathcal{A}_S$ denotes the adversary that consists of all elements of $\mathcal{A}$ that are subsets of S (including S itself if $S \in \mathcal{A}$). E.g., for $\mathcal{A} = \{pq, qr, q, r\}$ and $S = qr$, $\mathcal{A}_S = \{qr, q, r\}$. For

[6] Recall that the k-resilient adversary consists of all subsets of Π of size at least $n - k$.

$S \in \mathcal{A}$ and $a \in S$, let $A_{S,a}$ denote the adversary that consists of all elements of $\mathcal{A}_S$ that *do not* include a. E.g., for $\mathcal{A} = \{pq, qr, q, r\}$, $S = qr$, and $a = q$, $\mathcal{A}_{S,a} = \{r\}$.

Now we define a quantity denoted *setcon*($\mathcal{A}$), which we will show to be the set consensus power of $\mathcal{A}$. Intuitively, our goal is to split $\mathcal{A}$ into the minimal number k of sub-adversaries, such that every sub-adversary allows for solving consensus. Then $\mathcal{A}$ allows for solving k-set consensus, but not $(k-1)$-set consensus (otherwise, k would not be minimal).

Definition 2. *setcon*($\mathcal{A}$) *is defined as follows:*

- *If* $\mathcal{A} = \emptyset$, *then* $setcon(\mathcal{A}) = 0$
- *Otherwise,* $setcon(\mathcal{A}) = \max_{S \in \mathcal{A}} \min_{a \in S} setcon(\mathcal{A}_{S,a}) + 1$

Thus, *setcon*($\mathcal{A}$), for a non-empty adversary $\mathcal{A}$, is determined as $setcon(\mathcal{A}_{\overline{S},\overline{a}})+1$ where $\overline{S}$ is an element of $\mathcal{A}$ and $\overline{a}$ is a process in $\overline{S}$ that "max-minimize" $setcon(\mathcal{A}_{S,a})$. Note that for $\mathcal{A} \neq \emptyset$, $setcon(\mathcal{A}) \geq 1$.

We say that $S \in \mathcal{A}$ is *proper* if it is not a subset of any other element in $\mathcal{A}$. Let *proper*($\mathcal{A}$) denote the set of proper elements in $\mathcal{A}$. Note that since for all $S' \subset S$, $\min_{a \in S'} setcon(\mathcal{A}_{S',a}) \leq \min_{a \in S} setcon(\mathcal{A}_{S,a})$, we can replace $S \in \mathcal{A}$ with $S \in proper(\mathcal{A})$ in Definition 2.

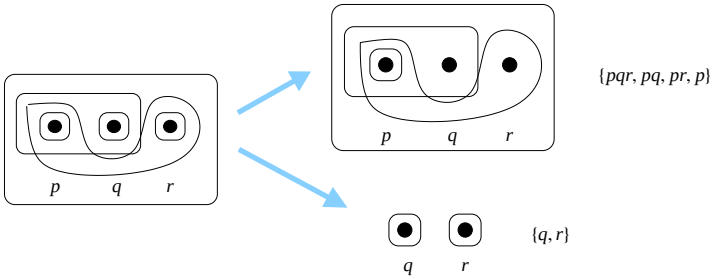


Figure 2: Adversary $\mathcal{A} = \{pqr, pq, pr, p, q, r\}$ decomposed in two sub-adversaries, $\{pqr, pq, pr, p\}$ and $\{q, r\}$, each with *setcon* = 1.

5.4 Calculating *setcon*($\mathcal{A}$): examples

Consider an adversary $\mathcal{A} = \{pqr, pq, pr, p, q, r\}$. It is easy to see that $setcon(\mathcal{A}) = 2$: for $S = pqr$ and $a = p$, we have $\mathcal{A}_{S,p} = \{q, r\}$ and $setcon(\mathcal{A}_{S,a}) = 1$. Thus, we decompose $\mathcal{A}$ into two sub-adversaries $\{pqr, pq, pr, p\}$ and $\{q, r\}$, each strong enough to solve consensus (Figure 2). Intuitively, in an execution where the correct set belongs to $\mathcal{A} - \mathcal{A}_{S,a} = \{pqr, pq, pr, p\}$, process p can act as a leader

for solving consensus. If the correct set belongs to $\mathcal{A}_{S,a} = \{q, r\}$ (either q or r eventually runs solo) then q and r can solve consensus using an obstruction-free algorithm. Running the two algorithms in parallel, we obtain a solution to 2-set consensus. The reader can easily verify that any other choice of $a \in pqr$ results in three levels of decomposition.

As another example, consider the t-resilient adversary $\mathcal{A}_{t\text{-}res} = \{S \subseteq \Pi, |S| \geq n - t\}$. It is easy to verify recursively that $setcon(\mathcal{A}_{t\text{-}res}) = t + 1$: at each level $1 \leq j \leq t + 1$ of recursion we consider a set S of $n - j + 1$ elements, pick up a process $p \in S$ and delegate the set of $n - j$ processes that do not include p to level $j+1$. At level $t+1$ we get one set of size $n-t$ and stop. Thus, $setcon(\mathcal{A}_{t\text{-}res}) = t+1$.

More generally, for any superset-closed adversary $\mathcal{A}$ ($\mathcal{A} \in \mathcal{SC}$), $setcon(\mathcal{A}) = csize(\mathcal{A})$, the size of a smallest-cardinality core of $\mathcal{A}$. To show this, we proceed by induction. The statement is trivially true for an empty adversary $\mathcal{A}$ with $csize(\mathcal{A}) = setcon(\mathcal{A}) = 0$. Now suppose that for all $0 \leq j < k$ and all $\mathcal{A}' \in \mathcal{SC}$ with $csize(\mathcal{A}') = j$, we have $setcon(\mathcal{A}') = j$. Consider $\mathcal{A} \in \mathcal{SC}$ such that $csize(\mathcal{A}) = k$. Note that the only proper element of $\mathcal{A}$ is the whole set of processes Π. Thus, $setcon(\mathcal{A}) = \min_{a \in \Pi} setcon(\mathcal{A}_{\Pi,a}) + 1$. By the induction hypothesis and the fact that $csize(\mathcal{A}) = k$, we have $\min_{a \in \Pi} setcon(\mathcal{A}_{\Pi,a}) = k - 1$. Thus, $setcon(\mathcal{A}) = k$.

Thus, by Theorem 3, *setcon*() indeed characterizes the disorienting power of adversaries $\mathcal{A} \in \mathcal{SC}$: a task is $\mathcal{A}$-resiliently solvable if and only if it is $(c - 1)$-resiliently solvable, where $c = setcon(\mathcal{A})$. In the rest of this section, we extend this result from $\mathcal{SC}$ to the universe of all adversaries.

5.5 Solving consensus with $setcon = 1$

Before we characterize the ability of adversaries to solve colorless tasks, we consider the special case of adversaries of $setcon = 1$.

Consider an adversary $\mathcal{A}$ and $S \in \mathcal{A}$. Suppose $csize(\mathcal{A}_S) = 1$, and let $\{a\}$ be a core of $\mathcal{A}_S$. Obviously, $\mathcal{A}_{S,a} = \emptyset$. On the other hand, if $\mathcal{A}_{S,a} = \emptyset$, then $\{a\}$ is a core of $\mathcal{A}_S$. Thus, $setcon(\mathcal{A}) = 1$ if and only if $\forall S \in \mathcal{A}$, $csize(\mathcal{A}_S) = 1$

Suppose $setcon(\mathcal{A}) = 1$. If S is the only proper element of $\mathcal{A}$, then we can easily solve consensus (and, thus, any other task [18]), by deciding on the value proposed by the only member of a core of $\mathcal{A}_S$. The process is guaranteed to be correct in every execution.

Now we extend this observation to the case when $\mathcal{A}$ contains multiple proper elements. The consensus algorithm, presented in Figure 3, is a "rotating coordinator" algorithm inspired by by Chandra and Toueg [7].

The algorithm proceeds in rounds. In each round r, every process p_i first tries to commit its current decision estimate in a new instance of commit-adopt CA_r. If p_i succeeds in committing the estimate, the committed value is written in the

Shared variables:
  D, initially $\perp$
  $R_1, \ldots, R_n$, initially $\perp$

propose(v)
1 $est := v$
2 $r := 0$
3 $S := P$
4 **repeat**
5 $r := r + 1$
6 $(flag, est) := CA_r.propose(est)$
7 **if** $flag = commit$ **then**
8 $D := est$; $return(est)$ {Return the committed value}
9 $R_i := (est, r)$
10 **wait until** $\exists S \in \mathcal{A}, \forall p_j \in S : R_j = (v_j, r_j)$ where $r_j \geq r$ **or** $D \neq \perp$
 {Wait until a set in $\mathcal{A}$ moves}
11 **if** $p_{r \mod n+1} \in S$ **then**
12 $est := v_{r \mod n+1}$ {Adopt the estimate of the current leader}
13 **until** $D \neq \perp$
14 $return(D)$

Figure 3: Consensus with a "one-level" adversary $\mathcal{A}$, $setcon(\mathcal{A}) = 1$

"decision" register D and returned. Otherwise, p_i adopts the returned value as its current estimate and writes it in R_i equipped with the current round number r. Then p_i takes snapshots of $\{R_1, \ldots, R_n\}$ until either a set $S \in \mathcal{A}$ reaches round r or a decision value is written in D (in which case the process returns the value found in D). If no decision is taken yet, then p_i checks if the coordinator of this round, $p_{r \mod n}$, is in S. If so, p_i adopts the value written in $R_{r \mod n}$ and proceeds to the next round.

The properties of commit-adopt imply that no two processes return different values. Indeed, the first round in which some process commits on some value v (line 8) "locks" the value for all subsequent rounds, and no other process can return a value different from v.

Suppose, by contradiction, that some correct process never returns in some $\mathcal{A}$-compliant execution e. Recall that $\mathcal{A}$-compliant means that some set in $\mathcal{A}$ is exactly the set of correct processes in e. If a process returns, then it has previously written the returned value in D. Since, in each round, a process performs a bounded number of steps, by our assumption, no process ever writes a value in D and every correct process goes through infinitely many rounds in e without returning.

Let $\bar{S} \in \mathcal{A}$ be the set of correct processes in e. After a round r' when all processes outside $\bar{S}$ have failed, every element of $\mathcal{A}$ evaluated by a correct process in line 10 is a subset of $\bar{S}$. Finally, since the minimal core size of $\mathcal{A}_{\bar{S}}$ is 1, all these elements of $\mathcal{A}$ overlap on some correct process p_j.

Consider round $r = mn + j \geq r' - 1$. In this round, p_j not only belongs to all sets evaluated by the correct processes, but it is also the coordinator ($j = r \mod n+1$). Thus, the only value that a process can propose to commit-adopt in round $r + 1$ is the value previously written by p_j in R_j. Hence, every process that returns from commit-adopt in round $r + 1$ must commit and return—a contradiction. Thus:

Theorem 4. *[14] If setcon$(\mathcal{A}) = 1$, then consensus can be solved $\mathcal{A}$-resiliently.*

5.6 Adversarial partitions

One way to interpret Definition 2 is to say that *setcon*$(\mathcal{A})$ captures the size of a minimal-cardinality partitioning of $\mathcal{A}$ into sub-adversaries $\mathcal{A}^1, \ldots, \mathcal{A}^k$, each of *setcon* $= 1$.

