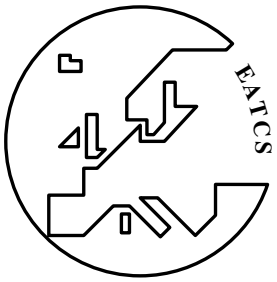
ISSN 0252–9742

# Bulletin

of the

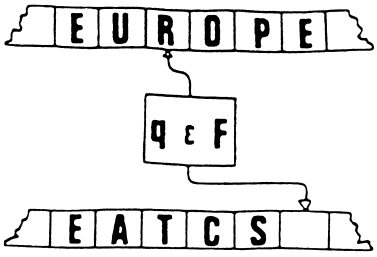
## European Association for Theoretical Computer Science

# EATCS




Number 134 June 2021

## Council of the European Association for Theoretical Computer Science




| | | |
|---|---|---|
| President: | Artur Czumaj | United Kingdom |
| Vice Presidents: | Anca Muscholl | France |
| | Giuseppe Persiano | Italy |
| Treasurer: | Jean-Francois Raskin | Belgium |
| Bulletin Editor: | Stefan Schmid | Austria |

| | | | |
|---|---|---|---|
| Ivona Bezakova | USA | Claire Mathieu | France |
| Michael Benedikt | UK | Elvira Mayordomo | Spain |
| Patricia Bouyer | France | Emanuela Merelli | Italy |
| Artur Czumaj | UK | Anca Muscholl | France |
| Javier Esparza | Germany | Luke Ong | UK |
| Thore Husfeldt | Sweden, Denmark | Maria Serna | Spain |
| Giuseppe F. Italiano | Italy | Jiri Sgall | Czech Republic |
| Christos Kaklamanis | Greece | Ola Svensson | Switzerland |
| Samir Khuller | USA | Till Tantau | Germany |
| Fabian Kuhn | Germany | Sophie Tison | France |
| Slawomir Lasota | Poland | Gerhard Wöeginger | The Netherlands |

### 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) |
| Burkhard Monien | (2009–2012) | Luca Aceto | (2012–2016) |
| Paul Spirakis | (2016–2020) | | |

| | | |
|---|---|---|
| Secretary Office: | Efi Chita | Greece |
| | Emanuela Merelli | Italy |

# EATCS Council Members

## email addresses

Ivona Bezakova . . . . . ib@cs.rit.edu
Michael Benedikt . . . . . michael.benedikt@cs.ox.ac.uk
Patricia Bouyer . . . . . bouyer@lsv.fr
Artur Czumaj . . . . . A.Czumaj@warwick.ac.uk
Javier Esparza . . . . . esparza@in.tum.de
Thore Husfeldt . . . . . thore@itu.dk
Giuseppe F. Italiano . . . . . giuseppe.italiano@uniroma2.it
Christos Kaklamanis . . . . . kakl@ceid.upatras.gr
Samir Khuller . . . . . samirkhuller@gmail.com
Fabian Kuhn . . . . . kuhn@cs.uni-freiburg.de
Slawomir Lasota . . . . . sl@mimuw.edu.pl
Claire Mathieu . . . . . cmathieu@di.ens.fr
Elvira Mayordomo . . . . . elvira@unizar.es
Emanuela Merelli . . . . . emanuela.merelli@unicam.it
Anca Muscholl . . . . . anca@labri.fr
Luke Ong . . . . . luke.Ong@cs.ox.a.uk
Jean-Francois Raskin . . . . . jraskin@ulb.ac.be
Maria Serna . . . . . mjserna@cs.upc.edu
Stefan Schmid . . . . . schmiste@gmail.com
Jiri Sgall . . . . . sgall@iuuk.mff.cuni.cz
Ola Svensson . . . . . Ola.Svensson@epfl.ch
Till Tantau . . . . . tantau@tcs.uni-luebeck.de
Sophie Tison . . . . . sophie.tison@lifl.fr
Gerhard Wöeginger . . . . . g.j.woeginger@math.utwente.nl

Bulletin Editor: Stefan Schmid, Vienna, Austria
Cartoons: DADARA, Amsterdam, The Netherlands

---

The bulletin is entirely typeset by pdfTEX and ConTEXt in TXfonts.

---

All contributions are to be sent electronically to

bulletin@eatcs.org

and must be prepared in LaTeX 2$\varepsilon$ using the class `beatcs.cls` (a version of the standard LaTeX 2$\varepsilon$ `article` class). All sources, including figures, and a reference PDF version must be bundled in a ZIP file.

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

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

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

---

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

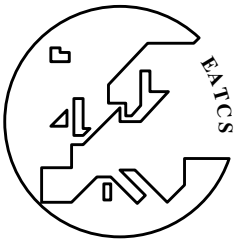
---

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

---

# Table of Contents

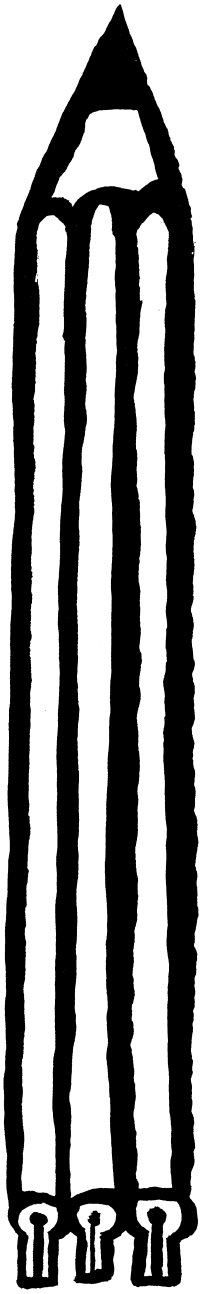
# EATCS Matters



*Dear EATCS members,*

*I hope my letter finds you and your family safe and in good health. We are living in the times marked by the global coronavirus pandemic and this has a major impact on our lives and our scientific activities. Most of our research activities are still run remotely and online, and it seems that this will stay with us for the next months. However, with the improved pandemic situation and progress in global vaccination, we all hope to see a return to better times soon; already now, there are some conferences announced to be run in the hybrid mode, with some physical participation as early as this summer. While the current situation is a major challenge for our scientific community, on a bright side, our community has been developing many new and exciting initiatives: we see more online seminars and workshops, a lot of remote collaborative research, and for many of us free or low-cost online conferences provide a great opportunity for broader participation and the increase of visibility of our research through these events. And most importantly, we see some fantastic research done by our community; and so I take the opportunity to wish you all the best and much success for your work.*

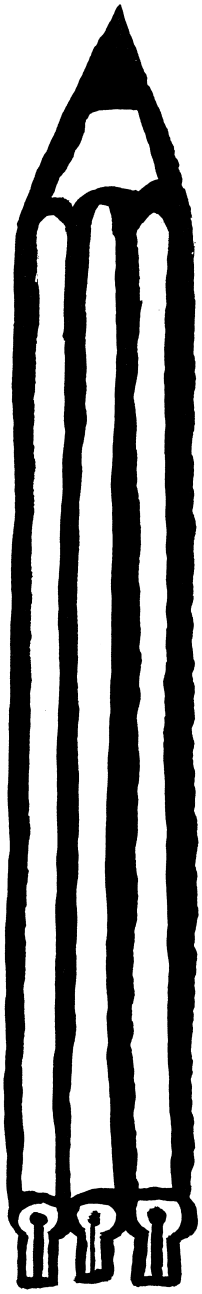
*This will be the first issue of the Bulletin under the new leadership of Stefan Schmid (University of Vienna), who has recently replaced Kazuo Iwama as a new editor in chief. Kazuo Iwama has been the editor in chief of the Bulletin for almost*



*eight years, since the October 2013 issue. During Kazuo's energetic leadership our flagship publication has consistently been of excellent quality, supporting the theoretical computer science community in many ways. I have learned much from Kazuo, and let me use this opportunity to thank him for many fantastic years in running the Bulletin.*

*And as for the new leadership, I wish Stefan the best of luck with his new position. I trust that you will offer your assistance in Stefan's efforts to continue improving the quality and the impact of the flagship publication of the EATCS, in cooperation with the Council of the EATCS and following on Kazuo's footsteps.*

*As usual, the June issue of the Bulletin will be available just before ICALP, the flagship conference of the EATCS and an important meeting of the theoretical computer science community world-wide. The 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021), will be held July 12-16, 2021 (URL: http://easyconferences.eu/icalp2021/). While we originally all hoped to meet at ICALP in Glasgow, Scotland, at the end the conference will be run online/virtually, as in the last year. Still, the conference chair Simon Gay, supported by his colleagues in Glasgow, promises us an exciting scientific event. I am very grateful to Simon Gay and his team (including Ornela Dardha, Gethin Norman, Oana Andrei, Michele Sevegnani, Jess Enright, Sofiat Olaosebikan, David Manlove, Kitty Meeks, Alice Miller) for the extraordinary work they have done in organizing ICALP 2021.*



*The Programme Committee chairs Nikhil Bansal (track A) and James Worrell (track B) and their PCs have done fantastic job selecting an impressive collection of papers, 108 accepted papers in track A and 29 in track B out of 463 submissions (362 for track A and 101 for track B). The acceptance rate was 29.6 percent. The programme of ICALP 2021 will highlight research across many areas within theoretical computer science. I invite you to watch the talks even outside your own research field.*
*The best paper awards at ICALP 2021 will go to the following two articles:*

- *Track A: Sayan Bhattacharya and Peter Kiss. Deterministic Rounding of Dynamic Fractional Matchings;*
- *Track B: Antoine Amarilli, Louis Jachiet and Charles Paperman. Dynamic Membership for Regular Languages.*

*The best student paper award for a paper that is solely authored by a student will go to the following paper from track A:*

- *Or Zamir. Breaking the $2^n$ barrier for 5-coloring and 6-coloring.*

*Congratulations to the authors of the award-receiving papers!*

*In addition to regular research talks, ICALP 2021 will feature six invited talks delivered by Christel Baier (Technical University of Dresden), Andrei Bulatov (Simon Fraser University), Keren Censor-Hillel (Technion), Toniann Pitassi (University of Toronto), Adi Shamir (Weizmann Institute of Science), and David Woodruff (Carnegie Mellon University).*

*Apart from the invited and contributed talks, ICALP 2021 will feature two special presentations:*

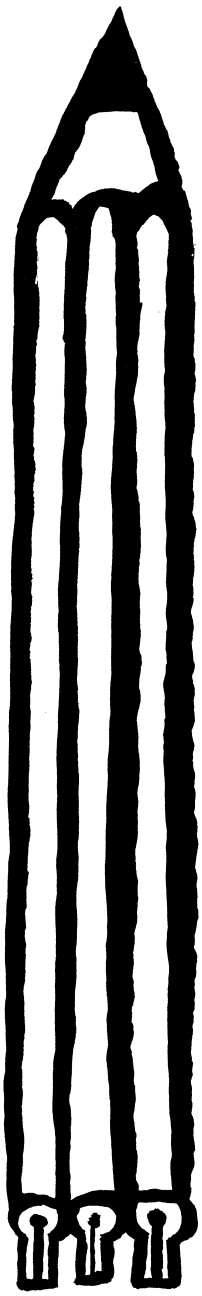
- *of the EATCS Award 2021 to Toniann Pitassi (University of Toronto) and*
- *of the Presburger Award 2021 to Shayan Oveis Gharan (University Washington).*