Indeed, for a proper set $S \in \mathcal{A}$, selecting an element $a \in S$ allows for splitting $\mathcal{A}_S$ into two sub-adversaries $\mathcal{A}_S - \mathcal{A}_{S,a}$ and $\mathcal{A}_{S,a}$. $\mathcal{A}_S - \mathcal{A}_{S,a}$ is the set of elements of $\mathcal{A}_S$ that contain a and, thus, *setcon*$(\mathcal{A}_S - \mathcal{A}_{S,a}) = 1$ (a can act as a leader). Moreover, selecting a so that *setcon*$(\mathcal{A}_{S,a})$ is minimized makes sure that $\mathcal{A}_{S,a} =$ *setcon*$(\mathcal{A}_S) - 1$.

Intuitively, $\mathcal{A}^1$, the first such sub-adversary, is the union of $\mathcal{A}_S - \mathcal{A}_{S,a}$, for all such proper $S \in \mathcal{A}$ and $a \in S$. Adversaries $\mathcal{A}_2, \ldots, \mathcal{A}_k$ are obtained by a recursive partitioning of all $\mathcal{A} - \mathcal{A}^1$. (A detailed description of this partitioning can be found in [14].)

Thus, given an adversary $\mathcal{A}$ such that $setcon(\mathcal{A}) = k$, we derive that $\mathcal{A}$ allows for solving k-set consensus. Just take the described above partitioning of $\mathcal{A}$ in to k sub-adversaries, $\mathcal{A}^1, \ldots, \mathcal{A}^k$ such that, for all $j = 1, \ldots, k$, $setcon(\mathcal{A}^j) = 1$. Then every process can run k parallel consensus algorithms, one for each $\mathcal{A}^j$, proposing its input value in each of these consensus instances (such algorithm exist by Theorem 4). Since the set of correct processes in every $\mathcal{A}$-compliant execution belongs to some $\mathcal{A}^j$, at least one consensus instance returns. The process decides on the first such returned value. Moreover, at most k different values are decided and each returned value was previously proposed. Thus:

Theorem 5. *[14] If $setcon(\mathcal{A}) = k$, then $\mathcal{A}$ allows for solving k-set consensus.*

5.7 Characterizing colorless tasks

But can we solve $(k-1)$-set consensus in the presence of $\mathcal{A}$ such that $setcon(\mathcal{A}) = k$? As shown in [14], the answer is no: $\mathcal{A}$ does not allow for solving any colorless task that cannot be solved $(k-1)$-resiliently. The result is derived by a simple application of BG simulation [4, 5].

The intuition here is the following. Suppose, by contradiction, that we are given an adversary $\mathcal{A}$ such that $setcon(\mathcal{A}) = k$ and a colorless task T that is solvable $\mathcal{A}$-resiliently but not $(k-1)$-resiliently. Let *Alg* be the corresponding $\mathcal{A}$-resilient algorithm. Then we can construct a $(k-1)$-resilient simulation of an $\mathcal{A}$-compliant execution of *Alg*. Roughly, we build upon BG-simulation, except that the *order* in which steps of *Alg* are simulated is not fixed in advance to be round-robin. Instead, the order is determined online, based on the currently observed set of participating processes.

We start with simulating steps of processes in $S \in \mathcal{A}$ such that $setcon(\mathcal{A}_S) = k$ (by Definition 2, such S exists). If the outcome of a simulated step of some process a cannot be resolved (the corresponding BG-agreement is blocked), we proceed to simulating processes in an element $S' \in \mathcal{A}_{S,a}$ with the largest *setcon* (if there is any). As soon as the blocked BG-agreement on the step of a resolves, the simulation returns to simulating S. Since $setcon(\mathcal{A}) = k$, we can obtain exactly k levels of simulation. Therefore, in a $(k-1)$-resilient execution, at most $k-1$ simulated processes (each in a distinct sub-adversary of $\mathcal{A}$) can be blocked forever. Since $\mathcal{A}$ allows for k such sub-adversaries, at least one set in $\mathcal{A}$ accepts infinitely many simulated steps. The resulting execution is thus $\mathcal{A}$-compliant, and

we obtain a $(k-1)$-resilient solution for T—a contradiction (detailed argument is given in [14]).

In fact, the set of colorless tasks that can be solved given an adversary $\mathcal{A}$ such that $setcon(\mathcal{A}) = k$ is *exactly* the set of colorless tasks that can be solved $(k-1)$-resiliently, but not k-resiliently. Indeed, $\mathcal{A}$ allows for solving k-set consensus, and we can employ the generic algorithm of [13] that solves any $(k-1)$-resilient colorless task using the k-set consensus algorithm as a black box. Thus:

Theorem 6. *[14] Let $\mathcal{A}$ be an adversary such that $setcon(\mathcal{A}) = k$ and T be a colorless task. Then $\mathcal{A}$ solves T if and only if T is $(k-1)$-resiliently solvable.*

Recall that the set consensus power of an adversary $\mathcal{A}$ is the smallest k such that $\mathcal{A}$ can solve k-set consensus. Theorem 6 implies:

Corollary 7. *The set consensus power of $\mathcal{A}$ is $setcon(\mathcal{A})$, and the disagreement power of $\mathcal{A}$ is $setcon(\mathcal{A}) - 1$.*

By Theorem 3, determining $setcon(\mathcal{A})$ may boil down to determining the minimum hitting set size of $\mathcal{A}$, and thus, by [27]:

Corollary 8. *Determining the set consensus power of an adversary is NP-complete.*

6 Concluding remarks

This survey primarily talks about colorless tasks (consensus, set agreement, simplex agreement, et cetera) in the read-write shared memory systems where processes may fail by crashing in a non-uniform (non-identical and correlated) way. We modeled such non-uniform failures using the language of adversaries [9] and we derived a complete characterization of an adversary via its set consensus power [14] (or, equivalently its disagreement power [9]).

The techniques discussed here can be extended to models where processes may also communicate through stronger objects than just read-write registers (e.g., k-process consensus objects). In particular, BG-simulation is used in [14] to capture the ability of leveled adversaries of [31] to prevent processes from solving consensus among n processes using k-process consensus objects ($k < n$).

Combinatorial topology proved to be a powerful instrument in analyzing a special class of superset-closed adversaries and colorless tasks, not only in read-write shared-memory models [20], but also in a variety of other models, including message-passing models and iterated models with k-set consensus objects.

However, the power of adversaries with respect to generic (not necessarily) colorless tasks is still poorly understood. Consider, for example, a task $\mathcal{T}_{pq}$ which requires processes p and q (in a system of three processes p, q, and r) to solve

consensus and allows r to output any value. The task is obviously not colorless: the output of r cannot always be adopted by p or q. The 2-obstruction-free adversary $\mathcal{A}_{2\text{-}OF} = \{pq, pr, qr, p, q, r\}$ does not allow for solving T_{pq}: otherwise, we would get a wait-free 2-process consensus algorithm. On the other hand, $\mathcal{A}_{pq} = \{pqr, pq, p, r\}$ (p is correct whenever q is correct) allows for solving T_{pq} (just use p as a leader for p and q). But $setcon(\mathcal{A}_{2\text{-}OF}) = setcon(\mathcal{A}_{pq}) = 2$!

One may say that the task T_{pq} is "asymmetric": it prioritizes outputs of some processes with respect to the others. Maybe our result would extend to symmetric tasks whose specifications are invariant under a permutation of process identifiers? Unfortunately, there are symmetric colored tasks that exhibit similar properties [33]. So we need a more fine-grained criterion than set consensus power to capture the power of adversaries with respect to colored tasks.

Finally, this paper focuses on non-uniform *crash* faults in asynchronous shared-memory systems. Non-uniform patterns of generic (Byzantine) types of faults are explored in the context of Byzantine quorum systems [29] (see also a survey in [32]) and secure multi-party computations [23]. Both approaches assume that a faulty process can deviate from its expected behavior in an arbitrary (Byzantine) manner. In particular, in [29], Malkhi and Reiter address the issues of non-uniform failures in the Byzantine environment by introducing the notion of a *fail-prone system* (*adversarial structure* in [23]): a set $\mathcal{B}$ of process subsets such that no element of $\mathcal{B}$ is contained in another, and in every execution some $B \in \mathcal{B}$ contains all faulty processes. Determining the set of tasks solvable in the presence of a given generic adversarial structure is an interesting open problem.

References

[1] Yehuda Afek, Hagit Attiya, Danny Dolev, Eli Gafni, Michael Merritt, and Nir Shavit. Atomic snapshots of shared memory. *Journal of the ACM*, 40(4):873–890, 1993.

[2] Michael Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols (extended abstract). In *PODC '83: Proceedings of the annual ACM symposium on Principles of distributed computing*, pages 27–30, 1983.

[3] A. Björner. *Topological methods*, pages 1819–1872. MIT Press, Cambridge, MA, USA, 1995.

[4] Elizabeth Borowsky and Eli Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *STOC*, pages 91–100. ACM Press, May 1993.

[5] Elizabeth Borowsky, Eli Gafni, Nancy A. Lynch, and Sergio Rajsbaum. The BG distributed simulation algorithm. *Distributed Computing*, 14(3):127–146, 2001.

[6] Tushar Deepak Chandra, Vassos Hadzilacos, and Sam Toueg. The weakest failure detector for solving consensus. *Journal of the ACM*, 43(4):685–722, July 1996.

[7] Tushar Deepak Chandra and Sam Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[8] Soma Chaudhuri. More *choices* allow more *faults*: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, 1993.

[9] Carole Delporte-Gallet, Hugues Fauconnier, Rachid Guerraoui, and Andreas Tielmann. The disagreement power of an adversary. *Distributed Computing*, 24(3-4):137–147, 2011.

[10] Faith Ellen Fich, Victor Luchangco, Mark Moir, and Nir Shavit. Obstruction-free algorithms can be practically wait-free. In *Proceedings of the International Symposium on Distributed Computing*, pages 493–494, 2005.

[11] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.

[12] Eli Gafni. Round-by-round fault detectors (extended abstract): Unifying synchrony and asynchrony. In *Proceedings of the 17th Symposium on Principles of Distributed Computing*, 1998.

[13] Eli Gafni and Rachid Guerraoui. Generalized universality. In *Proceedings of the 22nd international conference on Concurrency theory*, CONCUR'11, pages 17–27, Berlin, Heidelberg, 2011. Springer-Verlag.

[14] Eli Gafni and Petr Kuznetsov. Turning adversaries into friends: Simplified, made constructive, and extended. In *OPODIS*, pages 380–394, 2010.

[15] Eli Gafni and Petr Kuznetsov. On set consensus numbers. *Distributed Computing*, 24(3-4):149–163, 2011.

[16] Eli Gafni and Petr Kuznetsov. Relating L-Resilience and Wait-Freedom via Hitting Sets. In *ICDCN*, pages 191–202, 2011.

[17] Eli Gafni and Sergio Rajsbaum. Distributed programming with tasks. In *OPODIS*, pages 205–218, 2010.

[18] Maurice Herlihy. Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):123–149, January 1991.

[19] Maurice Herlihy. Advanced topics in distributed algorithms. Technion Lecture, 2011. http://video.technion.ac.il/Courses/Adv_Topics_in_Dist_Algorithms.html.

[20] Maurice Herlihy and Sergio Rajsbaum. The topology of shared-memory adversaries. In *PODC*, pages 105–113, 2010.

[21] Maurice Herlihy and Nir Shavit. The asynchronous computability theorem for t-resilient tasks. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 111–120, May 1993.

[22] Maurice Herlihy and Nir Shavit. The topological structure of asynchronous computability. *Journal of the ACM*, 46(2):858–923, 1999.

[23] Martin Hirt and Ueli M. Maurer. Complete characterization of adversaries tolerable in secure multi-party computation (extended abstract). In *PODC*, pages 25–34, 1997.

[24] Damien Imbs, Michel Raynal, and Gadi Taubenfeld. On asymmetric progress conditions. In *PODC*, 2010.

[25] Flavio Junqueira and Keith Marzullo. A framework for the design of dependent-failure algorithms. *Concurrency and Computation: Practice and Experience*, 19(17):2255–2269, 2007.

[26] Flavio Paiva Junqueira and Keith Marzullo. Designing algorithms for dependent process failures. In *Future Directions in Distributed Computing*, pages 24–28, 2003.

[27] Richard M. Karp. Reducibility among combinatorial problems. *Complexity of Computer Computations*, pages 85–103, 1972.

[28] M.C. Loui and H.H. Abu-Amara. Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, 4:163–183, 1987.

[29] Dahlia Malkhi and Michael K. Reiter. Byzantine quorum systems. *Distributed Computing*, 11(4):203–213, 1998.

[30] Michael Saks and Fotios Zaharoglou. Wait-free *k*-set agreement is impossible: The topology of public knowledge. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 101–110. ACM Press, May 1993.

[31] Gadi Taubenfeld. The computational structure of progress conditions. In *DISC*, 2010.

[32] Marko Vucolić. The origin of quorum systems. *Bulletin of EATCS*, 101:125–147, June 2010.

[33] Piotr Zieliński. Sub-consensus hierarchy is false (for symmetric, participation-aware tasks). https://sites.google.com/site/piotrzielinski/home/symmetric.pdf.

[34] Piotr Zieliński. Anti-*omega*: the weakest failure detector for set agreement. *Distributed Computing*, 22(5-6):335–348, 2010.

The Logic in Computer Science Column

by

Yuri Gurevich

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

Type Inference in Mathematics

Jeremy Avigad*

Abstract

In the theory of programming languages, *type inference* is the process of inferring the type of an expression automatically, often making use of information from the context in which the expression appears. Such mechanisms turn out to be extremely useful in the practice of interactive theorem proving, whereby users interact with a computational proof assistant to construct formal axiomatic derivations of mathematical theorems. This article explains some of the mechanisms for type inference used by the *Mathematical Components* project, which is working towards a verification of the Feit-Thompson theorem.

*This work has been partially supported by NSF grants DMS-0700174 and DMS-1068829. During the 2009–2010 academic year I spent a sabbatical year working on the verification of the Feit-Thompson theorem with the Mathematical Components group at INRIA-Microsoft Joint Research Centre in Orsay, and I am grateful to Georges Gonthier and the center for support. I am also grateful to Yuri Gurevich, Assia Mahboubi, Enrico Tassi, and an anonymous referee for many helpful comments, corrections, and suggestions.

1 Introduction

Consider the following mathematical assertions:

- For every x in $\mathbb{R}$, $e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!}$.
- If G and H are groups, f is a homomorphism from G to H, and a and b are in G, then $f(ab) = f(a)f(b)$.
- If F is a field of nonzero characteristic p, and a and b are in F, then

$$(a+b)^p = \sum_{i=0}^{p} \binom{p}{i} a^i b^{p-i} = a^p + b^p.$$

There is nothing unusual about these statements, but, on reflection, one notices that substantial background knowledge and assumptions are needed to parse them correctly. For example, in the first statement, we take it that the index of the summation i ranges over natural numbers, or, equivalently, nonnegative integers. Hence $i!$ is also an integer. Since x is explicitly flagged as a real number, the expression $x^i/i!$ involves division of two different types of objects, taking into account that any integer can be viewed as a real number. In the second statement, G and H are groups, which is to say, each is a set of elements equipped with a group operation and an identity element; so when we write that a and b are in G, we really mean that a and b are elements of the underlying set. The notation ab denotes multiplication in G, while the notation $f(a)f(b)$ can only be understood in terms of the multiplication in H. In the third statement, p is a nonnegative integer (in fact, a prime number, since nonzero field characteristics are prime). But unlike the summation symbol in the first statement, here the summation symbol refers to addition in F. In the third statement, $\binom{p}{i}$ is an integer, while a^i and b^{p-i} are elements of the field. How do we interpret multiplication in *that* case? One way is to notice that there is a canonical map from the integers to any ring with a 0 and a 1. Alternatively, any abelian group can be viewed as a $\mathbb{Z}$-module, which means that it supports scalar multiplication by integers, with all the expected properties; and the additive part of a ring is an abelian group.

Inferences like these are used not only to parse basic mathematical expressions, but also to reason about them correctly. For example, some "multiplications" and "additions" are commutative, and multiplication often distributes over the corresponding addition. Common manipulations with summations depend on such facts. Understanding mathematics presupposes the ability to keep track of the various domains that objects belong to and variables range over, as well as the relevant operations on those domains and their properties. Our faculties for doing

this are so ingrained that we are scarcely aware of the background knowledge we bring to the table when we read an ordinary mathematical text.

The problem is that such background knowledge has to be brought to the foreground when it comes to formalizing mathematics. Broadly speaking, *formal verification* is the practice of using formal methods to verify correctness, such as verifying that a circuit description, an algorithm, or a network or security protocol meets its specification. In this article, I will be concerned, instead, with the verification of mathematical theorems. To be sure, there is no sharp distinction between verifying mathematical statements and verifying claims about hardware and software systems, since the latter are typically expressed in mathematical terms. But ordinary mathematical theorems have a special character, and raise distinct issues and challenges.

Specifically, I will focus on *interactive theorem proving,* which involves working interactively with a proof assistant to provide enough information for the system to confirm that the theorem in question has, indeed, a formal proof. In fact, many systems actually construct a formal proof object, a complex piece of data that can be verified independently. Systems with substantial mathematical libraries include Coq [5] (including the Ssreflect extension [21]), HOL [24], HOL light [28], Isabelle [37], and Mizar [25]. In September 2004, assisted by some students at Carnegie Mellon, I verified a proof of the Hadamard/de la Vallée Poussin prime number theorem [3], using the Isabelle proof assistant. Since then number of nontrival theorems have been formalized.

, including the four-color theorem [18], the prime number theorem [3, 30], the Jordan curve theorem [26, 33], Gödel's first incompleteness theorem [42, 38], Dirichlet's theorem on primes in an arithmetic progression [29], the Cartan fixed-point theorems [9], and various theorems of measure theory [31, 35]. There are, moreover, some interesting large-scale verification projects underway. Thomas Hales is heading the *Flyspeck* project [27], which aims to verify a number of results in discrete geometry, including the Kepler conjecture. Georges Gonthier is heading the *Mathematical Components* project [17, 19], which aims to verify the Feit-Thompson theorem. Fields medalist Vladimir Voevodsky has launched a project to develop "univalent foundations" for algebraic topology, providing the basis for formal verification in a theorem prover like Coq.

Checking the details of a mathematical proof is by no means the most interesting or important part of mathematics, and formal verification is not meant to serve as a substitute for mathematical creativity and understanding. But it is generally recognized that the mathematical literature is filled with misstatements, gaps, ambiguities, overlooked cases, omitted hypotheses, and so on, and that the lack of reliability is problematic [36]. Moreover, an increasing number of proofs today rely on extensive calculation, and there are currently no standards to ensure that mathematical software is sound. Mathematicians always strive for correctness,

and formal verification is simply a technology that is designed to support that goal.

Despite the achievements to date, however, formal verification is still not "ready for prime time." There is a steep learning curve to working with an interactive theorem prover, and verifying even straightforward mathematical results can be frustrating and time consuming. We need better libraries, automated methods, and infrastructure to support verification efforts. This is an exciting time for a young and rapidly evolving field.

In this article, I will focus on one small aspect of formal verification, namely, type inference. In the mathematical setting, the challenge of type inference, roughly speaking, is to keep track of the kinds of objects that appear in a mathematical statement and put that information to good use. What is common to the previous examples is that in each case the relevant information can be inferred from context:

- In the expression "a is in G," the object of the word "in" is expected to be a set.
- In "ab," multiplication takes place in the group that a and b are assumed to be an element of.
- In "$x^i/i!$," one expects the arguments to be elements of a common structure, for which a division operation is defined.

Type inference thus involves not only inferring type information, but also inferring data and facts from type considerations. Of course, type inference is central to the theory of programming languages [39], and many of the ideas and methods that have been developed there have been transferred to the mathematical setting. But, as will become clear, mathematical type inference has a distinct flavor. Here I will focus primarily on the approach to type inference used in the Mathematical Components project, which relies on a proof language, Ssreflect, and the Coq theorem prover.

In Section 2, I will consider what is desired from a mathematical perspective. In Section 3, I will discuss some of the underlying axiomatic frameworks, and dependent type theory in particular. In Section 4, I will describe some of the mechanisms in Coq that are designed to meet the challenges posed in Section 2. In Section 5, I will describe the way some of these mechanisms are used in the Mathematical Components library, and in Section 6, I will briefly indicate some alternative approaches.

2 Mathematical type inference

One hallmark of modern mathematics is the tendency to identify mathematical objects as elements of algebraically characterized structures. Such structures, and classes of such structures, can be related in various ways:

- Structures in one class may be viewed as elements of a broader one. For example, every abelian group is, more generally, a group, and every group is, more generally, a monoid. Sometimes the inclusions are obtained by taking reducts, which is to say, ignoring parts of the structure. For example, the additive part of a ring is an abelian group, while the multiplicative part is a monoid.

- A particular structure or a structure in one class can often be embedded in a larger structure. For example, the integers can be embedded in the reals, and every integral domain can be embedded in its field of fractions.

- Uniform constructions can be used to build elements of one class of structures from elements of another. For example, the units in any ring form a group, under the associated multiplication; the set of automorphisms of a field (or those fixing some chosen subfield) form a group under composition; any metric space gives rise to a topological space determined by the metric; the field of fractions of any integral domain is a field; and the quotient of a group by a normal subgroup is again a group.

What makes this perspective useful is that it allows one to transfer insights and results gained from one domain to another, and apply background knowledge and expertise uniformly in different settings. The challenge for interactive proof assistants is to reap these benefits.

There are various ways that algebraic methods promote efficiency:

- They allow us to reuse notation. For example, one may wish to use the symbols 0 and + with respect to the integers, the reals, and arbitrary rings.

- They allow us to reuse constructions. For example, summation $\sum_{i \in I} a_i$ in the integers, reals, and an arbitrary ring can be viewed as instances of the same construction, namely an iteration of the corresponding addition. In fact, various "big" operations, including multiplication, logical operations of conjunction and disjunction, lattice operations of meet and join, and so on can be viewed as iterations of an associative operation in an arbitrary monoid.

- They allow us to reuse facts. Various identities involving big operations can be viewed as instances of general laws that can be instantiated in the different settings. For example, some identities involving summations presuppose that the addition is commutative. Other identities hold in the presence of a multiplication that distributes over addition. We implicitly recognize that such facts hold at various degrees of generality, and instantiate them as appropriate.

Any proof assistant that is designed to formalize contemporary mathematical arguments should support these types of reuse.

In the theory of programming languages, type inference allows users to omit information that can be inferred from context. For example, if we write $f(i)$ and i is known to range over the integers, we can infer that f is a function from the integers to some other domain. Various kinds of "polymorphism" allow one to reuse symbols and code across different domains. In the context of formally verified mathematics, there are really two types of information that can be inferred:

- data: for example, the appropriate multiplication in an expression $a \cdot b$, or the appropriate summation operation in an expression $\sum_{i \in A} f(i)$.

- facts: for example, the fact that $(a \cdot b) \cdot c$ is equal to $a \cdot (b \cdot c)$, when the multiplication in the relevant structure is associative.

In the next section, we will see that in certain formulations of logic, these two can be understood as instances of a common phenomena. In other words, inferring a fact can be viewed as inferring a special kind of data, namely "evidence" or "the fact" that the associated claim is true.