*Moreover, during the conference, we will honor the recipients of the 2020 EATCS Distinguished Dissertation Award and the new group of EATCS Fellows.*
*The recipients of the 2020 EATCS Distinguished Dissertation Award are*

- *Talya Eden, Tel Aviv University (advisor: Dana Ron),*
- *Marie Fortin, Université Paris-Saclay (advisors: Paul Gastin and Benedikt Bollig),*
- *Vera Traub, University of Bonn (advisor: Jens Vygen).*

*The new group of EATCS Fellows (class 2021) recognized for their scientific achievements in the field of Theoretical Computer Science consists of*

- *Luca Aceto (Reykjavik University and Gran Sasso Science Institute),*
- *Rajeev Alur (University of Pennsylvania),*
- *Samir Khuller (Northwestern University),*
- *David Peleg (Weizmann Institute of Science),*



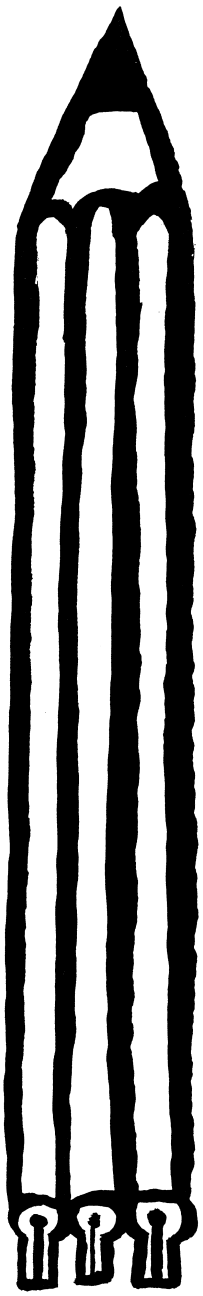
- *Davide Sangiorgi (University of Bologna), and*
- *Saket Saurabh (The Institute of Mathematical Sciences, Chennai).*

*Congratulations to the award winners and new EATCS Fellows!*
*On behalf of the EATCS, I also heartily thank the members of the awards, dissertation and fellow committees for their work in the selection of this stellar set of award recipients and fellows. It will be a great honor to celebrate the work of these colleagues during ICALP 2021. (More details about the EATCS Award 2021, the Presburger Award 2021, the 2020 EATCS Distinguished Dissertation Award, and the EATCS Fellows are presented on the later pages of this issue of the Bulletin.)*

*Finally, let me also mention that ICALP has joined the SafeToC (http://safetoc.org/) initiative, aiming to help prevent and combat harassment in the Theory of Computing community. As the result, all ICALP 2021 participants will be required to agree to follow an appropriate SafeToC Code of Conduct. I believe this is a very important initiative that supports our scientific community and we all should endorse it.*

*ICALP 2021 will also have seven satellite workshops co-located with the main conference, taking place (online) on Sunday and Monday before the main event:*

- *Algorithmic Aspects of Temporal Graphs IV,*
- *VEST: Verification of Session Types,*



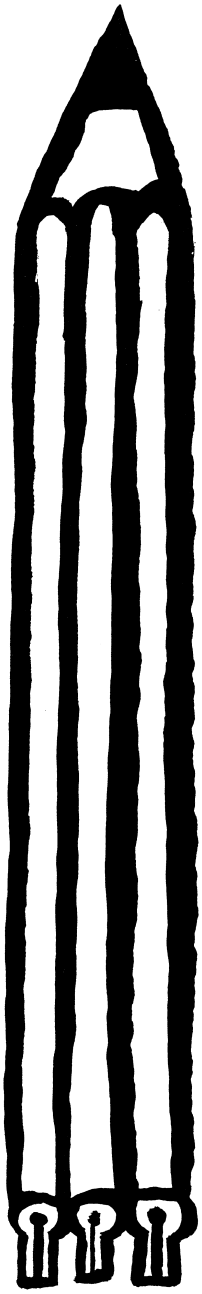
- *2nd Workshop on Programming Research in Mainstream Languages (PRiML 2021),*
- *Graph Width Parameters: from Structure to Algorithms (GWP 2021),*
- *Combinatorial Reconfiguration,*
- *Formal Methods Education Online: Tips, Tricks & Tools, and*
- *Flavours of Uncertainty in Verification, Planning and Optimization (FUNCTION).*

*As usual, a more detailed report on the ICALP 2021 conference will be published in the October 2021 issue of the Bulletin.*

*Also, please allow me to remind you about three other EATCS affiliated conferences that will be taking place later this year.*

- *MFCS 2021, the 46th International Symposium on Mathematical Foundations of Computer Science, will be held in Tallinn, Estonia, August 23-27, 2021 (https://compose.ioc.ee/mfcs/).*
- *ESA 2021, the 29th Annual European Symposium on Algorithms, will be held in Lisbon, Portugal, September 6-8, 2021 (http://algo2021.tecnico.ulisboa.pt/ESA2021/).*
- *DISC 2021, the 34th International Symposium on Distributed Computing, will be held in Freiburg, Germany, October 4-8, 2021 (http://www.disc-conference.org/wp/disc2021/).*

*We still do not know whether these conferences will take place online or as hybrid conference, or maybe even physically.*



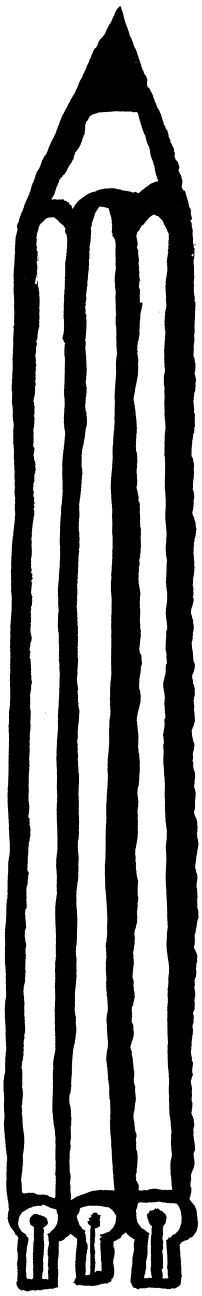
*In the recent months we have seen announcements of numerous further awards given to the members of theoretical computer science. While more details about many of these awards can be found on the pages of this Bulletin, let me list the main highlights here.*

*The Gödel Prize for outstanding papers in the area of theoretical computer science is sponsored jointly by the EATCS and the ACM SIGACT. This year it has been awarded for the seminal work on the constraint satisfaction problem to the following three papers:*

- *Andrei Bulatov. The Complexity of the Counting Constraint Satisfaction Problem. J. ACM 60(5): 34:1-34:41, 2013.*
- *Martin E. Dyer and David Richerby. An Effective Dichotomy for the Counting Constraint Satisfaction Problem. SIAM J. Computing 42(3): 1245-1274, 2013.*
- *Jin-Yi Cai and Xi Chen. Complexity of Counting CSP with Complex Weights. J. ACM 64(3): 19:1-19:39, 2017.*

*The Edsger W. Dijkstra Prize in Distributed Computing is awarded for outstanding papers on the principles of distributed computing, and is sponsored jointly by the ACM Symposium on Principles of Distributed Computing (PODC) and the EATCS Symposium on Distributed Computing (DISC). The 2021 Dijkstra Prize has been awarded to the following seminal paper*

- *Paris C. Kanellakis and Scott A. Smolka. CCS Expressions, Finite State Processes, and Three Problems of*

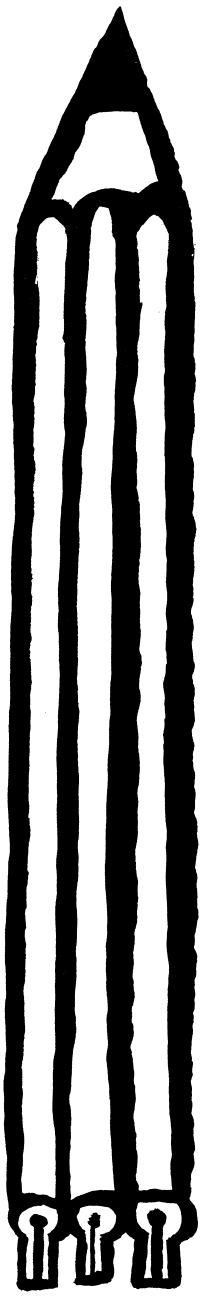


*Equivalence*, Information and Computation *86(1):43-68, May 1990.*

*The Alonzo Church Award for outstanding contributions to logic and computation is awarded annually in a collaboration of the European Association for Theoretical Computer Science (EATCS), the ACM Special Interest Group on Logic (SIGLOG), the European Association for Computer Science Logic (EACSL), and the Kurt Gödel Society (KGS). The 2021 Alonzo Church Award has been awarded to Georg Gottlob, Christoph Koch, Reinhard Pichler, Klaus U. Schulz, and Luc Segoufin for fundamental work on logic-based web data extraction and querying tree-structured data, published in the following papers:*

- *Georg Gottlob and Christoph Koch. Monadic Datalog and the Expressive Power of Languages for Web Information Extraction. J. ACM 51(1): 74-113, 2004,*
- *Georg Gottlob, Christoph Koch, and Klaus U. Schulz. Conjunctive Queries Over Trees. J. ACM 53(2): 238-272, 2006,*
- *Georg Gottlob, Christoph Koch, and Reinhard Pichler. Efficient Algorithms for Processing XPath Queries. ACM TODS 30(2): 444-491, 2005, and*
- *Georg Gottlob, Christoph Koch, Reinhard Pichler, and Luc Segoufin. The Complexity of XPath Query Evaluation and XML Typing. J. ACM 52(2): 284-335, 2005.*

*EATCS also sponsors the Best ETAPS Paper Award 2021 for the best theory paper at*



*ETAPS, which this year was awarded to the following two papers:*

- *Robin Piedeleu and Fabio Zanasi, A String Diagrammatic Axiomatisation of Finite-State Automata, and*
- *Bartek Klin, Sławomir Lasota, and Szymon Toruńczyk, Nondeterministic and co-Nondeterministic Implies Deterministic, for Data Languages.*

*Finally, I am very happy to report that this year, the Norwegian Academy of Science and Letters awarded the Abel Prize for 2021 to two giants of the theoretical computer science community,*

- *László Lovász (Alfréd Rényi Institute of Mathematics and Eötvös Loránd University in Budapest) and*
- *Avi Wigderson (Institute for Advanced Study, Princeton).*

*The Abel Prize is one of the biggest and most prestigious prizes in mathematics, awarded annually by the King of Norway to one or more outstanding mathematicians, and is directly modeled after the Nobel Prizes. I am especially thrilled by the citation, which closely links the advances in mathematics with our field: "for their foundational contributions to theoretical computer science and discrete mathematics, and their leading role in shaping them into central fields of modern mathematics."*