To summarize, in interactive theorem proving, type inference may be invoked when the system parses an expression, but also when the user applies a lemma, or searches for a lemma to apply. The goal of type inference is to allow the user to omit information systematically when such information can be inferred from context. Not only does this save time and energy and reduce tedium, but it also ensures that the expressions we type look like the mathematics we are familiar with, lending support to the claim that our formalizations adequately "capture" informal mathematical practice.

3 Dependent type theory

In order to verify mathematical proofs in a given domain, one has to first choose a formal axiomatic framework that is flexible enough to model arguments in that domain. Experience from the last century has shown that the Zermelo-Fraenkel

axioms of set theory provides a remarkably robust foundation for mathematics. Indeed, the Mizar system [25], which has perhaps the most extensive mathematical library, is based on an extension of ZF known as Tarski-Groethendieck set theory.

But, in set theory, every object is a set, meaning that the axiomatic framework does not distinguish between numbers, functions, structures, and other objects. For the purposes of type inference, it is often useful to have such distinctions built into the underlying formal system. A number of proof assistants today, including HOL [24], HOL light [28], and Isabelle [37], are based on formulations of higher-order logic like Church's simple type theory [8]. One starts with basic types, such as a type `nat` of natural numbers and a type `bool` of boolean truth values, and adds constructors for forming new types. The most important of these are function types: whenever `A` and `B` are types, so is $\texttt{A} \to \texttt{B}$, intended to denote the type of functions from `A` to `B`. One can also allow, for example, product types $\texttt{A} \times \texttt{B}$, denoting the type of ordered pairs from `A` and `B`. Most proof systems have additional mechanisms to support the definition of common mathematical data types and structures, and allow "polymorphic" variables ranging over types.

The problem with simple type theory, however, is that it is too simple, since ordinary mathematical structures often depend on parameters. For example, for each n, $\mathbb{R}^n$ is a vector space, and for every $n \geq 1$, the integers modulo n form a ring. Thus one may wish to have types

- `list A n`, denoting sequences of objects of type A, with length n; and
- `Zmod n`, denoting the ring of integers modulo n.

In *dependent type theory*, types can depend on parameters in this way. Notice that such a move tends to blur the distinction between types and terms. For example, in `list A n`, the first argument is supposed to denote a type, whereas the second argument is supposed to be a term of type `nat`. In some presentations of type theory, this is achieved by having special types, called *universes*, whose terms are also construed as types (see, for example, the presentation of Martin-Löf type theory in [47, Section 7.1]). Contemporary presentations more often take types to be inhabitants of a third level of syntactic objects, known as "sorts" or "kinds" (see [4]). The specific details need not concern us here; only the fact that terms as well as types can depend on parameters that are again terms or types.

In dependent type theory, the type $\texttt{A} \to \texttt{B}$ of functions which take an argument in `A` and return a value in `B` can be generalized to a dependent product $\prod_{\texttt{x:A}} \texttt{B(x)}$, where `B(x)` is a type that can depend on `x`. Intuitively, elements of this type are functions that map an element `a` of `A` to an element of `B(a)`. When `B` does not depend on `x`, the result is just $\texttt{A} \to \texttt{B}$. Similarly, product types $\texttt{A} \times \texttt{B}$ can be generalized to dependent sums $\sum_{\texttt{x:A}} \texttt{B(x)}$. Intuitively, elements of this type are

pairs `(a, b)`, where `a` is an element of `A` and `b` is an element of `B(a)`. When `B` does not depend on `x`, this is just $\mathtt{A} \times \mathtt{B}$.

In the next section, we will consider one particular theorem prover, Coq. Coq's underlying logic is a dependent type theory known as the *calculus of inductive constructions*, or *CIC* [12], which extends the original *calculus of constructions* due to Coquand and Huet [11]. The calculus of inductive constructions has four distinguishing features:

- It is a powerful and expressive dependent type theory.
- It incorporates the "propositions as types" correspondence.
- It is constructive, in that every expression in the system has a computational interpretation.
- The computational interpretation of terms is used in type checking.
- Type checking is decidable.

These features are not to everyone's taste, and we will see in Section 6 that other proof assistants can reject any or all of them. I will elaborate on each, in turn.

One striking feature of the Calculus of Inductive Constructions is that there are only two basic type-forming operations: dependent products and inductive types. We have already discussed dependent products. Inductive types allow one to define structures that can be characterized as the closure of a set under some basic operations, like the natural numbers, or lists and trees over a type. But, in the CIC, the construction is general enough to include dependent sums, as well as to interpret basic logical notions, like conjunction, disjunction, universal and existential quantification, and equality. In fact, the system has the logical strength of strong systems of set theory [49].

In order to interpret logical operations in terms of type-theoretic constructions, the CIC relies on what has come to be known as the Curry-Howard "propositions as types" correspondence. The point is that logical operations look a lot like operations on datatypes. For example, in propositional logic, from `A` and `B` one can conclude $\mathtt{A} \wedge \mathtt{B}$. One can read this as saying that given a proof `a` of `A` and a proof `b` of `B` of B one can "pair" them to obtain a proof `(a, b)` of $\mathtt{A} \wedge \mathtt{B}$; or given the "fact" `a` that `A` holds, and the fact `b` that `B` holds, one obtains the fact `(a, b)` that $\mathtt{A} \wedge \mathtt{B}$ holds. Moreover, from the fact that $\mathtt{A} \wedge \mathtt{B}$ holds, one can extract the fact that `A` holds, and, similarly, `B`. If you replace $\mathtt{A} \wedge \mathtt{B}$ by $\mathtt{A} \times \mathtt{B}$, this is nothing more than a characterization of the product type. In other words, if we posit a new collection `Prop` of types and take the product constructor to map elements `A : Prop` and `B : Prop` to $\mathtt{A} \times \mathtt{B}$ `: Prop`, the rules governing products for elements of `Prop` are exactly the desired logical rules for conjunction.

Under this correspondence, implications A → B are just instances of function types, and bounded universal quantifiers ∀x : A. B(x) are just instances of the dependent product construction. In other words, a proof of ∀x : A. B(x) can be viewed as a procedure which, given any object a : A returns a proof of B(a). This explains Coq's notation forall x : A, B x for dependent products. Similarly, the logical construction ∃x : A. B(x) is just an instance of the dependent sum. Using inductively defined types, given any type A one can form I_A(x,y) : Prop which, intuitively, denotes the proposition that x is equal to y as elements of A.

One can take the propositions-as-types as expressing a deep insight into the nature and meaning of logical operations [34, 48]. But one can just as well view it as a notational convenience which, moreover, allows a proof assistant to treat logical and mathematical operations uniformly. For example, one can take the transitivity of inequality on the natural numbers, leq_trans, to be a term of type

$$\forall x{:}\mathtt{nat},\ y{:}\mathtt{nat},\ z{:}\mathtt{nat},\ x \le y \to y \le z \to x \le z.$$

This last expression, in turn, it a term of type Prop. One can view leq_trans not just as the fact that less-than-or-equal is transitive, but also as a function which, given elements x, y, and z in the natural numbers as well as the facts that x ≤ y and y ≤ z, return the fact that x ≤ z. Thus, given a : nat, b : nat, and c : nat, the term leq_trans a b c denotes the implication a ≤ b → b ≤ c → a ≤ c. Moreover, we can express that H is the fact that a ≤ b by writing H : a ≤ b, in which case leq_trans a b c H denotes the implication b ≤ c → a ≤ c.

The propositions-as-types correspondence is particularly popular as a foundation for constructive mathematics, where assertions are expected to have direct computational significance. Every term in Coq can be viewed as a computational object, subject to evaluation. For example, if π_0 and π_1 denote the two projections from a product type A × B, the a term π_0(a, b) can be "reduced" or "evaluated" to a. In fact, every term in Coq can, at least in principle, be reduced to a canonical normal form. In particular, if t is a closed term of type nat, then t reduces to a numeral. Coq, moreover, makes use of this computational interpretation when checking types. For example, If C(x) is a type that depends on a value x of type A, the system can recognize that C(π_0(a, b)) is the same type as C(a).

The decidability of type checking amounts to the fact that given a term, t, and a type, T, the type-checker can, deterministically, decide whether or not t has type T. This is clearly a useful property to have, though we will see, in Section 6, that it imposes strong restrictions. Under the propositions-as-types correspondence, the decidability of type checking takes on additional significance. Suppose P is a term of type Prop, expressing, for example, Fermat's last theorem. Then a term t of type P is a proof that P is true. Proving Fermat's last theorem thus amounts to

constructing a term of type P, and the decidability of type checking implies that such a term can be recognized, algorithmically, as a valid proof.

4 Type inference in Coq

Now that we have a sense of Coq's axiomatic framework, let us explore some of the mechanisms the system offers to address the challenges raised in Section 2. Generally speaking, type inference is triggered when the system is called on to determine the type of a term, or to check that a term has an appropriate type, when some information has been left implicit. But because dependent types depend on the values of their parameters, inferring a type can entail inferring such values. Recall that in Section 2 we distinguished between two types of information that can be inferred, namely, data and facts. With the propositions-as-types correspondence in place, inferring a fact—such as the fact that multiplication is associative—is a matter of inferring a value of a type P, which is in turn of type Prop, where P expresses the expected associativity property.

We will consider three principal mechanisms. *Implicit arguments* allow users to systematically leave information out of an expression when this information can be inferred from context. *Coercions* allow users to cast, implicitly, objects of one type to objects of another. Finally, *canonical structures* let the user register certain objects as components of a larger structure, providing useful information to the type inference process.[1]

It will be helpful to illustrate these with a running example. The following definition declares a new type, group:

```
Record group : Type := Group
{
  carrier : Type;
  mulg : carrier -> carrier -> carrier;
  oneg : carrier;
  invg : carrier -> carrier;
  mulgA : forall x y z : carrier,
    mulg x (mulg y z) = mulg (mulg x y) z;
  ...
}.
```

[1]For more detail than is provided below, see Coq's online reference manual. All three mechanisms were initially introduced to Coq by Amokrane Saïbi [32, 40, 41], who credits the idea of using implicit arguments in the theorem proving context to Peter Aczel. Implicit arguments were further extended by Hugo Herbelin and Matthieu Sozeau. Canonical structures received little attention until they were revived and used aggressively by Gonthier; see, for example, [17].

According to this type declaration, `group` is a record type, consisting of a carrier, a multiplication, an identity, and an inverse. These are assumed to satisfy the requisite axioms, such as the associativity of multiplication. If `G` has type `group`, that is, `G : group`, then the components of `G` are `carrier G`, `mulg G`, `oneg G`, and so on. Conversely, given elements `my_carrier`, `my_mul`, `my_one` and so on of the right type, the term `Group my_carrier my_mul my_one ...` denotes the corresponding group.

Notice that we are relying on dependent type theory here. The type `group` is a classic example of a dependent sum, since, for example, the type of the second component, `carrier -> carrier -> carrier`, depends on the value `carrier` of the first component. The arguments of the corresponding projections bear the associated dependences. For example, the term `mulg`, which picks out the the second component, has type `forall G : group, carrier G -> carrier G -> carrier G`, a dependent product.

Notice also that the proposition-as-types correspondence is being put to good use. For example, the type of the fifth component, `mulgA`, is the proposition that `mulg` is associative. Assuming `G : group`, the term `mulgA G` has type

```
forall x y z : carrier G,
  mulg G x (mulg G y z) = mulg G (mulg G x y) z
```

which is itself a term of type `Prop`. Thus `mulgA G` denotes the fact that multiplication in `mulg G` is associative, a fact that can be applied to elements of the carrier of `G` just as in the example of `leq_trans` above. In this way, the propositions-as-types correspondence provides a natural and convenient way to think of the group structure as including not only the relevant data—the carrier of the group and group operations—but also the relevant properties.

In a context where we have `G : group`, `g : carrier G`, and `h : carrier G`, the term `mulg G g h` represents the product of `g` and `h` under the multiplication operation of `G`. The implicit arguments mechanism in Coq allows us to write `mulg _ g h`, replacing the first argument by an underscore. Doing so means that we expect the type inference algorithm to infer the value of that argument from context, by finding a solution to the constraints imposed by the fact that the resulting term should be well typed. The algorithm proceeds by instantiating the first element with a variable, `?`. The term `mulg ?` then has type `carrier ? -> carrier ? -> carrier ?`. Since this term is applied to `g : carrier G`, to get the types to work out the system has to solve a simple unification problem, namely, instantiating `?` to unify `carrier ?` with `carrier G`. Thus `?` is instantiated to `G`, and the algorithm has inferred the relevant parameter. With this in mind, one can introduce a new notation:

```
Notation "g * h" := (mulg _ g h).
```

This enables one to write `g * h` for multiplication in any group, allowing the group in question to be inferred from the type of `g`.

In this example, the implicit argument mechanism was used to infer a parameter in the application of a function, `mulg`. But the mechanism can be used just as well to infer parameters during the application of a lemma. For example, recall the transitivity lemma `leq_trans` from the last section. This takes five arguments—three natural numbers, `x`, `y`, `z`, and the facts $x \leq y$ and $y \leq z$—and returns the fact $x \leq z$. Suppose we declare the first three arguments to be implicit. Then given `H1 : a ≤ b` and `H2 : b ≤ c`, the term `leq_trans H1 H2` has type $a \leq c$. Moreover, when we are building a proof interactively in Coq, if we apply `leq_trans H1` to a subgoal $a \leq c$, type inference similarly infers the missing arguments and leaves the us with the goal $b \leq c$.

Coercions are commonly used in programming languages, for example, when adding a real and an integer triggers the coercion of the integer to a real. In the context of mathematical theorem proving, coercions have other uses as well. In our running example, one would ordinarily write `g : carrier G` to specify that `g` is an element of the carrier of `G`. Writing `g : G` instead yields an error, because the system expects something of type `Type` on the right side of the colon, and `G` has type `group`. But declaring

```
Coercion carrier : group >-> Type.
```

informs Coq that the function `carrier` can always be used to coerce a group to a type. If one then enters `g : G`, the algorithm finds itself facing a group on the right side of the colon but expecting a type, and readily inserts the coercion.

The last feature that we will discuss, canonical structures, provides an inverse to coercion, of sorts. In the example above, we used the `carrier` function to coerce a record structure to one of its projections. Canonical structures makes it possible for the type inference algorithm to pass in the other direction, and recognize a particular object as the projection of a larger structure. To illustrate, suppose we define

```
IntGroup := Group int addi zeroi negi addiA ...
```

thereby declaring the integers with addition to be an instance of a group. Somewhat perversely, this will allow us to write `mulg IntGroup i j` instead of `i + j`, when we have `i j : int`. Less perversely, this will allow us to apply general theorems about groups to this particular instance. But what happens now when we write `i * j`? This expression is shorthand for `mulg _ i j`. After instantiating the first argument to a variable, `?`, the type inference algorithm is faced with the unification problem `carrier ? = int`, and gets stuck. Declaring

```
Canonical Structure IntGroup.
```

registers the fact `carrier IntGroup = int` with the system for use in type inference. One can view this as a "hint" to the unification process [2]. Now when the type inference algorithm gets stuck as above, it can appeal to a table of such hints, and use the relevant one to recognize that the integers can be viewed as the carrier of the `IntGroup` structure. The algorithm then replaces `int` by `carrier IntGroup` and solves the unification problem.

The mechanisms just described are not exceedingly complicated, but we will see in the next section that they are remarkably robust with respect to the challenges posed in Section 2. Canonical structures can, moreover, be used in clever ways to trick the type inference algorithm into carrying out various kinds of useful automation [23].

To summarize, type checking is triggered when the user enters an expression or applies a lemma, possibly leaving some arguments and facts implicit. Coq's type inference engine has four resources at its disposal to fill in the remaining information:

1. *unification* can be used to infer implicit arguments;

2. *coercions* can be inserted to resolve a type mismatch;

3. the unification algorithm can refer to a database of *unification hints* to solve unification problems involving a projection of a *canonical structure*; and

4. when all else fails, the algorithm can simplify terms or unfold definitions according to the CIC's computational interpretation of terms, and then retry the previous steps.

Generally speaking, implicit arguments can trigger arbitrary instances of higher-order unification, which is known to be undecidable [14]. So, at best, type inference can only aim to search a reasonable fragment of the space of possible instantiations for an implicit argument. And even within decidable fragments, unpacking definitions and unfolding terms can easily lead to combinatorial explosion. Nonetheless, Coq's type inference algorithm consists, essentially, of iterating the steps above, relying on heuristics to limit the possibilities in the fourth step.

5 The mathematical components library

This section provides a brief indication of some of the ways that the mechanisms for type inference discussed in Section 4 have been used towards Gonthier's formalization of the Feit-Thompson theorem [15], which asserts that finite groups of

odd order are solvable. These examples only scratch the surface; for more detail, see [6, 17, 18, 21, 22].

Recall that Coq's logic is constructive. In contrast, many principles and methods that are commonly used in contemporary mathematics are not constructively valid. For example, constructively, one cannot assume the law of the excluded middle, or prove the existence of an x satisfying a property P by assuming there is no such x and deriving a contradiction. Extensionality fails: one cannot, in general, prove that two functions f and g from A to B are equal by proving that $f(x) = g(x)$ for every x. Choice fails as well: even if one has proved that for every x in A there is a y in B such that some property holds, one cannot assume that there is a function f that picks out such a y for every x.

On the other hand, these properties generally hold in *finite* domains. Since the Feit-Thompson theorem is an extended exploration of properties of finite groups, one would like to take advantage of these features when they are available. Thus, in the Ssreflect library, there are general structures for types with a decidable equality relation (that is, ones where the relation can be computed by a function returning a boolean value of "true" or "false," ensuring that it satisfies the law of the excluded middle); finite structures; and structures that can be equipped with choice functions. For example, one can define a structure for types with decidable equality as follows:

```
Record eqType : Type := EqType
{
  carrier : Type;
  rel : carrier -> carrier -> bool;
  ax : forall x y, (x = y) = (rel x y)
}.
```

In the last line of the record, the expression `rel x y` of type `bool` is coerced to the proposition that the value of this expression is equal to `true`. In other words, `ax` is the proposition that `rel x y` holds if and only if `x = y`. Declaring `carrier` to be a coercion allows one to write `x : T` whenever we have `T : eqType`. Implicit arguments allow one to use the notation `x == y` in place of `rel T x y` whenever `x` and `y` are elements of the carrier of such a `T`. Finally, canonical structures allow one to associate the relevant boolean equality relation with the natural numbers, so that one can write `x == y` when we have `x y : nat`, as well. (This is a slight simplification of the implementation in the Ssreflect library [17].)

Section 2 noted that "big operations" such as $\sum$, $\prod$, $\bigcap$, $\bigcup$, $\bigwedge$, $\bigvee$ can all be viewed as instances of iterations of an associative binary operation. But such operations come in many different flavors: one can sum over a list, a numeric range, or a finite set, and these summations will satisfy different properties depending on whether the underlying structure is a semigroup, an abelian semigroup, or a

ring. Ssreflect comes with an overarching "bigop" library, and once again type inference plays a key role in making it work [6].

Type inference is also used to manage algebraic class inclusions (between rings, commutative rings, fields, and son on) and algebraic constructions: for example, the set of n by n matrices over a ring forms a ring when $n > 0$, and the set of polynomials over a commutative ring again forms a commutative ring. Type inference ensures that the relevant algebraic facts are readily available, and allows a uniform use of notation [17, 20]. Definitions in the Ssreflect library have been carefully chosen so that if `G` and `H` are groups of the same type (more precisely, subgroups of some ambient group type), then the quotient notation `G / H` makes sense; but when `H` is in fact a normal subgroup of `G`, as in the usual construction of a quotient group, `G / H` is a group with all the expected properties [22]. For another example, when a group `G` happens to be abelian, it is often treated as a $\mathbb{Z}$-module and written additively. So, for example, one can write `g *+ n` for scalar multiplication of `g` by `n` whenver `g` is an element of the group and `n` is a natural number. Type inference is used to mediate between these two "views" of an abelian group.

Type inference also helps with mundane mathematical conventions. For example, Section 2 noted the conflation of groups with sets. If `G` and `H` are subgroups of an ambient finite group, and A is a subset of that group, then `G ∩ H` and `C_G(A)` (the centralizer of `A` in `G`) are both groups. But they are also just sets with the ambient group operation; an element `x` is in `G ∩ H` if and only if it is in `G` and `H`, and `x` is in `C_G(A)` if and only if `x` is in `G` and commutes with every element of `A`. Type inference mediates between these two views of a construction—that is, of yielding both a group and a set—allowing one to apply lemmas involving groups in some instances and lemmas involving sets in others. For another example, a homomorphism between groups G and H is a function between G and H equipped (using a record type) with additional properties. Coercion allows one to use ordinary function notation with morphisms, such as `f x` and `f ∘ g`. In the other direction, canonical structures automatically infer the fact that `f ∘ g` is a homomorphism when `f` and `g` are, giving `f ∘ g` a similarly dual status as function and morphism.

Canonical structures can even be used to make sense of mildly abusive mathematical notation. For example, if U and W are subspaces of a vector space V, it is common to write $U + W$ for set $\{u + w \mid u \in U, w \in W\}$. Mathematicians will often say "$U + W$ is a direct sum" when U and W have trivial intersection, ignoring the fact that this is a property of the pair (U, W) which is impossible to read off from the $U + W$ alone. Gonthier has shown, however, that canonical structures provide a convenient way of supporting this abuse of language [20].

6 Limitations and other approaches

The mechanisms supporting type inference that were described in Section 4 are not the only ones available in Coq. In particular, Coq now has a "type class" mechanism [44]. Type classes and canonical structures serve similar purposes, but whereas canonical structures are handled within the type inference loop described at the end of Section 4, the type class mechanism collects constraints that are passed to a separate inference engine at the end of the process. Spitters and van der Weegen [45] have experimented with type classes in the context of mathematical type inference, with positive results.

But one may wish to stray even further from Coq's mindset. Recall some of the key features of that proof assistant:

- An elaborate type theory is built in to the underlying axiomatic framework.
- Using the propositions-as-types correspondence, data and facts are handled in the same way, so theorems can be applied to arguments and hypotheses just as functions are applied to arguments.
- The underlying logic is constructive, and every term has computational significance.
- Type checking makes use of the computational interpretation of terms.
- Type checking is decidable.

These are very strong constraints, which interact with each other in subtle ways and place strong restrictions on the way mathematics is represented and carried out. Not every proof assistant adopts such a framework. In fact, most reject the third, allowing classical reasoning that is ubiquitous in contemporary mathematics. Similarly, the propositions-as-types correspondence is usually linked to constructive theories, though there is no reason that it cannot be adopted in classical frameworks as well.

Although the mechanisms for type inference described in this article scale reasonably well, their use in real mathematical settings can be complex and delicate. Moreover, when an expression fails to typecheck, error messages from the system are often uninformative, and it can be frustrating and difficult to diagnose the problem. There are, moreover, rigid limitations to dependent type theory that stem from the commitment to keep type checking decidable. This is so because type checking algorithms are constrained to focus on syntactic structure, without incorporating background knowledge. For example, if `list A n` denotes the type of vectors of elements of `A` of length `n`, and we have `t : list A (0 + n)`, then,

in Coq, `t` also typechecks as an element of `list A n`. In other words, Coq recognizes these two types as being the same. But entering `t : list A (n + 0)` yields a type error; Coq refuses to recognize that `list A (n + 0)` is the same as `list A n`. What is going on is that addition in Coq is defined by recursion on the first argument, so that the the term `0 + n` reduces to `n` under the computational interpretation. But the fact that `n + 0` is equal to `0` is a *mathematical* fact, and there is no general way to incorporate arbitrary mathematical information in type checking while maintaining decidability.