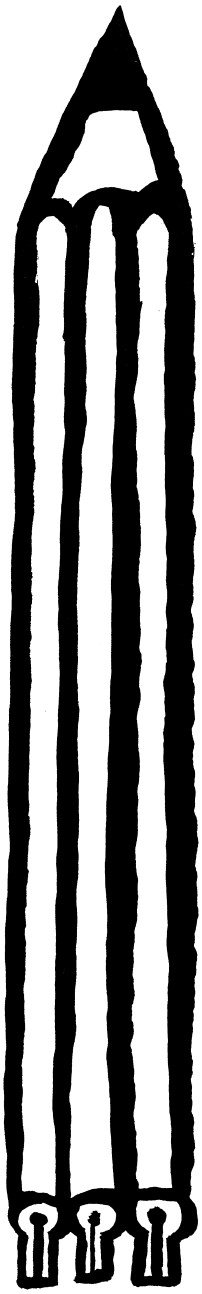
*As usual, let me close this letter by reminding you that you are always most welcome to send me your comments, criticisms and suggestions for improving the impact of the EATCS on the Theoretical*

*Computer Science community at president@eatcs.org. We will consider all your suggestions and criticisms carefully.*

*I look forward to seeing many of you at ICALP 2021 and to discussing ways of improving the impact of the EATCS within the theoretical computer science community at the general assembly.*

*Artur Czumaj*
*University of Warwick, UK*
*President of EATCS*
president@eatcs.org

*June 2021*



*Dear Reader,*

*With this 134th issue of the Bulletin of the EATCS, I will take over the editor-in-chief position from Kazuo Iwama, who led the Bulletin for almost 10 years now. On this occasion, I would like to thank Kazuo for his fantastic efforts to run the Bulletin so successfully for so many years. I still remember well when Kazuo visited Berlin during one of his numerous travels in Europe, probably around 2014, and invited me to have a coffee with him at Ernst Reuter Platz. I think he visited Rolf Niedermeier at that time, and used the opportunity to find a new editor for the Distributed Computing column. When Kazuo asked me whether I like to join his team, I immediately accepted, and 6 years and 17 issues later, I am grateful for all the opportunities the position opened for me and for the great collaboration I had with Kazuo and all the authors who contributed wonderful articles to the Bulletin.*

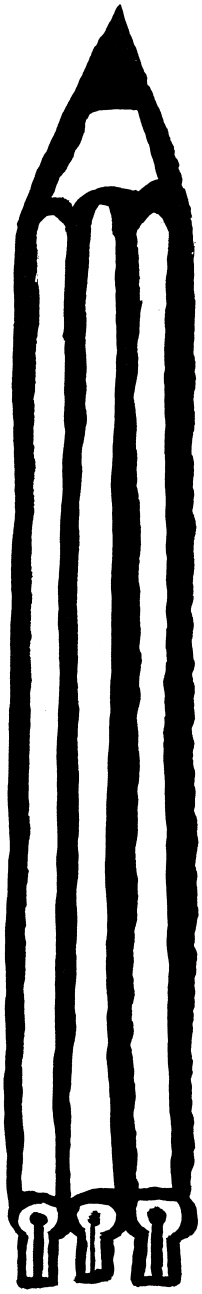
*I am very much looking forward to working together with Artur, the president of the EATCS, with Giovanni, Juraj, Nobuko, Thomas, Vikraman, and Yuri, our column editors, with Efi from the secretary office of the EATCS, as well as with everyone else involved in the Bulletin.*

*My plan is to build upon the successful model and continue the Bulletin in a similar format. In parallel, I also start exploring innovative ideas on how the Bulletin can be evolved and rendered even more attractive for the community. While I already have some ideas myself, I am*

*grateful about any suggestions and inputs you may have. Accordingly, if something comes to your mind, please do not hesitate to contact me at schmiste@gmail.com.*

*Thank you for your interest in the Bulletin and I would like now to give the word to Kazuo!*

*Stefan Schmid, Vienna*
*June 2021*



*Many thanks, Stefan, for giving me this extra space in your first Letter. Congratulations and I am confident that our BEATCS will continue, and enter a new stage of, its further enrichment under your editorship.*

*I have served three EATCS Presidents, Luca, Paul, and Artur for almost a decade, many thanks for their continuous support and for providing me a full sense of accomplishment. I would like to express my sincere appreciation to our readers, hundreds of authors, and especially to Column Editors, Vikraman Arvind, Thomas Erlebach, Panagiota Fatourou, Yuri Gurevich, Juraj Hromkovic, Giovanni Pighizzini, Gerhard Woeginger, Nobuko Yashida, and of course to you, Stefan. Technical Columns are definitely the most important part of BEATCS and their contribution is beyond description. Finally Efi, without your great cooperation, it would have been totally impossible for me to make this achievement. Thank you, everyone, again and again.*

*Kazuo Iwama, Kyoto*
*June 2021*

# The EATCS Award 2020

## Laudatio for Toniann (Toni) Pitassi

The EATCS Award 2021 is awarded to

**Toniann (Toni) Pitassi**

University of Toronto, as the recipient of the 2021 EATCS Award for her fundamental and wide-ranging contributions to computational complexity, which includes proving long-standing open problems, introducing new fundamental models, developing novel techniques and establishing new connections between different areas. Her work is very broad and has relevance in computational learning and optimisation, verification and SAT-solving, circuit complexity and communication complexity, and their applications.

The first notable contribution by Toni Pitassi was to develop lifting theorems: a way to transfer lower bounds from the (much simpler) decision tree model for any function f, to a lower bound, the much harder communication complexity model, for a simply related (2-party) function f'. This has completely transformed

our state of knowledge regarding two fundamental computational models, query algorithms (decision trees) and communication complexity, as well as their relationship and applicability to other areas of theoretical computer science. These powerful and flexible techniques resolved numerous open problems (e.g., the super quadratic gap between probabilistic and quantum communication complexity), many of which were central challenges for decades.

Toni Pitassi has also had a remarkable impact in proof complexity. She introduced the fundamental algebraic Nullstellensatz and Ideal proof systems, and the geometric Stabbing Planes system. She gave the first nontrivial lower bounds on such long-standing problems as weak pigeon-hole principle and models like constant-depth Frege proof systems. She has developed new proof techniques for virtually all proof systems, and new SAT algorithms. She found novel connections of proof complexity, computational learning theory, communication complexity, circuit complexity, LP hierarchies, graph theory and more.

In the past few years Toni Pitassi has turned her attention to the field of algorithmic fairness, whose social importance is rapidly growing, in particular providing novel concepts and solutions based on causal modelling.

Summarising, Toni Pitassi's contributions have transformed the field of computational complexity and neighbouring areas of theoretical computer science, and will continue to have a lasting impact. Furthermore, she is an outstanding mentor, great teacher and a dedicated TCS community member.

The EATCS Award Committee 2021

- Johan Håstad
- Marta Kwiatkowska (chair)
- Éva Tardos

The EATCS Award is given to acknowledge extensive and widely recognized contributions to theoretical computer science over a life-long scientific career.

The Award will be assigned during a ceremony that will take place during ICALP 2021, where the recipient will give an invited presentation during the Award Ceremony.

The following is the list of the previous recipients of the EATCS Awards:

2020 Mihalis Yannakakis
2019 Thomas Henzinger
2018 Noam Nisan
2017 Éva Tardos
2016 Dexter Kozen
2015 Christos Papadimitriou
2014 Gordon Plotkin
2013 Martin Dyer
2012 Moshe Y. Vardi
2011 Boris (Boaz) Trakhtenbrot
2010 Kurt Mehlhorn
2009 Gérard Huet
2008 Leslie G. Valiant
2007 Dana S. Scott
2006 Mike Paterson
2005 Robin Milner
2004 Arto Salomaa
2003 Grzegorz Rozenberg
2002 Maurice Nivat
2001 Corrado Böhm
2000 Richard Karp

# 2021 Alonzo Church Award for Outstanding Contributions to Logic and Computation

The European Association for Theoretical Computer Science (EATCS), the ACM Special Interest Group on Logic (SIGLOG), the European Association for Computer Science Logic (EACSL), and the Kurt Goedel Society (KGS) are pleased to announce that

- **Georg Gottlob**, **Christoph Koch**, **Reinhard Pichler**, **Klaus U. Schulz**, and **Luc Segoufin**

have been selected as the winners of the *2021 Alonzo Church Award for Outstanding Contributions to Logic and Computation* for fundamental work on logic-based web data extraction and querying tree-structured data, published in:

1. Georg Gottlob and Christoph Koch. Monadic Datalog and the Expressive Power of Languages for Web Information Extraction. *Journal of the ACM* 51(1): 74–113, January 2004 (DOI: 10.1145/962446.962450),

2. Georg Gottlob, Christoph Koch, and Klaus U. Schulz. Conjunctive Queries Over Trees. *Journal of the ACM* 53(2): 238–272, March 2006 (DOI: 10.1145/1131342.1131345),

3. Georg Gottlob, Christoph Koch, and Reinhard Pichler. Efficient Algorithms for Processing XPath Queries. *ACM Transactions on Database Systems* 30(2): 444–491, June 2005 (DOI: 10.1145/1071610.1071614), and

4. Georg Gottlob, Christoph Koch, Reinhard Pichler, and Luc Segoufin. The Complexity of XPath Query Evaluation and XML Typing. *Journal of the ACM* 52(2): 284–335, March 2005 (DOI: 10.1145/1059513.1059520).

Paper (1) establishes a comprehensive logical theory of Web data extraction. At its core, this is the problem of selecting relevant nodes (subtrees) from HTML text. While the set of relevant nodes can be expressed in Monadic Second-Order logic (MSO) over finite trees, MSO has high computationally complexity. The authors prove that Monadic Datalog on trees has exactly the same expressive power

as full MSO and that, surprisingly, evaluating Monadic Datalog is feasible in time linear in the size of query and input tree. These results greatly influenced theoretical and applied research, and gave rise to logic-based systems for data extraction that have been successfully used in industry.

Papers (2,3,4) present deep investigations into logical queries over tree-structured data. The complexity of evaluating XPath, a key technology in Web browsers and other systems, was unclear, and available implementations required exponential time. Paper (2) gives a full characterization of, and a dichotomy theorem for, the complexity of conjunctive queries on various representations of trees. Paper (3) shows that the full XPath standard can be evaluated in PTIME and proposes a logical core which has become seminal to research efforts at the intersection of Web data processing and (modal) logics. Finally, paper (4) establishes the precise complexity of evaluating XPath fragments.

The 2021 Alonzo Church Award Committee:

- Mariangiola Dezani,
- Thomas Eiter,
- Javier Esparza (chair),
- Radha Jagadeesan, and
- Igor Walukiewicz.

The list of the previous recipients of the Alonzo Church Award for Outstanding Contributions to Logic and Computation is available at `https://siglog.org/awards/alonzo-church-award/`.

# The Presburger Award 2021

## Laudatio for Shayan Oveis Gharan

The 2021 Presburger Committee has unanimously selected

**Shayan Oveis Gharan**

as the recipient of the 2021 EATCS Presburger Award for Young Scientists for his creative, profound, and ambitious contributions to the Traveling Salesman problem.

Traveling Salesman is a drosophilia of theoretical computer science: immediately accessible, intellectually appealing, computationally difficult, and extremely well-studied. It serves as a milestone and showcase for progress in our scientific understanding of the design and analysis of algorithms.

Oveis Gharan's work began during his PhD thesis, in a paper that presents an asymptotically sublogarithmic approximation guarantee for the Traveling Salesman problem for the general (or, "asymmetric") case. The primary tool is the negative dependence between the presence of edges in a certain distribution on random spanning trees of a graph, a theme that has matured and expanded throughout

much of the recipient's work. In a remarkable tour de force, Oveis Gharan and his coauthors followed this up with a $(3/2 - \varepsilon)$-approximation for the symmetric graphic case, and later also for the metric case. These results rely on a deeper understanding of negative dependence through the lens of strongly Rayleigh measures and real-stable polynomials, the polyhedral structure of the Held–Karp polytope, and the combinatorics of near-minimum cuts in a graph.

Besides his work on TSP, Oveis Gharan exhibits an amazingly consistent ability to make progress on many other problems that had remained open for decades. For example, a Cheeger inequality proved by Oveis Gharan and co-authors was the main technical tool in Miclo's proof of a 40-year old conjecture of Simon and Høegh-Krohn in the field of mathematical physics. The interplay between analysis, probability, and combinatorics is a hallmark of his research programme and a sterling example of how questions arising from theoretical computer science lead to profound progress in related fields.

The Presburger Committee 2021

- Mikołaj Bojańczyk
- Thore Husfeldt (chair)
- Meena Mahajan

The Presburger Award is given to a young scientist (in exceptional cases to several young scientists) for outstanding contributions in theoretical computer science, documented by a published paper or a series of published papers. The award is named after Mojzesz Presburger who accomplished his path-breaking work on decidability of the theory of addition (which today is called Presburger arithmetic) as a student in 1929.

The award includes an amount of 1000 € and an invitation to ICALP 2021 for a lecture.

The following is the list of the previous recipients of the EATCS Pressburger Awards:

2020 Dmitriy Zhuk
2019 Karl Bringmann and Kasper Green Larsen
2018 Aleksander Mądry
2017 Alexandra Silva
2016 Mark Braverman
2015 Xi Chen
2014 David Woodruff
2013 Erik Demaine
2012 Venkatesan Guruswami and Mihai Patrascu
2011 Patricia Bouyer-Decitre
2010 Mikolaj Bojanczyk

# 2021 Edsger W. Dijkstra Prize in Distributed Computing

The Edsger W. Dijkstra Prize in Distributed Computing is awarded for outstanding papers on the principles of distributed computing, whose significance and impact on the theory or practice of distributed computing have been evident for at least a decade. It is sponsored jointly by the EATCS Symposium on Distributed Computing (DISC) and the ACM Symposium on Principles of Distributed Computing (PODC). The prize is presented annually, with the presentation taking place alternately at DISC and PODC. The committee decided to award the 2021 Edsger W. Dijkstra Prize in Distributed Computing to

- **Paris C. Kanellakis** and **Scott A. Smolka**

for their paper:

- CCS Expressions, Finite State Processes, and Three Problems of Equivalence, *Information and Computation*, Volume 86, Issue 1, pages 43—68, 1990.

A preliminary version of this paper appeared in the *Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing (PODC 1983)*, pages 228—240.

This paper was a foundational contribution to the fundamental challenge of assigning semantics to concurrent processes, for specification and verification. It addressed the computational complexity of the previously introduced celebrated notion of behavioral equivalence, a cornerstone of Milner's Calculus of Communicating Systems (CCS), aimed at tackling semantics by considering equivalence classes.

With the publication of their PODC 1983 paper, Kanellakis and Smolka pioneered the development of efficient algorithms for deciding behavioral equivalence of concurrent and distributed processes, especially bisimulation equivalence, which is the cornerstone of the process-algebraic approach to modeling and verifying concurrent and distributed systems. Specifically, the main result of their paper is what has come to be known as the K-S Relational Coarsest Partitioning

algorithm, which at the time was a new combinatorial problem of independent interest.

The paper also presented complexity results that showed certain behavioral equivalences are computationally intractable. Collectively, Kanellakis and Smolka's results founded the subdiscipline of algorithmic process theory, and helped jump-start the field of Formal Verification.

The 2021 Edsger W. Dijkstra Prize Award Committee:

- Keren Censor-Hillel (chair), Technion
- Pierre Fraigniaud, Université de Paris and CNRS
- Cyril Gavoille, LaBRI — Université de Bordeaux
- Seth Gilbert, National University of Singapore
- Andrzej Pelc, Université du Québec en Outaouais
- David Peleg, Weizmann Institute of Science

The list of the previous recipients of the Edsger W. Dijkstra Prize in Distributed Computing is available at `https://www.podc.org/dijkstra/`.

# EATCS Distinguished Dissertation Award for 2020

EATCS is proud to announce that, after examining the nominations received from our research community, the EATCS Distinguished Dissertation Award Committee 2020, has selected the following three theses as recipients of the EATCS Distinguished Dissertation Award for 2020:

Counting, Sampling and Testing Subgraphs in Sublinear-Time, by **Talya Eden,** (talyaa01@gmail.com), Tel Aviv University. Advisor: Dana Ron (danaron@tauex.tau.ac.il),

Expressivity of first-order logic, star-free propositional dynamic logic and communicating automata, by **Marie Fortin,** (Marie.Fortin@liverpool.ac.uk), l'Université Paris-Saclay Advisors, Paul Gastin (gastin@lsv.fr) and Benedikt Bollig (bollig@lsv.fr),

Approximation Algorithms for Traveling Salesman Problems, by **Vera Traub,** (vera.traub@ifor.math.ethz.ch), University of Bonn Advisor: Jens Vygen (vygen@or.uni-bonn.de)

The EATCS Distinguished Dissertation Award Committee 2020 consisted of

- Susanne Albers
- Nikhil Bansal
- Elvira Mayordomo
- Jaroslav Nesetril
- Damian Niwinski
- David Peleg (chair)
- Vladimiro Sassone
- Alexandra Silva

The EATCS Distinguished Dissertation Award has been established to promote and recognize outstanding dissertations in the field of Theoretical Computer Science. Any PhD dissertation in the field of Theoretical Computer Science successfully defended in 2020 has been eligible. The dissertations were evaluated on the basis of originality and potential impact on their respective fields and on Theoretical Computer Science. Each of the selected dissertations will receive a prize of 1000 Euro. The award receiving dissertations will be published on the EATCS web site, where all the EATCS Distinguished Dissertations will be collected.

The three recipients will be presented during the ICALP 2021 conference.

The list of the previous recipients of the EATCS Distinguished Dissertation Award is available at https://eatcs.org/index.php/dissertation-award.

# EATCS-Fellows 2021

The EATCS has recognized six of its members for their outstanding contributions to theoretical computer science by naming them as recipients of an EATCS fellowship.

The EATCS Fellows for 2021 are:

**Luca Aceto,** Reykjavik University, Iceland, and Gran Sasso Science Institute, Italy: for his fundamental contributions to concurrency theory, and outstanding merits for the community of theoretical computer science, in particular as an inspiring president of EATCS.

**Rajeev Alur,** University of Pennsylvania, USA: for his fundamental contributions to the theory of verification, especially of timed and hybrid, concurrent and multi-agent, and hierarchical and recursive systems.

**Samir Khuller,** Northwestern University, USA: for his fundamental contributions to combinatorial approximation algorithms – specifically for work in graph algorithms and scheduling, and for mentoring and building community.

**David Peleg,** Weizmann Institute of Science, Israel: for his fundamental contributions to the areas of distributed graph algorithms, wireless networks, robotics and social networks, and his longstanding support for the development of theoretical computer science in Europe.

**Davide Sangiorgi,** University of Bologna, Italy : for his fundamental contributions to concurrency and the foundations of programming languages, contributing notably to the $\pi$-calculus and to coinduction-based proofs.

**Saket Saurabh,** The Institute of Mathematical Sciences, Chennai, India: for his fundamental contributions to algorithms, including parameterized algorithms and kernelization.

The aforementioned members of the EATCS were selected by the EATCS Fellow Selection Committee, after examining the nominations received from our research community.

The EATCS Fellow Selection Committee consisted of

- Christel Baier
- Mikołaj Bojanczyk (chair)
- Mariangiola Dezani
- Josep Diaz
- Giuseppe F. Italiano

The EATCS Fellows Program was established by the association in 2014 to recognize outstanding EATCS members for their scientific achievements in the field of Theoretical Computer Science. The Fellow status is conferred by the EATCS Fellows-Selection Committee upon a person having a track record of intellectual and organizational leadership within the EATCS community. Fellows are expected to be "model citizens" of the TCS community, helping to develop the standing of TCS beyond the frontiers of the community.

The EATCS is very proud to have the above-mentioned members of the association among its fellows.

The list of EATCS Fellows is available at http://www.eatcs.org/index.php/eatcs-fellows.

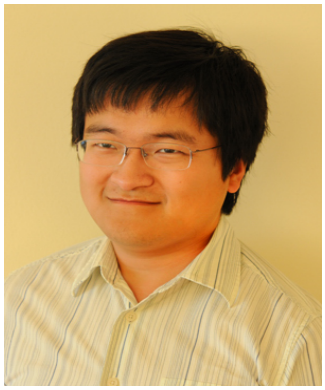
# Gödel Prize 2021

The Gödel Prize for outstanding papers in the area of theoretical computer science is sponsored jointly by the EATCS and the ACM SIGACT. The Prize is named in honor of Kurt Gödel in recognition of his major contributions to mathematical logic and of his interest, discovered in a letter he wrote to John von Neumann shortly before Neumann's death, in what has become the famous "P versus NP" question. The Prize includes an award of $5000 (US). This award is presented annually, with the presentation taking place alternately at the International Colloquium on Automata, Languages, and Programming (ICALP) and ACM Symposium on the Theory of Computing (STOC); it will be presented at STOC this year.

The 2021 Gödel Prize is jointly awarded to the following three papers

- Andrei Bulatov: The Complexity of the Counting Constraint Satisfaction Problem. J. ACM 60(5): 34:1-34:41 (2013).


• Martin E. Dyer and David Richerby: An Effective Dichotomy for the Counting Constraint Satisfaction Problem. SIAM J. Computing. 42(3): 1245-1274 (2013).

• Jin-Yi Cai and Xi Chen: Complexity of Counting CSP with Complex Weights J. ACM 64(3): 19:1– 19:39 (2017).

Constraint satisfaction is a subject of central significance in computer science, since a very large number of combinatorial problems, starting from Boolean Satisfiability and Graph Coloring, can be phrased as constraint satisfaction problems (CSP). The papers above, taken together, are the culmination of a large body of work on the classification of counting complexity of CSPs and prove an all-encompassing Complexity Dichotomy Theorem for counting CSP-type problems that are expressible as a partition function.