Still, some have explored ways of making type judgments more flexible while maintaining decidability [1, 7, 46]. An alternative is to give up the decidability of type checking, and accept the fact that some type judgments will require proof from the user. This is the path chosen by NuPrl [10] and PVS [43]. Yet another alternative is to jettison type theory altogether, and move to an axiomatic system like set theory, which offers maximum flexibility while relinquishing all the benefits of types; and then try to recapture some of those benefits by adding an extra layer of automation to register and manage domain information outside the axiomatic theory. Such "soft typing" mechanisms can be found, for example, in Mizar [25].

This article has focused on the modeling of mathematical language from the point of view of contemporary interactive theorem provers. Others [13, 16] have come at the problem from the perspective of natural language processing. In the long run, it seems likely that the various approaches will converge.

Inferring domain information is essential to modeling mathematical language and reasoning. Gonthier's work on the Feit-Thompson theorem shows that it is possible to model full-blown algebraic reasoning in an interactive proof systems. But other approaches should also be explored, and continued experimentation and innovation is needed to develop better support for verifying ordinary mathematical proofs.

References

[1] Thorsten Altenkirch, Conor McBride, and Wouter Swierstra. Observational equality, now! In Aaron Stump and Hongwei Xi, editors, *Proceedings of the ACM Workshop Programming Languages meets Program Verification (PLPV) 2007*, pages 57–68. ACM, 2007.

[2] Andrea Asperti, Wilmer Ricciotti, Claudio Sacerdoti Coen, and Enrico Tassi. Hints in unification. In *Theorem Proving in Higher Order Logics (TPHOLs) 2009*, pages 84–98. Springer, Berlin, 2009.

[3] Jeremy Avigad, Kevin Donnelly, David Gray, and Paul Raff. A formally verified proof of the prime number theorem. *ACM Trans. Comput. Logic*, 9(1):2, 2007.

[4] Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

[5] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Springer, Berlin, 2004.

[6] Yves Bertot, Georges Gonthier, Sidi Ould Biha, and Ioana Pasca. Canonical big operators. In *Theorem Proving in Higher Order Logics (TPHOLs) 2008*, pages 86–101. Springer, Berlin, 2008.

[7] Frédéric Blanqui, Jean-Pierre Jouannaud, and Mitsuhiro Okada. The calculus of algebraic constructions. In *10th International Conference on Rewriting Techniques and Applications (RtA) 1999*, pages 301–316. Springer, Berlin, 1999.

[8] Alonzo Church. A formulation of the simple theory of types. *J. Symbolic Logic*, 5:56–68, 1940.

[9] Gianni Ciolli, Graziano Gentili, and Marco Maggesi. A Certified Proof of the Cartan Fixed Point Theorems. *J. Autom. Reasoning*, 47(3):319–336, 2011.

[10] Robert L. Constable et al. *Implementing Mathematics with the Nuprl Proof Development System*. Prentice-Hall, Englewood Cliffs, NJ, 1986.

[11] Thierry Coquand and Gérard Huet. The calculus of constructions. *Inform. and Comput.*, 76(2-3):95–120, 1988.

[12] Thierry Coquand and Christine Paulin. Inductively defined types. In *International Conference on Computer Logic (COLOG) 1988*, pages 50–66. Springer, Berlin, 1990.

[13] Marcos Cramer, Peter Koepke, and Bernhard Schröder. Parsing and disambiguation of symbolic mathematics in the Naproche system. In Davenport, James H. (ed.) et al., eds., Intelligent computer mathematics. Springer, Berlin, 2011.

[14] Gilles Dowek. Higher-order unification and matching. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, volume II, chapter 16, pages 1009–1062. Elsevier Science, Amsterdam, 2001.

[15] Walter Feit and John G. Thompson. Solvability of groups of odd order. *Pacific Journal of Mathematics*, 13:775–1029, 1963.

[16] Mohan Ganesalingam. *The Language of Mathematics*. PhD thesis, University of Cambridge, 2009.

[17] François Garillot, Georges Gonthier, Assia Mahboubi, and Laurence Rideau. Packaging mathematical structures. In *Theorem Proving in Higher Order Logics (TPHOLs) 2009*, pages 327–342. Springer, Berlin, 2009.

[18] Georges Gonthier. Formal proof—the four-color theorem. *Notices Amer. Math. Soc.*, 55(11):1382–1393, 2008.

[19] Georges Gonthier. Advances in the formalization of the odd order theorem. In Marko C. J. D. van Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk, editors, *Interactive Theorem Proving (ITP) 2011*, page 2. Springer, Berlin, 2011.

[20] Georges Gonthier. Point-free, set-free concrete linear algebra. In Marko C. J. D. van Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk, editors, *Interactive Theorem Proving (ITP) 2011*, pages 103–118. Springer, Berlin, 2011.

[21] Georges Gonthier and Assia Mahboubi. An introduction to small scale reflection in Coq. *J. Formaliz. Reason.*, 3(2):95–152, 2010.

[22] Georges Gonthier, Assia Mahboubi, Laurence Rideau, Enrico Tassi, and Laurent Théry. A modular formalisation of finite group theory. In *Theorem Proving in Higher Order Logics (TPHOLs) 2009*, pages 86–101. Springer, Berlin, 2007.

[23] Georges Gonthier, Beta Ziliani, Aleksandar Nanevski, and Derek Dreyer. How to make ad hoc proof automation less ad hoc. In Manuel M. T. Chakravarty, Zhenjiang Hu, and Olivier Danvy, editors, *International Conference on Functional Programming (ICFP) 2011*, pages 163–175. ACM, 2011.

[24] M. J. C. Gordon and T. F. Melham, editors. *Inroduction to HOL: A Theorem Proving Environment for Higher-Order Logic*. Cambridge University Press, 1993.

[25] Adam Grabowski, Artur Korniłowicz, and Adam Naumowicz. Mizar in a nutshell. *J. Formaliz. Reason.*, 3(2):153–245, 2010.

[26] Thomas C. Hales. The Jordan curve theorem, formally and informally. *Amer. Math. Monthly*, 114(10):882–894, 2007.

[27] Thomas C. Hales, John Harrison, Sean McLaughlin, Tobias Nipkow, Steven Obua, and Roland Zumkeller. A revision of the proof of the Kepler conjecture. *Discrete Comput. Geom.*, 44(1):1–34, 2010.

[28] John Harrison. HOL light: a tutorial introduction. In Mandayam Srivas and Albert Camilleri, editors, *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design*, pages 265–269, 1996.

[29] John Harrison. A formalized proof of Dirichlet's theorem on primes in arithmetic progression. *J. Formaliz. Reason.*, 2(1):63–83, 2009.

[30] John Harrison. Formalizing an analytic proof of the prime number theorem. *Journal of Automated Reasoning*, 43:243–261, 2009.

[31] Johannes Hölzl and Armin Heller. Three chapters of measure theory in Isabelle/HOL. In Marko C. J. D. van Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk, editors, *Interactive Theorem Proving (ITP) 2011*, pages 135–151. Springer, Berlin, 2011.

[32] Gérard Huet and Amokrane Saïbi. Constructive category theory. In Gordon Plotkin, Colin P. Stirling, and Mads Tofte, editors, *Proof, language, and interaction: essays in honour of Robin Milner*, pages 235–275. MIT Press, Cambridge, MA, 2000.

[33] Artur Korniłowicz. A proof of the Jordan curve theorem via the Brouwer fixed point theorem. *Mechanized Mathematics and Its Applications*, 6(1):33–40, November 2007.

[34] Per Martin-Löf. An intuitionistic theory of types: predicative part. In H. E. Rose and J. C. Shepherdson eds., *Logic Colloqium '73*, North-Holland, Amsterdam, 1973.

[35] Tarek Mhamdi, Osman Hasan, and Sofiène Tahar. Formalization of entropy measures in hol. In Marko C. J. D. van Eekelen, Herman Geuvers, Julien Schmaltz, and Freek Wiedijk, editors, *Interactive Theorem Proving (ITP) 2011*, pages 233–248. Springer, Berlin, 2011.

[36] Melvyn B. Nathanson. Desperately seeking mathematical proof. *Notices Amer. Math. Soc.*, 55(7):773, 2008.

[37] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL. A Proof Assistant for Higher-Order Logic*. Springer, Berlin, 2002.

[38] Russell O'Connor. Essential incompleteness of arithmetic verified by Coq. In *Theorem Proving in Higher Order Logics (TPHOLs) 2005*, pages 245–260. Springer, Berlin, 2005.

[39] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, Cambridge, MA, 2002.

[40] Amokrane Saïbi. Typing algorithm in type theory with inheritance. In *Symposium on Principles of Programming Languages (POPL) '97*, pages 292–301. ACM, 1997.

[41] Amokrane Saïbi. Outils Génériques de modélisation et de démonstration pour la Formalisation des Mathématiques en théorie des Types, Application à la théorie des catégories. Ph.D. thesis, Universiy of Paris 6, 1999.

[42] Natarjan Shankar. *Metamathematics, machines, and Gödel's proof*. Cambridge University Press, Cambridge, 1994.

[43] Natarajan Shankar and Sam Owre. Principles and pragmatics of subtyping in PVS. In D. Bert, C. Choppy, and P. D. Mosses, editors, *Recent Trends in Algebraic Development Techniques (WADT) 1999*, pages 37–52. Springer, Berlin, 2000.

[44] Matthieu Sozeau and Nicolas Oury. First-class type classes. In *Theorem proving in higher order logics (TPHOLs) 2008*, pages 278–293. Springer, Berlin, 2008.

[45] Bas Spitters and Eelis van der Weegen. Type classes for mathematics in type theory. *Mathematical Structures in Computer Science*, 21(4):795–825, 2011.

[46] Pierre-Yves Strub. Coq modulo theory. In Anuj Dawar and Helmut Veith, editors, *19th EACSL Annual Conference on Computer Science Logic*, pages 549–543. Springer, Berlin, 2010.

[47] A. S. Troelstra and Dirk van Dalen. *Constructivism in Mathematics*, volume 2. North-Holland, Amsterdam, 1988.

[48] William W. Tait. Truth and proof: the Platonism of mathematics. *Synthese*, 69:341–370, 1986. Reproduced in W. D. Hart, editor, *The philosophy of mathematics*, Oxford University Press, Oxford, 1996, pages 142–167.

[49] Benjamin Werner. Sets in types, types in sets. In *Theoretical aspects of computer software*, pages 530–546. Springer, Berlin, 1997.

Reports from Conferences

Report on AFL 2011

13*th* International Colloquium on Automata and Formal Languages 17-22 August 2011, Debrecen, Hungary

Manfred Kudlek

AFL 2011, the 13th conference in this series, founded by István Peák in 1980, took place in Debrecen from August 17-22, 2011. Conference site was Debreceni Akadémiai Bizottság (DAB), the house of the Debrecen Academic Society.

It was organized by Institute of Mathematics and Informatics, College of Nyíregyháza, Institute of Mathematics, University of Szeged, Faculty of Mathematics, Univeristy of Debrecen, and Debrecen Regional Commitee of the Hungarian Academy of Sciences.

The organizing committee consisted of Pál Dömösi (chair), Attila Egri-Nagy, János Falucskai, Szilárd Fazekas, Géza Horváth, Szabolcs Iván, Zita Kovács, Zoltán Mecsei, Benedek Nagy, Zsolt Nagy, György Vaszil, and Viktor Bovdi, as well as Éva Dömösi-Rápolti, Mária Komjáthy, Katalin Puzsár.