The class of problems that the final form of this dichotomy classifies is exceedingly broad. It includes all counting CSPs, all types of graph homomorphisms

(undirected or directed, unweighted or weighted), and spin systems (and thus a large variety of problems from statistical physics). Examples include counting vertex covers, independent sets, antichains, graph colorings, the Ising model, the Potts model, the q-particle Widom-Rowlinson model, the q-type Beach model, and more. For all these problems this theorem gives a complexity dichotomy classification: Every problem in the class is either solvable in polynomial time or is #P-hard.

Award Committee:

- Samson Abramsky (University of Oxford)
- Nikhil Bansal (CWI Amsterdam)
- Robert Krauthgamer (Weizmann Institute)
- Ronitt Rubinfeld (Massachusetts Institute of Technology)
- Daniel Spielman, Chair (Yale University)
- David Zuckerman (University of Texas at Austin)

The list of the previous recipients of the Gödel Prize is available at https://eatcs.org/index.php/goedel-prize.

# Ackermann Award 2021

## The EACSL Outstanding Dissertation Award for Logic in Computer Science 2021

## Call for Nominations

**Deadline: 1 July 2021**

Nominations are now invited for the 2021 Ackermann Award. PhD dissertations in topics specified by the CSL and LICS conferences, which were formally accepted as PhD theses at a university or equivalent institution between 1 January 2019 and 31 December 2020 are eligible for nomination for the award. The deadline for submission is 1 July 2021. Submission details follow below.

The 2021 Ackermann Award will be presented to the recipient(s) at CSL 2022, the annual conference of the EACSL.

The award consists of

- a certificate,
- an invitation to present the thesis at the CSL conference,
- the publication of the laudatio in the CSL proceedings,
- an invitation to the winner to publish the thesis in the FoLLI subseries of Springer LNCS, and
- financial support to attend the conference.

The jury consists of:

- Christel Baier (TU Dresden);
- Michael Benedikt (Oxford University);

- Mikolaj Bojanczyk (University of Warsaw);
- Jean Goubault-Larrecq (ENS Paris-Saclay);
- Prakash Panangaden (McGill University);
- Simona Ronchi Della Rocca (University of Torino), the vice-president of EACSL;
- Thomas Schwentick (TU Dortmund) , the president of EACSL;
- Alexandra Silva, (University College London), ACM SigLog representative.

The jury is entitled to give the award to more (or less) than one dissertation in a year.

The candidate or his/her supervisor should submit

1. the thesis (ps or pdf file);
2. a detailed description (not longer than 20 pages) of the thesis in ENGLISH (ps or pdf file); it is recommended to not squeeze as much material as possible into these 20 pages, but rather to use them for a gentle introduction and overview, stressing the novel results obtained in the thesis and their impact;
3. a supporting letter by the PhD advisor and two supporting letters by other senior researchers (in English); supporting letters can also be sent directly to Thomas Schwentick (thomas.schwentick@tu-dortmund.de);
4. a short CV of the candidate;
5. a copy of the document asserting that the thesis was accepted as a PhD thesis at a recognized University (or equivalent institution) and that the candidate has received his/her PhD within the specified period.

The submission should be sent by e-mail as attachments to the chair of the jury, Thomas Schwentick: thomas.schwentick@tu-dortmund.de

The e-mail should have the subject line *Ackermann Award 21 Submission* and as text the name of the candidate and the list of attachments. Submissions can be sent via several e-mail messages. If this is the case, please indicate it in the text.

# Institutional Sponsors

**CTI, Computer Technology Institute & Press "Diophantus"**
Patras, Greece

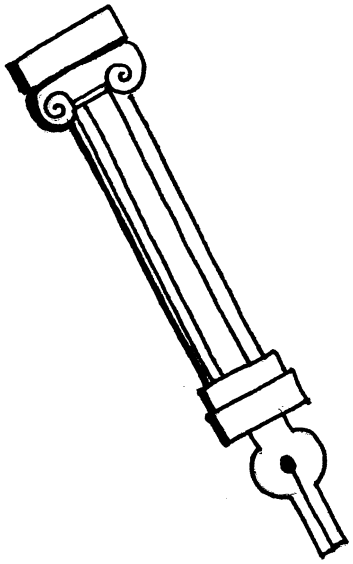
**CWI, Centum Wiskunde & Informatica**
Amsterdam, The Netherlands

**MADALGO, Center for Massive Data Algorithmics**
Aarhus, Denmark

**Microsoft Research Cambridge**
Cambridge, United Kingdom

**Springer-Verlag**
Heidelberg, Germany

# EATCS
# Columns



# THE LOGIC IN COMPUTER SCIENCE COLUMN

BY

**YURI GUREVICH**

Computer Science and Engineering
University of Michigan, Ann Arbor, MI 48109, USA
gurevich@umich.edu

*BEATCS no 134*

# REVERSIFY ANY SEQUENTIAL ALGORITHM

Yuri Gurevich

**Abstract**

To reversify an arbitrary sequential algorithm $A$, we gently instrument $A$ with bookkeeping machinery. The result is a step-for-step reversible algorithm that mimics $A$ step-for-step and stops exactly when $A$ does.

Without loss of generality, we presume that algorithm $A$ is presented as an abstract state machine that is behaviorally identical to $A$. The existence of such representation has been proven theoretically, and the practicality of such representation has been amply demonstrated.

*Darn the wheel of the world! Why must it*
*continually turn over? Where is the reverse gear?*
*— Jack London*

## 1 Introduction

In 1973, Charles Bennett posited that an"irreversible computer can always be made reversible" [2, p. 525]. To this end, he showed how to transform any one-tape Turing machine $M$ that computes a function $F(x)$, into a reversible three-tape Turing machine $M^R$ computing the function $x \mapsto (x, F(x))$. First, $M^R$ emulates the computation of $M$ on $x$, saving enough information to ensure step-for-step reversibility. If and when the output is computed, the emulation phase ends, and $M^R$ proceeds to erase all saved information with the exception of the input.

Bennett's construction shows that, in principle, every sequential algorithm, is reversifiable[1]. In practice, you don't want to compile your algorithms to a one-tape Turing machine $M$ and then execute the three-tape Turing machine $M^R$.

[1] We attempt to give a new useful meaning to the word reversify. To reversify an algorithm means to transform it into a reversible form (rather than to formulate it anew in verse, which is the current dictionary meaning of reversify).

It had been discussed in the programming community, in particular by Edsger Dijkstra [8, pp 351–354] and David Gries [10, pp 265–274], which programs are reversible, but Bennett's reversification idea was either unknown to or neglected by programming experts.

The progress was led by physicists. They reversified Boolean circuits and other computation models. "We have shown", wrote Edward Fredkin and Tommaso Toffoli [9, p 252], "that abstract systems having universal computing capabilities can be constructed from simple primitives which are invertible". The interest in reversible computations and especially in reversible circuit computations soared with the advent of quantum computing. This is related to the fact that pure (involving no measurements) quantum computations are reversible. There are books on reversible computations [1, 6, 17, 18]. The International Conference on Reversible Computation will have its 13th meeting in 2021 [19]

In this paper, we use sequential abstract state machines, in short sequential ASMs, to address the problem of practical reversification of arbitrary sequential algorithms. Why ASMs? Let us explain.

ASMs were introduced to faithfully simulate arbitrary algorithms on their natural abstraction levels [11]. One instructive early result was the formalization of the C programming language [14].

In [12], an ambitious ASM thesis was formulated: For every algorithm $A$, there is an ASM $B$ that is behaviorally equivalent to $A$. If $A$ is sequential, then $B$ has the same initial states as $A$ and the same state transition function. In [13], we axiomatized sequential algorithms and proved the ASM thesis for them[2]. Thus, semantically, sequential algorithms are sequential ASMs. In the meantime, substantial evidence has been accumulated to support the practicality of faithful ASM modeling. Some of it is found in the 2003 book [3].

The main result of the present paper is a simple construction, for every sequential ASM $A$, of a reversible sequential ASM $B$ that step-for-step simulates $A$ and stops when $A$ does. $B$ does exactly what $A$ does plus some bookkeeping. If $A$ uses some input and output variables and computes some function, then $B$ uses the same input and output variables and computes the same function.

[2]Later these results were generalized to other species of algorithms, e.g. to synchronous parallel algorithms [4] and interactive algorithms [5].

# 2 Preliminaries

The purpose of this section is to make the current paper self-contained.

## 2.1 Sequential algorithms

By sequential algorithms we mean algorithms as the term was understood before modern computer science generalized the notion of algorithm in various directions, which happened in the final decades of the 20th century. In this connection, sequential algorithms are also called classical.

While the term “sequential algorithm" is short and convenient, it also is too laconic. Some explication is in order. “Algorithms,” said Andrei Kolmogorov in a 1953 talk [16], “compute in steps of bounded complexity.” Let’s look more closely at the two aspects mentioned by Kolmogorov. One aspect is computing in steps, one step after another. Kolmogorov didn’t say “one step after another.” He didn’t have to. That was understood at the time.

The other aspect is a somewhat vague constraint: the bounded complexity of any one step of the algorithm. We prefer a related constraint, arguably a version of Kolmogorov’s constraint: the bounded resources of any one step of the algorithm. The bounded resources constraint, still informal, seems to us clearer and more suitable. It might have been Kolmogorov’s intention all along. We do not know exactly what Kolmogorov said during that talk[3].

To summarize, sequential algorithms can be characterized informally as transition systems that compute in bounded-resources steps, one step after another.

In our axiomatization of sequential algorithms [13], the bounded resources constraint gives rise to the crucial bounded-exploration axiom. It is also used to justify that a sequential algorithm doesn’t hang forever within a step; time is a bounded resource.

In the following subsections, we recall some basic notions of mathematical logic in the form appropriate to our purposes.

[3] Vladimir Uspensky, who chaired the Logic Department of Moscow State University after Kolmogorov’s death, admitted to me that the abstract [16] of Kolmogorov’s talk for the Moscow Mathematical Society was written by him (Uspensky) after many unsuccessful attempts to squeeze an abstract from Kolmogorov.

## 2.2 Vocabularies

A *vocabulary* is a finite collection of function symbols where each symbol $f$ is endowed with some metadata according to the following clauses (V1)–(V4). We interleave the four clauses with auxiliary definitions and explanations.

(V1) Each symbol $f$ is assigned a natural number, the *arity* of $f$.

Define *terms* (or *expressions*) by induction. If $f$ is an $r$-ary symbol in and $t_1, \ldots, t_r$ are terms, then $f(t_1, \ldots, t_r)$ is a term. (The case $r = 0$ is the basis of induction.)

(V2) Some symbols $f$ are marked as *relational*.

Clauses (V1) and (V2) are standard in logic, except that, traditionally, relations are viewed as separate category, not as a special functions.

(V3) $f$ may be marked as *dynamic*; if not then $f$ is called *static*. Nullary static symbols are called *constants*; nullary dynamic symbols are called *variables*.

Clause (V3) is related to our use of structures as states of algorithms. The intention is that, during computation, only dynamic functions may be assigned new values. We say that a term is *static* if it involves only static functions.

We presume that every vocabulary contains the following *obligatory* symbols which are all static.

- Constants $\top$ and $\bot$ (read "true" and "false"), unary `Bool`, and the standard propositional connectives. All these symbols are relational.
- Constant 0, and unary `Num`, `increment`, and `decrement`. Of these four symbols, only `Num` is relational.
- Constant `nil` (called `undef` in [13] and other early papers) and the (binary) equality sign =. Of these two symbols, only the equality sign is relational.

(V4) Every dynamic symbol $f$ is assigned a static term, the *default term* of $f$. If $f$ is relational then so is its default term.

Clauses (V2) and (V4) constitute rudimentary typing which is sufficient for our purposes in this paper. As a rule, the default term for any relational symbol is $\bot$. If a variable $v$ is supposed to take numerical values, then typically the default term for $v$ would be 0, but it could be 1. This concludes the definition of vocabularies.

If $\Upsilon$ and $\Upsilon'$ are vocabularies, we write $\Upsilon \subseteq \Upsilon'$, and we say that $\Upsilon$ is *included* in $\Upsilon'$ and that $\Upsilon'$ *includes* or *extends* $\Upsilon$, if every $\Upsilon$ symbol belongs to $\Upsilon'$ and has the same metadata in $\Upsilon'$.

## 2.3 Structures

A *structure* $X$ of vocabulary $\Upsilon$ is a nonempty set $|X|$, the *universe* or *base set* of $X$, together with interpretations of the function symbols in $\Upsilon$. The vocabulary $\Upsilon$ may be denoted $\mathrm{Voc}(X)$.

An $r$-ary function symbol $f$ is interpreted as a function $f : |X|^r \to |X|$ and is called a *basic function* of $X$. If $f$ is nullary then $f$ is just a name of an element of (the universe of) $X$. If $f$ is dynamic and $d$ is the default term for $f$, then the value (denoted by) $d$ is the *default value* of $f$.

If $f$ is relational, then the elements $\top$ are $\bot$ are the only possible values of $f$. If $f(\bar{x}) = \top$, we say that $f$ is true (or holds) at $\bar{x}$; otherwise we say that $f$ is false (or fails) at $\bar{x}$. If $f, g$ are relations of the same arity $r$, then $f, g$ are *equivalent* in $X$ if their values at every $r$-tuple of elements of $X$ are the same.

Any basic relation $f$ is the characteristic function of the set $\{x : f(x) = \top\}$. It is often convenient to treat $f$ as that set. We will do that in §5.

**Remark 2.1** (Names and denotations)**.** Syntactic objects often denote semantical objects. For example, vocabulary symbols denote basic functions. Different conventions may be used for disambiguation, e.g. a basic function may be denoted $f_X$. We will use no disambiguation convention in this paper. It should be clear from the context whether a symbol means a syntactic or semantic object. ◁

The equality sign has its usual meaning. `Bool` comprises (the values of) $\top$, $\bot$ which, together with the propositional connectives, form a two-element Boolean algebra.

Given a structure $X$, the *value* $\mathcal{V}_X(f(t_1,\ldots,t_r))$ of a $\mathrm{Voc}(X)$ term $f(t_1,\ldots,t_r)$ in $X$ is defined by induction:

$$\mathcal{V}_X(f(t_1,\ldots,t_r)) = f(\mathcal{V}_X(t_1),\ldots,\mathcal{V}_X(t_r)). \tag{1}$$

Again, the case $r = 0$ is the base of induction.

Instead of $\texttt{increment}(x)$, we will write $x + 1$ and $x - 1$. `Num` comprises the values of terms $0, 0 + 1, (0 + 1) + 1, \ldots$ which are all distinct. These values are denoted $0, 1, 2, \ldots$ respectively; we call them the *natural numbers* of structure $X$, and we say that these values are *numerical*. `decrement` is interpreted as expected as well. Instead of $\texttt{decrement}(x)$, we write $x - 1$. $\texttt{decrement}(0) = \texttt{nil}$. The value of `nil` is neither Boolean nor numerical.

**Remark 2.2** (Totality)**.** In accordance with §2.1, all basic functions are total. In applications, various error values may arise, in particular *timeout*. But, for our purposes in this paper (as in [13]), an error value is just another value.

A *location* in a structure $X$ is a pair $\ell = (f, \bar{x})$ where $f$ is a dynamic symbol in $\mathrm{Voc}(X)$ of some arity $r$ and $\bar{x}$ is an $r$-tuple of elements of $X$. The value $f(\bar{x})$ is the *content* of location $\ell$.

An *update of location* $\ell = (f, \bar{x})$ is a pair $(\ell, y)$, also denoted $(\ell \leftharpoonup y)$, where $y$ an element of $X$; if $f$ is relational then $y$ is Boolean. To *execute* an update $(\ell \leftharpoonup y)$ in $X$, replace the current content $\mathcal{V}_X(f(\bar{x}))$ of $\ell$ with $y$, i.e., set $f_X(\bar{x})$ to $y$. An update $(\ell \leftharpoonup y)$ of location $\ell = (f, \bar{x})$ is *trivial* if $y = f(\bar{x})$.

An *update of structure* $X$ is an update of any location in $X$. A set $\Delta$ of updates of $X$ is *contradictory* if it contains updates $(\ell \leftharpoonup y_1)$ and $(\ell \leftharpoonup y_2)$ with distinct $y_1, y_2$; otherwise $\Delta$ is *consistent*.

## 2.4 Sequential abstract state machines

Fix a vocabulary $\Upsilon$ and restrict attention to function symbols in $\Upsilon$ and terms over $\Upsilon$.

**Definition 2.3** (Syntax of rules)**.** *Rules* over vocabulary $\Upsilon$ are defined by induction.

1. An *assignment rule* or simply *assignment* has the form

$$f(t_1, \ldots, t_r) := t_0 \tag{2}$$

   where $f$, the *head* of the assignment, is dynamic, $r = \text{Arity}(f)$, and $t_0, \ldots, t_r$ are terms. If $f$ is relational, then the head function of $t_0$ is relational. The assignment (2) may be called an $f$ assignment.

2. A *conditional rule* has the form

$$\texttt{if } \beta \texttt{ then } R_1 \texttt{ else } R_2 \tag{3}$$

   where $\beta$ is a Boolean-valued term and $R_1, R_2$ are $\Upsilon$ rules.

3. A *parallel rule* has the form

$$R_1 \quad \| \quad R_2 \quad \| \quad \cdots \quad \| \quad R_k \tag{4}$$

   where $k$ is a natural number and $R_1, \ldots, R_k$ are $\Upsilon$ rules. In case $k = 0$, we write `Skip`. ◁

**Definition 2.4** (Semantics of rules)**.** Fix an $\Upsilon$ structure $X$. Every $\Upsilon$ rule $R$ generates a finite set $\Delta$ of updates in $X$. $R$ *fails* in $X$ if $\Delta$ is contradictory; otherwise $R$ *succeeds* in $X$. To *fire* (or *execute*) rule $R$ that succeeds in structure $X$ means to execute all $\Delta$ updates in $X$.

1. An assignment $f(t_1, \ldots, t_r) := t_0$ generates a single update $(\ell, \mathcal{V}_X(t_0))$ where $\ell = (f, (\mathcal{V}_X(t_1), \ldots, \mathcal{V}_X(t_r)))$.

2. A conditional rule $\texttt{if } \beta \texttt{ then } R_1 \texttt{ else } R_2$ works exactly as $R_1$, if $\beta = \top$ in $X$, and exactly as $R_2$ otherwise.

3. A parallel rule $R_1 \quad \| \quad R_2 \quad \| \quad \cdots \quad \| \quad R_k$ generates the union of the update sets generated by rules $R_1, \ldots, R_k$ in $X$. ◁

**Definition 2.5.** A *sequential ASM A* is given by the following three components.

1. A vocabulary $\Upsilon$, denoted Voc($A$).

2. A nonempty collection of Voc($A$) structures, closed under isomorphisms. These are the *initial states* of $A$. ◁

3. A Voc($A$) rule, called the *program* of $A$ and denoted Prog($A$).

As we mentioned in §1, every sequential algorithm $A$ is behaviorally identical to some sequential ASM $B$; they have the same initial states and the same state-transition function.

In the rest of the paper, by default, all ASMs are sequential.

Consider an ASM $A$. A Voc($A$) structure $X$ is *terminal* for $A$ if Prog($A$) produces no updates (not even trivial updates[4]) in $X$. A *partial computation* of $A$ is a finite sequence $X_0, X_1, \ldots, X_n$ of $\Upsilon$ structures where

- $X_0$ is an initial state of $A$,
- every $X_{i+1}$ is obtained by executing Prog($A$) in $X_i$, and
- no structure in the sequence, with a possible exception of $X_n$, is terminal.

If $X_n$ is terminal, then the partial computation is *terminating*. A *(reachable) state* of $A$ is an Voc($A$) structure that occurs in some partial computation of $A$.

A Boolean expression $\gamma$ is a *green light* for an ASM $A$ if it holds in the non-terminal states of $A$ and fails in the terminal states.

**Lemma 2.6.** *Every ASM has a green light.*

*Proof.* By induction on rule $R$, we construct a green light $\gamma_R$ for any ASM with program $R$. If $R$ is an assignment, set $\gamma_R = \top$. If $R$ is the parallel composition of rules $R_i$, set $\gamma_R = \bigvee_i \gamma_{R_i}$. If $R =$ `if` $\beta$ `then` $R_1$ `else` $R_2$, set $\gamma_R = (\beta \wedge \gamma_{R_1}) \vee (\neg\beta \wedge \gamma_{R_2})$. □

In examples and applications, typically, such conditions are easily available. Think of terminal states of finite automata or of halting control states of Turing machines. In a while-loop program, the while condition is a green light.

[4] In applications, trivial updates of $A$ may mean something for its environment.

# 3 Reducts and expansions

In mathematical logic, a structure $X$ is a *reduct* of a structure $Y$ if $\mathrm{Voc}(X) \subseteq \mathrm{Voc}(Y)$, the two structures have the same universe, and every $\mathrm{Voc}(X)$ symbol $f$ has the same interpretations in $X$ and in $Y$. If $X$ is a reduct of $Y$, then $Y$ is an *expansion* of $X$. For example, the field of real numbers expands the additive group of real numbers.

We say that an expansion $Y$ of a structure $X$ is *uninformative* if the additional basic functions of $Y$ (which are not basic functions of $X$) are dynamic and take only their default values in $Y$. (The default values are defined in §2.3.) Clearly, $X$ has a unique uninformative expansion to $\mathrm{Voc}(B)$.

**Definition 3.1.** An ASM $B$ is a *faithful expansion* of an ASM $A$ if the following conditions hold.

(E1) $\mathrm{Voc}(A) \subseteq \mathrm{Voc}(B)$. The symbols in $\mathrm{Voc}(A)$ are the *principal* symbols of $\mathrm{Voc}(B)$, and their interpretations in $\mathrm{Voc}(B)$ structures are *principal* basic functions; the other $\mathrm{Voc}(B)$ symbols and their interpretations are *ancillary*.

(E2) All ancillary symbols are dynamic, and the initial states of $B$ are the uninformative expansions of the initial states of $A$.