AFL 2011 was supported by Institute of Mathematics and Informatics, College of Nyíregyházá, Institute of Informatics, Szeged University, Debrecen Regional Committee of the Hungarian Academy of Sciences, Novadat Company, and Czech-Hungarian Bilateral Research Project.

The conference was attended by 62 participants from 13 countries:

HU	29	JP	5	CA	2	IN	1	UK	1
DE	8	FR	3	AT	1	IT	1		
CZ	6	US	3	FI	1	SK	1		

The scientific program consisted of 5 invited talks, and 25 contributions (22 long, 3 short), selected from 31 submissions (another one was withdrawn) from 13 countries. Apart from the contribution by María Dolores Jiménez López, Gemma Bel-Enguix, Adela Grando, all were presented by one of the authors.

C	I	S	AL	AS	C	I	S	AL	AS	C	I	S	AL	AS
CA		$4\frac{1}{12}$	$3\frac{7}{12}$		FR	2	$\frac{5}{6}$	$\frac{5}{6}$		SK		$1\frac{1}{4}$	$\frac{1}{4}$	
CZ		5	2		HU		$2\frac{5}{6}$	2	$1\frac{5}{12}$	UK		3	$\frac{1}{3}$	$1\frac{7}{12}$
DE	1	5	$5\frac{1}{4}$		IN		$1\frac{1}{3}$	1		US	1	$1\frac{1}{2}$	$1\frac{1}{2}$	
ES		$1\frac{1}{6}$	$1\frac{1}{2}$		JP	1	3	3						
FI		$\frac{1}{2}$	$\frac{1}{2}$		RO		$1\frac{1}{2}$	$\frac{1}{4}$						
Σ											5	31	22	3
W												1		

Bulletin of the EATCS no 106, pp. 101–104, February 2012

The program of AFL 2011 can be found at `http://www.nyf.hu/afl11/`.

The conference was opened on Wednesday morning by György Gát, vice-rector for science and innovation of Nyíregyháza College, talking on István Peák, the history of AFL, 13 as the luckiest prime, the Debrecen flower festival, thanking Pál Dömösi, for the organization, and wishing success and good weather for AFL.

Jean-Éric Pin, with the first invited lecture *'Equational Descriptions of Languages'*, presented an excellent survey on history and motivations of the field, ranging over 45 years, (Schützenberger, McNaughton, Eilenberg, Brzozowski, I. Simon, Thomas, also showing pictures of them). Starting with Birkhoff's, Reitermann's, and Eilenberg's theorems on varieties, he talked on syntactic orders (Bloom, Pin, Weil) and preorders, $\mathcal{C}$-varieties, equational theories of lattices, presented examples, and finished with *dreams* for home work, and *'Whole thing so abstract, probably useless'*.

In the good second one, *'K-Restricted Duplication Closure of Languages'*, Masami Ito presented interesting results on the closure properties of language classes under K-restricted duplication, contextfree being closed whereas regular in general not.

In the third one, Andreas Maletti (co-authors Paweł Gawrychowski, Markus Holzer, Artur Jeż, Daniel Querheim) with *'Notes on Hyper-minimization'* gave a very good overview on the history of a new area of hyper-minimization, on hyper-minimization (kernel states, almost equivalent and merging states), on hyper-optimization, as well as on algorithms and their complextity for them, and their restrictions and limitations.

Christian Choffrut (co-author Serge Grigorieff), in the forth invited talk *'On Relations of Finite Words over Infinite Alphabets'*, presented a good and interesting survey on background and context of the area of first order theories on word relations (Eilenberg, Elgot, Shepherdson, *prefix of, equal length, last letter*, etc.), keeping the notion of finite automata, of first order logic, and showing results of decidable and undecidable theories.

With the fifth invited lecture *'Open Problems on Avoidable Patterns in Partial Words'*, Francine Blanchet-Sadri presented a good and interesting overview on the area, namely on pattern avoidance (Thue-Morse word, avoidability index, K-(un)avoidability, classification of binary patterns, ternary patterns, avoiding Abelian squares and other powers, insertion of arbitrary many holes), and on subword complexity (minimal Sturm'ian partial words, deBruijn partial words).

Also to mention are the very good presentations by Janusz Brzozowski on quotient complexity of star-free languages, and by Shinnosuke Seki on characterizations of bounded semilinear languages by 1- and 2-way deterministic automata, as well as the good and interesting ones by Galina Jirásková on quotient complexity of bifix-, factor-, subword-free regular languages, by Andreas Maletti (due to

an accident his coauthor couldn't come) on hyper-minimization of deterministic weighted FA over semifields, by Friedrich Otto on deterministic pushdown-CD systems of stateless deterministic R(1)-automata, by Hiroshi Umeo on new small 2-dimensional cellular automata for firing squad synchronization, and by Sarah Nelson on operations preserving primitivity of partial words with 1 hole. A very fast presentation was given by Kanduru Venkata Krishna.

A special birthday session took place on Sunday morning, celebrating 5 anniversaries. Pál Dömösi held the first laudatio in honour of Masami Ito's 70th birthday (* 1941 December 17), mentioning research of more than 40 years (codes, semigroups Q), and Masami's hobby of travelling with trains.

The second laudatio was given by Zoltán Ésik honouring Werner Kuich's 70th birthday (* 1941 June 17), talking on the scientific CV and research (continuous, inductive, partial Conway semirings, automata theory, algebraic series) of the honoured.

Sándor Horváth presented the third in honour of Christian Choffrut's 65th birthday (* 1946 May 5), talking on his long personal cooperations with Christian who speaks many languages, and on the research of the honoured (rational relations, combinatorics, Presburger logic, algebra, number theory, and *'so much much more'*).

In reverse, Zoltán Ésik's 60th birthday (* 1951 June 25) was honoured in the fourth laudatio by Werner Kuich, who presented Zoltán's scientific CV, in particular the cooperation with Stephen Bloom who passed away on October 14, 2010, on equational fix point theories, and finished with *'Ad multos annos!'*.

Finally, Benedek Nagy gave the fifth laudatio honouring Martin Kutrib's 50th anniversary (1961 August 18), presenting the scientific CV (cellular automata, descriptional complexity) of the jubilee, illustrated with photos.

All jubilees received presents as e.g.Hungarian wine.

The conference was closed on Monday early afternoon with some concluding remarks by Manfred Kudlek, András Ádám, Kanduru Venkata Krishna, and Pál Dömösi.

The proceedings, edited by Pál Dömösi and Szabolcs Iván, containing all contributions and invited lectures, unfortunately that by Christian Choffrut only as extended abstract, have been published by Institute of Mathematics and Informatics, College of Nyíregyháza.

In the coffee breaks coffee, tea, mineral water, juice, and snacks were offered. Lunch was served in the university restaurant Nagyerdei Étterem, at 10 minutes walking distance from the conference site.

Wireless access to internet was available in the conference building.

The social program started on Wednesday evening with a reception in the uni-

versity restaurant. A warm buffet, mineral water, juice, beer, Hungarian wine Nagyréde (white, rosé, red) were offered. It lasted well until 22 h.

The excursion on Friday afternoon brought us to Hortobágy, about 35 km west of Debrecen, and famous for the traditional Hungarian herdsmen, inparticular cowboy (*csikos*) culture, horse riding and excursions into Puszta (unfortunately we didn't have time for that). We could walk around, buy Hungarian souvenirs at a big market, or have some recreation in one of the cafeterias or in the *Big Csárda* (Nagycsárda).

After that, from 18 till 22 h we had the conference dinner at Látóképi Csárda (*Watching Picture Inn*), where we got typical Hungarian dishes, wine, mineral water and coffee. A music band was perfoming (not only Hungarian) gipsy tunes.

On Sunday we could watch the procession of the traditional *Flower Festival* (Debreceni Virág Karnevál) for which we got free tickets. It lasted from 8 till 12 h. In the afternoon we could visit some of the many other events such as folklore performances with dance and music, or typical Hungarian dishes.

Most participants not from Debrecen stayed in the academy house, in one of the three buildings of Kossuth Lajos Kollégium on the university campus, or in hotels in town. Pictures of AFL 2011 are available on the web site of the conference. Weather was hot and sunny, with highest temperatures above 30° C.

Thus AFL 2011 was a successful conference again, well organized and in a hospitable atmosphere. *Viszontlátásra Debrecen!*.

Report on CS&P 2011

20*th* International Workshop on Concurrency, Specification and Programming, 28-30 September 2011, Pułtusk, Poland

Manfred Kudlek

CS&P 2011, the in this series of conferences, alternating between Germany and Poland, took place in the historical town Pułtusk at river Narew in the region Mazowsze (Mazovia), 60 km north of Warszawa, from September 28 to 30, 2011. Conference site was Zamek Pułtusk (Pułtusk castle) serving as hotel and conference place (Hotel Dom Polonii, Polonia is the Polish diaspora). It is a 16th century building, partly destroyed in 2nd world war and restored after it.

CS&P 2011 was organized and supported by Universitas Varsoviensis (Faculty of Informatics), Humboldt Universität zu Berlin (Fakultät für Mathematik, Informatik und Mechanik), together with Politechnika Białystocka and NCBiR (Polish National Centre for Research and Development) project SYNAT (system Nauki i techniki).

The conference was attended by 65 participants from 7 countries, as given in the following table:

DE	14	IT	3	PL	39	RU	4
SA	3	SK	1	UK	1		

The scientific program consisted of 54 accepted (another 3 withdrawn, 2 rejected) papers as shown in the following table:

DE	$10\frac{1}{3}$	FR	$\frac{5}{8}$	IT	$3\frac{7}{8}$	PL	$27\frac{7}{8}$	RU	$3\frac{2}{3}$
SA	$2\frac{1}{8}$	SE	1	SK	2	UA	1	UK	$1\frac{1}{2}$

The program can be found on the website of CS&P 2011:

`http://www.csp2011.mimuw.edu.pl`.

The workshop was run in two parallel tracks, roughly theoretical and applied, in Sala Koncertowa and Sala Kominkowa. On Friday due to a commercial meeting, one track was shifted to Sala Biblioteki.

The presentation by Olena Yatsenko was cancelled. Due to a medical operation the contribution of Roman Rędziejowski was presented by Ludwik Czaja.

The workshop was opened on Wednesday by Marcin Szczuka, giving useful information, in particular *'You may suffer terrible pains with the internet'* (there were problems with access), and Ludwik Czaja, saying some words on CS&P.

Bulletin of the EATCS no 106, pp. 105–106, February 2012

To mention are good presentations by Jörg Bachmann on characterizations of Petri net languages, by Irina Lomazova on compositionality of boundedness and liveness for nested Petri nets, and on cellular resource-driven automata, by Frank Heitmann on liveness and reachability for elementary object nets, and restrictions of generalized state machines, by Michał Kózak on finite model property of infinitary action logics, and by Dominik Strzalka on entropy production in special algorithms.

An interesting contribution on letter frequency distribution in the *Voynich* manuscript was presented by Grzegorz Jaśkiewicz.

The proceedings, edited by Ludwik Czaja, Magdalena Kaprzak, Andrzej Skowron, and Marcin Szczuka, containing all contributions, have been published in electronic form on CD and USB stick, and are also available on the workshop website. Pictures of CS&P 2011 can be found on the conference web site.

In the coffee breaks coffee, tea, juice, mineral water and snacks were offered. Breakfast, lunch and dinner were served in the castle restaurant.

The social program consisted of a guided sightseeing tour through the old town on Thursday afternoon. It started with a tour through the castle, built in 16th century, from the tower of which there is a nice view over the town with its long market square and big town hall, and continued with a walk through the town with a visit to the 15th century Gothic church Bazylika Zwiastowania NMP (*Mary's Announcement*).