(E3) If $Y$ is a $\mathrm{Voc}(B)$ structure and $X$ the $\mathrm{Voc}(A)$ reduct of $Y$, then the principal-function updates (including trivial updates) generated by $\mathrm{Prog}(B)$ in $Y$ coincide with the those generated by $\mathrm{Prog}(A)$ in $X$, and the ancillary-function updates generated by $\mathrm{Prog}(B)$ in $Y$ are consistent. ◁

**Corollary 3.2.** *Suppose that $B$ is a faithful expansion of an ASM $A$, then the following claims hold.*

1. *If $X_0, \ldots, X_n$ is a partial computation of $A$ then there is a unique partial computation $Y_0, \ldots, Y_n$ of $B$ such that every $X_i$ is the* $\mathrm{Voc}(A)$ *reduct of the corresponding $Y_i$.*

2. *If states $Y_0, \ldots, Y_n$ of $B$ form a partial computation of $B$, then their* $\mathrm{Voc}(A)$ *reducts form a partial computation $X_0, \ldots, X_n$ of $A$.*

3. *The* $\mathrm{Voc}(A)$ *reduct of a state of $B$ is a state of $A$.*

If an ASM $A$ computes a function $F$, one would expect that any faithful expansion of $A$ computes function $F$ as well. To confirm this expectation, we need to formalize what it means to compute a function. In the context of this paper, every ASM state is endowed with a special copy of the set $\mathbb{N}$ of natural numbers. This makes the desired formalization particularly easy for numerical partial functions $F : \mathbb{N}^k \to \mathbb{N}$.

**Corollary 3.3.** *Suppose that an ASM $A$ computes a partial numerical function $F : \mathbb{N}^k \to \mathbb{N}$ in the following sense:*

1. *$A$ has input variables $\iota_1, \ldots, \iota_n$ taking numerical values in the initial states, and $A$ has an output variable $o$,*
2. *all initial states of $A$ are isomorphic except for the values of the input variables, and*
3. *the computation of $A$ with initial state $X$ eventually terminates if and only if $F$ is defined at tuple $\bar{x} = (\mathcal{V}_X(\iota_1), \ldots, \mathcal{V}_X(\iota_n))$, in which case the final value of $o$ is $F(\bar{x})$.*

*Then every faithful expansion of $A$ computes $F$ in the same sense.* ◁

Corollary 3.3 can be generalized to computing more general functions and to performing other tasks, but this is beyond the scope of this paper.

An ASM may be faithfully expanded by instrumenting its program for monitoring purposes. For example, if you are interested how often a particular assignment $\sigma$ fires, replace $\sigma$ with a parallel composition

$$\sigma \quad \| \quad \kappa := \kappa + 1$$

where a fresh variable $\kappa$, initially zero, is used as a counter. A similar counter is used in our Reversibility Theorem below.

# 4 Reversibility

**Definition 4.1.** An ASM $B$ is *reversible (as is)* if there is an ASM $C$ which reverses all $B$'s computations in the following sense. If $Y_0, Y_1, \ldots, Y_n$ is a partial computation of $B$, then $Y_n, Y_{n-1}, \ldots, Y_0$ is a terminating computation of $C$.

**Theorem 4.2** (Reversification Theorem)**.** *Every ASM A has a faithful reversible expansion.*

*Proof.* Enumerate the (occurrences of the) assignments in Prog($A$) in the order they occur:

$$\sigma_1, \sigma_2, \ldots, \sigma_N$$

It is possible that $\sigma_i, \sigma_j$ are identical even though $i \neq j$. The metavariable $n$ will range over numbers $1, 2, \ldots, N$. For each $n$, let $f^n$ be the head of $\sigma_n$, $r_n$ = Arity($f^n$), and $t^n_0, t^n_1, \ldots, t^n_{r_n}$ the terms such that

$$\sigma_n = \left(f^n(t^n_1, \ldots, t^n_{r_n}) := t^n_0\right).$$

We construct an expansion $B$ of $A$. The ancillary symbols of $B$ are as follows.

1. A variable $\kappa$.
2. For every $n$, a unary relation symbol $\texttt{Fire}^n$.
3. For every $n$, unary function symbols $f^n_0, f^n_1, \ldots, f^n_{r_n}$.

The default term for $\kappa$ is 0. The default term for all relations $\texttt{Fire}^n$ is $\bot$. The default term for all functions $f^n_0, f^n_1, \ldots, f^n_{r_n}$ is $\texttt{nil}$. Accordingly, the initial states of $B$ are obtained from the initial states of $A$ by setting $\kappa = 0$, every $\texttt{Fire}^n(x) = \bot$, and every $f^n_i(x) = \texttt{nil}$,

The intention is this. If $X_0, X_1, \ldots, X_l$ is a partial computation of $B$, then for each $k = 0, \ldots, l$ we have the following.

1. The value of $\kappa$ in $X_k$ is $k$, so that $\kappa$ counts the number of steps performed until now; we call it a *step counter*.
2. $\texttt{Fire}^n(\kappa)$ holds in $X_{k+1}$ if and only if $\sigma_n$ fires in $X_k$.
3. The values of $f^n_1(\kappa), \ldots, f^n_{r_n}(\kappa)$ in $X_{k+1}$ record the values of the terms $t^n_1, \ldots, t^n_{r_n}$ in $X_k$ respectively, and the value of $f^n_0(\kappa)$ in $X_{k+1}$ records the value of the term $f^n(t^n_1, \ldots, t^n_{r_n})$ in $X_k$.

The program of $B$ is obtained from Prog($A$) by replacing every assignment $\sigma_n$ with Instr($n$) (an allusion to "instrumentation") where

Instr($n$) =

$\sigma_n \quad \| \quad \kappa := \kappa + 1 \quad \| \quad \mathtt{Fire}^n(\kappa + 1) := \top \quad \|$
$f_0^n(\kappa + 1) := f^n(t_1^n, \ldots, t_{r_n}^n) \quad \|$
$f_1^n(\kappa + 1) := t_1^n \quad \| \quad \ldots \quad \| \quad f_{r_n}^n(\kappa + 1) := t_{r_n}^n$

It is easy to check that the conditions (E1)–(E3) of Definition 3.1 hold, and $B$ is indeed a faithful expansion of $A$. In particular, if $Y$ and $X$ are as in (E3) and $X$ is terminal, then no assignment $\sigma_n$ fires in $X$, and therefore no Instr($n$) fires in $Y$, so that $Y$ is terminal as well.

**Lemma 4.3.** *If $Y_0, \ldots, Y_k$ is a partial computation of $B$, then*

- *$\kappa = k$ in $Y_k$ and*
- *if $j > k$ then $\mathtt{Fire}^n(j), f_0^n(j), \ldots, f_{r_n}^n(j)$ have their default values in $Y_k$.*

*Proof of lemma.* Induction on $k$. □

Now, we will construct an ASM $C$ which reverses $B$'s computations. The vocabulary of $C$ is that of $B$, and any Voc($C$) structure is an initial state of $C$. The program of $C$ is

$$\mathtt{if}\ \kappa > 0\ \mathtt{then}\ \Big(\kappa := \kappa - 1 \quad \| \quad \mathop{\mathbf{PAR}}_{n} \mathrm{Undo}(n)\Big)$$

where **PAR** is parallel composition, $n$ ranges over $\{1, 2, \ldots, N\}$, and

Undo($n$) =

$\mathtt{if}\ \mathtt{Fire}^n(\kappa) = \top\ \mathtt{then}$
$\quad \mathtt{Fire}^n(\kappa) := \bot \quad \|$
$\quad f^n\Big(f_1^n(\kappa), \ldots, f_{r_n}^n(\kappa)\Big) := f_0^n(\kappa) \quad \| \quad f_0^n(\kappa) := \mathtt{nil} \quad \|$
$\quad f_1^n(\kappa) := \mathtt{nil} \quad \| \quad \ldots \quad \| \quad f_{r_n}^n(\kappa) := \mathtt{nil}$

**Lemma 4.4.** *Let $Y$ be an arbitrary nonterminal* Voc($B$) *structure such that all functions $\mathtt{Fire}^n$ and $f_i^n$ have their default values at argument $k = \mathcal{V}_Y(\kappa)$ in $Y$. If* Prog($B$) *transforms $Y$ to $Y'$, then* Prog($C$) *transforms $Y'$ back to $Y$, i.e.,* Prog($C$) *undoes the updates generated by* Prog($B$) *and does nothing else.*

*Proof of lemma.* The updates generated by Prog($B$) in $Y$ are the updates generated by the rules Instr($n$) such that $\sigma_n$ fires in $Y$. Since $k = \mathcal{V}_Y(\kappa)$, we have $\mathcal{V}_{Y'}(\kappa) = k + 1 > 0$, and therefore Prog($C$) decrements $\kappa$. It also undoes the other updates generated by the rules Instr($n$). Indeed, suppose that $\sigma_n$ fires in $Y$.

To undo the update $\mathtt{Fire}^n(k+1) \leftharpoonup \top$, Prog($C$) sets $\mathtt{Fire}^n(k+1)$ back to $\bot$.

To undo the update $f^n(t_1^n, \ldots, t_{r_n}^n) \leftharpoonup t_0^n$, generated by $\sigma_n$ itself, Prog($C$) sets $f^n\big(f_1^n(k+1), \ldots, f_{r_n}^n(k+1)\big)$ to $f_0^n(k+1)$. Recall that $f_1^n(k+1), \ldots, f_{r_n}^n(k+1)$ record $t_1^n, \ldots, t_{r_n}^n$ in $Y$ and $f_0^n(k+1)$ records the value of $f^n(t_1^n, \ldots, t_{r_n}^n)$ in $Y$. Thus, Prog($C$) sets $f^n(t_1^n, \ldots, t_{r_n}^n)$ back to its value in $Y$.

To undo the updates of $f_0^n(k+1), f_1^n(k+1), \ldots, f_{r_n}^n(k+1)$, Prog($C$) sets $f_0^n(k+1), f_1^n(k+1), \ldots, f_{r_n}^n(k+1)$ back to $\mathtt{nil}$.

Thus, being executed in $Y'$, Prog($C$) undoes all updates generated by the rules Instr($n$) in $Y$. A simple inspection of Prog($C$) shows that it does nothing else. Thus, Prog($C$) transforms $Y'$ to $Y$. □

Now suppose that $Y_0, \ldots, Y_n$ is a computation of $B$, $k < n$, $Y = Y_k$, and $Y' = Y_{k+1}$. Then $Y$ is nonterminal and, by Lemma 4.3, all $\mathtt{Fire}^n(\kappa)$ and $f_i^n(\kappa)$ have their default values in $Y$. By Lemma 4.4, Prog($C$) transforms $Y_{k+1}$ to $Y_k$. The $Y_0$ is a terminal state of $C$. Thus, $C$ reverses all $B$'s computations. □

The proof of Reversification Theorem uses notation and the form of Prog($B$) which is convenient for the proof. In examples and applications, notation and Prog($B$) can be simplified.