The conference dinner took place in the evening in Tawerna in the castle grounds, where we had an open fire (useful for repelling mosquitos), barbecue, salads, *bigos*, bread, draught beer, mineral water, coffee, and tea. It ended by 21:30 h.

All participants stayed in the castle hotel. Weather was nice, sunny with occasional rainfall, and highest temperatures at 20° C.

CS&P 2011 was successful again, in an interesting and recreational area. CS&P 2012 will be held in Berlin.

Report on DISCO 2011

Workshop on the Dynamics of Complex Systems
24-26 November 2011, Valparaíso, Chile

Andrés Moreira

The workshop DISCO 2011 on the Dynamics of Complex Systems (named after its Spanish title, "DInámica de Sistemas COmplejos") took place in the Chilean port of Valparaíso between the 24th and 26th of November. It was held to celebrate the 60th birthday of Eric Goles, and its venue was the Instituto de Sistemas Complejos de Valparaíso (ISCV).

The organizing committee consisted of Anahí Gajardo and Julio Aracena from the Universidad de Concepción, Andrés Moreira from the Universidad Técnica Federico Santa María, and Ricardo Espinoza from the Pontificia Universidad Católica de Valparaíso, with a strong support from the staff at the ISCV. The meeting was supported by the project IMSA (Conicyt ACT-88) as well as other institutions listed at the conference web site (`http://www.ci2ma.udec.cl/disco/`).

The celebration congregated more than 18 Chilean researchers, 32 colleagues from other countries, and 14 students. The foreign visitors came mostly from France, a token of Eric Goles' long-standing relationship with that country, but a number of other countries were also represented. Since Automata 2011, the yearly meeting on the theory of cellular automata and discrete dynamical systems, took place in Santiago in the first days of the same week, many participants made the best of their plane tickets and attended both events.

The career of Eric Goles spans the last three decades and has been mostly devoted to automata networks and related discrete dynamical systems, ranging from discrete neural networks to sand piles to cellular automata; his interest has been in the dynamics (bounds on attractors and transients) and computational capabilities of these systems. He has also created scientific centers (the Center for Mathematical Modeling at the Universidad de Chile, and the ISCV itself), directed many large projects, and presided the Chilean office for science and technology for six years. This time, however, it was his research that was celebrated, and hence the topics of most of the talks fell within or around the areas mentioned above.

The list of the speakers of the first day, along with brief descriptions of their topics, are listed below:

- **Andrés Moreira** opened the meeting with an overview of Dr Goles' career, showing the growth of his network of collaborators and describing the topics that dominated each period.

Bulletin of the EATCS no 106, pp. 107–110, February 2012

- **Cristian Calude** discussed the complexity of mathematical problems defined by computable predicates, and which would therefore be solved by an oracle for the halting problem.
- **Dominique Perrin** described recent results on the relation between bifix codes, Sturmian words and subgroups of free groups.
- **Guillaume Theyssier** recounted several results on the limit behaviour of cellular automata, in terms of both the possible and the typical configurations appearing after arbitrarily long runs.
- **Ioan Todinca** gave positive and negative results, as well as many open questions, on a model of frugal computation on a graph to which a universal vertex is added.
- **Iván Rapaport** reported results on the computational power of graph-based models of distributed computing in which each node additionally has (limited) access to a global whiteboard.
- **Michel Cosnard** talked about directed acyclic graphs with the unique directed path property, applied to the problem of minimizing the number of wavelengths used for routing.
- **Marcos Kiwi** presented a result on the growth of the number of perfect matchings of cubic bridgeless planar graphs, obtained through the study of the Ising model on their associated triangulations.
- **Martín Matamala** showed an alternative proof of a lemma by N. Thiant, arriving to it in terms of the existence of a winning strategy for a special two-player domino game.

The first day continued with a cocktail at the Lord Cochrane Museum, with a splendid view over the harbor and several delicious birthday cakes. A subset of the participants went on to finish the evening with an unofficial social activity at the typical Bar Cinzano, where a certain 60 years old mathematician is rumoured to have joined the singers. The second day of the workshop comprised the talks listed here:

- **Petr Kůrka** discussed fast arithmetical algorithms for "Möbius number system", which are representations of the unitary circle in terms of sequences of Möbius transformations.
- **Eric Rémila** talked about the avalanches and fixed points of the Kadanoff sand pile model.

- **Jarkko Kari** answered a question first formulated by Stanislaw Ulam, showing a 1-D cellular automaton which, starting from a finite configuration, produces all possible finite patterns over its alphabet.

- **Gregory Chaitin** mentioned part of his work towards a mathematical understanding of biological evolution, and commented on some other scientific novelties.

- **Gregory Lafitte** gave a talk on cellular automata and games.

- **Bruno Durand** provided an overview of the difficulties of performing universal computation in faulty cellular automata, and of even defining the problem properly.

- **Eric Goles** reminisced about his early research and its intellectual context, and went on to describe some of the main topics of his career and the questions guiding it.

- **Alejandro Maass** reviewed a number of rigidity results on cellular automata; in particular, results on the iteration of measures and on invariant measures.

- **Hans Herrmann** in his "Apollonian variations" described his work on constructing space-filling Apollonian packings of bearings where contiguous discs rotate in opposite senses.

- **Sergio Rica** described his recent work with Eric and Nicolás Goles on Schelling's social segregation model; he discussed qualitative and quantitative observations from a physical point of view.

- **Pablo Marquet**, an ecologist, talked about several mathematical models on which he has worked, ranging from the formation of plant stripes in the desert to mitigation strategies for climate change.

The last day of the meeting was a Saturday, and the program finished with a barbecue for lunch. The crowd waited for it by listening to the last set of talks:

- **Bruno Martin** sketched a "how-to" for the construction of universal computation, by describing and comparing the strategies and chains of simulation used in several of Eric Goles' results.

- **Nicolas Ollinger** showed his construction of aperiodic tile sets based on 2×2 substitution systems, and related it to the work "hidden" in an appendix of Robert Berger's 1966 thesis.

- **Nicolas Schabanel** gave some reflexions on the relation between computer science and complex systems theory, discussing different ways in which the first has contributed (and still can contribute) to the second.
- **Henning Mortveit** described how the tools from group theory may be used to assess the long-term dynamics of asynchronous discrete dynamical systems.
- **Julio Aracena** talked about update schedules in Boolean networks, and how their relation to the dynamics can be studied through the "update digraphs" they induce.
- **Jacques Demongeot** made a tour de force through several levels of biological complexity, discussing the requisites for the robustness of different systems. In addition, he shared some memories of the early 80's in Grenoble, when Eric Goles was first a student and then a CNRS researcher.

In summary, it was a successful meeting, with great contributions by all the participants (and here we do *not* refer to the bottle of wine which was required as the sole registration fee). The talks dealt with a number of areas in theoretical computer science and discrete mathematics; there were even some forays into other disciplines. However, a sense of unity was provided by their relation to the research done in the last 30 years by Eric Goles and his collaborators, both in the wide world and in the "Chilean school" that has flourished under his guidance.

The slides from the talks can be found at the workshop's web site. Videos of the talks are available through the web site of the ISCV (`http://www.iscv.cl`). In addition, a special issue of *Theoretical Computer Science* will be published, based on the research and surveys presented during the meeting.

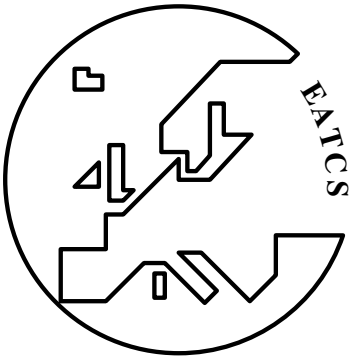
European

Association for

Theoretical

Computer

Science



E A T C S

EATCS

HISTORY AND ORGANIZATION

EATCS is an international organization founded in 1972. Its aim is to facilitate the exchange of ideas and results among theoretical computer scientists as well as to stimulate cooperation between the theoretical and the practical community in computer science.

Its activities are coordinated by the Council of EATCS, which elects a President, Vice Presidents, and a Treasurer. Policy guidelines are determined by the Council and the General Assembly of EATCS. This assembly is scheduled to take place during the annual **I**nternational **C**olloquium on **A**utomata, **L**anguages and **P**rogramming (ICALP), the conference of EATCS.

MAJOR ACTIVITIES OF EATCS

- Organization of ICALP;
- Publication of the "Bulletin of the EATCS;"
- Award of research and academic careers prizes, including the "EATCS Award," the "Gödel Prize" (with SIGACT) and best papers awards at several top conferences;
- Active involvement in publications generally within theoretical computer science.

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: ETAPS (The European Joint Conferences on Theory and Practice of Software), STACS (Symposium on Theoretical Aspects of Computer Science), MFCS (Mathematical Foundations of Computer Science), LICS (Logic in Computer Science), ESA (European Symposium on Algorithms), 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: algorithms, computational complexity, game theory, automata theory, formal language theory, logic, semantics, and theory of programming languages, foundations of networked computation, parallel, distributed, and external memory computing, foundations of logic programming, models of concurrent, distributed and mobile systems, software specification, computational geometry, data types and data structures, models for complex networks, theory of security.

SITES OF ICALP MEETINGS:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, Great Britain 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Nafplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, Great Britain 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologne, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisbon, Portugal 2005
- Venezia, Italy 2006
- Wrocław, Poland 2007
- Reykjavik, Iceland 2008
- Rhodes, Greece 2009
- Bordeaux, France 2010
- Zürich, Switzertland 2011

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for *rapid* publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D.-Theses;
- Entertainments and pictures related to computer science.

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at `http://www.eatcs.org/bulletin`. Questions and proposals can be addressed to the Editor by email at `bulletin@eatcs.org`.

(3) OTHER PUBLICATIONS

EATCS has played a major role in establishing what today are some of the most prestigious publication within theoretical computer science.

These include the *EATCS Texts* and the *EATCS Monographs* published by Springer-Verlag and launched during ICALP in 1984. The Springer series include *monographs* covering all areas of theoretical computer science, and aimed at the research community and graduate students, as well as *texts* intended mostly for the graduate level, where an undergraduate background in computer science is typically assumed.

Updated information about the series can be obtained from the publisher.

The editors of the series are W. Brauer (Munich), J. Hromkovic (Aachen), G. Rozenberg (Leiden), and A. Salomaa (Turku). Potential authors should contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:
Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands
who acknowledges EATCS membership and forwards the order to Springer-Verlag.

The journal *Theoretical Computer Science*, founded in 1975 on the initiative of EATCS, is published by Elsevier Science Publishers. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.
The Editors-in-Chief of the journal currently are G. Ausiello (Rome), D. Sannella (Edinburgh), G. Rozenberg (Leiden), and M.W. Mislove (Tulane).

ADDITIONAL EATCS INFORMATION

For further information please visit `http://www.eatcs.org`, or contact the President of EATCS:
Prof. Dr. Burkhard Monien, Department of Computer Science
Universität Paderborn, Fürstenalle 11, 33102 Paderborn, Germany
Email: `president@eatcs.org`

EATCS MEMBERSHIP

DUES

The dues are € 30 for a period of one year. A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.
In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for € 25 per year. Additional € 25 fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe.

HOW TO JOIN EATCS

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website `www.eatcs.org`, where you will find an online registration form and the possibility of secure online payment. Alternatively, a subscription form can be downloaded from `www.eatcs.org` to be filled and sent together with the annual dues (or a multiple thereof, if membership for multiple years is required) to the **Treasurer** of EATCS:
Prof. Dr. Dirk Janssens, Dept. of Math. and Computer Science, University of Antwerp
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