**Remark 4.5** (Notation)**.** Let $\sigma_n$ be an assignment $(g(t_1^n, \ldots, t_r^n) := t_0^n)$ so that $f^n$ is $g$. If $\sigma_n$ is the only $g$ assignment in Prog($A$) or if every other $g$ assignment $\sigma_m$ in Prog($A$) is just another occurrence of $\sigma_n$, then the ancillary functions $f_i^n$ may be denoted $g_i$; no confusion arises. ◁

Recall that a green light for an ASM $A$ is a Boolean-valued expression that holds in the nonterminal states and fails in the terminal states.

**Remark 4.6** (Green light and step counter)**.** In Prog($B$), every Instr($n$) has an occurrence of the assignment $\kappa := \kappa + 1$. A green light for $A$ provides an efficient way to deal with this excess. Notice that $B$ increments the step counter exactly when the green light is on.

Case 1: Prog($A$) has the form $\texttt{if}\ \gamma\ \texttt{then}\ (k := k + 1 \parallel \Pi)$.

In this case, $\gamma$ is a green light for $A$, and $A$ has already a step counter, namely $k$. Without loss of generality, $k$ is the step counter $\kappa$ used by Prog($B$); if not, rename one of the two variables. Notice that the assignment $\sigma_1 = (k := k + 1)$ needs no instrumentation. There is no need to signal firings of $\sigma_1$ because $\sigma_1$ fires at every step. And, when a step is completed, we know the previous value of the step counter; there is no need to record it.

Let $\text{Instr}^-(\Pi)$ be the rule obtained from $\Pi$ by first replacing every assignment $\sigma_n$ with the rule Instr($n$) defined in the proof of the program, and then removing all occurrences of $k := k + 1$. Then the program

$$\texttt{if}\ \gamma\ \texttt{then}\ (k := k + 1 \parallel \text{Instr}^-(\Pi))$$

has only one occurrence of $k := k + 1$ and is equivalent to Prog($B$).

Case 2: Prog($A$) = $(\texttt{if}\ \gamma\ \texttt{then}\ \Pi)$ where $\gamma$ is a green light for $A$ and the step counter $\kappa$ of Prog($B$) does not occur in Prog($A$).

The modified program $\texttt{if}\ \gamma\ \texttt{then}\ (\kappa := \kappa + 1 \parallel \Pi)$,
where $\kappa$ is the step counter of $B$, is a faithful expansion of Prog($A$), and thus Case 2 reduces to Case 1.

Case 3 is the general case.

By Lemma 2.6, every ASM program has a green light. If $\gamma$ is a green light for $A$ and $\Pi$ = Prog($A$), then the program $\texttt{if}\ \gamma\ \texttt{then}\ \Pi$ is equivalent to Prog($A$), and thus Case 3 reduces to Case 2. ◁

The rules Instr($n$) and Undo($n$), described in the proof of the theorem, are the simplest in the case when $r_n = 0$. In such a case, $\sigma_n$ has the form $v := t$ where $v$ is a variable, so that $f^n = v$ and $t_0^n = t$. Then

Instr($n$) =

$$\sigma_n \ \parallel\ \kappa := \kappa + 1 \ \parallel\ \texttt{Fire}^n(\kappa + 1) := \top \ \parallel\ v_0(\kappa + 1) := v,$$

Undo($n$) =

$$\texttt{if Fire}^n(\kappa) = \top \texttt{ then}$$
$$\texttt{Fire}^n := \bot \ \parallel\ v := v_0(\kappa) \ \parallel\ v_0(\kappa) := \texttt{nil}.$$

**Lemma 4.7.** *Suppose that an assignment $\sigma_n$ to a variable $v$ can fire only at the last step of A and that the update generated by $\sigma_n$ is never trivial. Then,* Instr($n$) *and* Undo($n$) *can be simplified to*

$$\begin{aligned} \mathrm{Instr}(n) &= && \sigma_n \quad \| \quad \kappa := \kappa + 1 \\ \mathrm{Undo}(n) &= && \texttt{if } v \neq d \texttt{ then } v := d \end{aligned}$$

*where $d$ is the default term for $v$ in* Voc($A$).

We do not assume that $\sigma_n$ fires at the last step of every computation of $A$, and so the expression $v \neq d$ is not necessarily a green light for $A$.

*Proof.* Suppose that $\sigma_n$ fires in state $Y$ of $B$. Then $Y$ is nonterminal, $v = d$ in $Y$, the next state $Y'$ is terminal, and $v \neq d$ in $Y'$. It is easy to see the simplified version of Undo($n$) indeed undoes the updates generated by the simplified version of Instr($n$) in $Y$. □

# 5 Examples

To illustrate the reversification procedure of §4, we consider three simple examples. By the reversification procedure we mean not only the constructions in the proof of Reversification Theorem, but also Remarks 4.5 and 4.6 and Lemma 4.7. Unsurprisingly, in each case, the faithful reversible expansion produced by the general-purpose procedure can be simplified.

## 5.1 Bisection algorithm

The well-known bisection algorithm solves the following problem where $\mathbb{R}$ is the field of real numbers. Given a continuous function $F : \mathbb{R} \to \mathbb{R}$ and reals $a, b, \varepsilon$ such that $F(a) < 0 < F(b)$ and $\varepsilon > 0$, find a real $c$ such that $|F(c)| < \varepsilon$. Here is a draft program for the algorithm:

```
if  |F((a + b)/2)| ≥ ε     then
    if      F((a + b)/2) < 0   then  a := (a + b)/2
    elseif  F((a + b)/2) > 0   then  b := (a + b)/2
elseif c = nil then  c := (a + b)/2
```

The condition $c = \texttt{nil}$ in the last line ensures that computation stops when $c$ is assigned a real number for the first time.

The Boolean expression $|F((a+b)/2)| \geq \varepsilon$ is not quite a green light for the algorithm. When it is violated for the first time, $c$ is still equal to `nil`. But the equality $c = \texttt{nil}$ is a green light. With an eye on using Remark 4.6, we modify the draft program to the following program, our "official" program of an ASM $A$ representing the bisection algorithm.

```
if  c = nil  then
    if       F((a+b)/2) < −ε    then  a := (a+b)/2
    elseif   F((a+b)/2) > ε     then  b := (a+b)/2
    else     c := (a+b)/2
```

Voc($A$) consists of the obligatory symbols, the symbols in Prog($A$), and the unary relation symbol `Real`. In every initial state of $A$, `Real` is (a copy of) the set of real numbers, the static functions of Prog($A$) have their standard meaning, and $c = \texttt{nil}$.

Notice that Lemma 4.7 applies to Prog($A$) with $\sigma_n$ being $c := (a+b)/2$. Taking this into account, the reversification procedure of §4 gives us a reversible expansion $B$ of $A$ with the following program.

```
if c = nil then
    κ := κ + 1   ||
    if F((a+b)/2) < −ε then
        a := (a+b)/2   ||  Fire¹(κ+1) := ⊤   ||  a₀(κ+1) := a
    elseif F((a+b)/2) > ε then
        b := (a+b)/2   ||  Fire²(κ+1) := ⊤   ||  b₀(κ+1) := b
    else c := (a+b)/2
```

This program can be simplified (and remain reversible). Notice that

- if $\texttt{Fire}^1(k+1) = \top$, then $\texttt{Fire}^2(k+1) = \bot$, the previous value of $b$ is the current value of $b$, and the previous value of $a$ is $2a - b$ where $a, b$ are the current values; and
- if $\texttt{Fire}^1(k+1) = \bot$, then $\texttt{Fire}^2(k+1) = \top$, the previous value of $a$ is the current value of $a$, and the previous value of $b$ is $2b - a$ where $a, b$ are the current values.

Thus, there is no need for functions $a_0, b_0$, recording the previous values of variables $a, b$, and there is no need for $\texttt{Fire}^2$. We get:

```
if c = nil then
    κ := κ + 1   ||
    if F((a + b)/2) < −ε then
        a := (a + b)/2   ||   Fire¹(κ + 1) := ⊤
    elseif F((a + b)/2) > ε then b := (a + b)/2
    else c := (a + b)/2
```

The corresponding inverse algorithm may have this program:

```
if κ > 0 then
  κ := κ − 1   ||   if c ≠ nil then c := nil
  || if Fire¹(κ) = ⊤ then (Fire¹(κ) := ⊥ || a := 2a − b)
    else b := 2b − a
```

## 5.2 Linear-time sorting

The information needed to reverse each step of the bisection algorithm is rather obvious; you don't have to use our reversification procedure for that. Such information is slightly less obvious in the case of the following sorting algorithm.

For any natural number $n$, the algorithm sorts an arbitrary array $f$ of distinct natural numbers $< n$ in time $\leq 2n$. Let $m$ be the length of an input array $f$, so that $m \leq n$. The algorithm uses an auxiliary array $g$ of length $n$ which is initially composed of zeroes.

Here is a simple illustration of the sorting procedure where $n = 7$ and $f = \langle 3, 6, 0\rangle$. Traverse array $f$ setting entries $g[f[i]]$ of $g$ to 1 for each index $i$ of $f$, i.e., setting $g[3], g[6]$ and $g[0]$ to 1, so that $g$ becomes $\langle 1, 0, 0, 1, 0, 0, 1\rangle$. Each index $j$ of $g$ with $g[j] = 1$ is an entry of the input array $f$. Next, traverse array $g$ putting the indices $j$ with $g[j] = 1$ — in the order that they occur — back into array $f$, so that $f$ becomes $\langle 0, 3, 6\rangle$. Voila, $f$ has been sorted in $m + n$ steps.

We describe an ASM $A$ representing the sorting algorithm. Arrays will be viewed as functions on finite initial segments of natural numbers. The nonobligatory function symbols in $\mathrm{Voc}(A)$ are as follows.

0. Constants $m, n$ and variables $k, l$.

1. Unary dynamic symbols $f$ and $g$.

2. Binary static symbols $<, +, -$ where $<$ is relational.

In every initial state of $A$,

0. $m$ and $n$ are natural numbers such that $m \leq n$, and $k = l = 0$,

1. $f$, $g$ are arrays of lengths $m$, $n$ respectively, the entries of $f$ are distinct natural numbers $< n$, and all entries of $g$ are zero,

2. the arithmetical operations $+, -$ and relation $<$ work as expected on natural numbers.

In the following program of $A$, $k$ is the step counter, and $l$ indicates the current position in array $f$ to be filled in.

```
if k < m + n then
    k := k + 1   ||
    if k < m then g(f(k)) := 1
    elseif g(k − m) = 1 then (f(l) := k − m || l := l + 1)
```

The reversification procedure of §4 plus some simplifications described below give us a faithful reversible expansion $B$ of $A$ with a program

```
if k < m + n then
    k := k + 1   ||
    if k < m then (g(f(k)) := 1 || g_1(k + 1) := f(k))
    elseif g(k − m) = 1 then
        f(l) := k − m   ||   l := l + 1   ||   f_0(k + 1) := f(l)
```

We made some simplifications of Prog($B$) by discarding obviously unnecessary ancillary functions.

- It is unnecessary to record the firings of assignment $\sigma_2 = (g(f(k)) := 1)$ because, in the states of $B$, the condition $\texttt{Fire}^2(k) = \top$ is expressed by the inequality $k \leq m$.

- The ancillary function $g_0$ recording the previous values of $g$ is unnecessary because those values are all zeroes.

- The final two assignments in Prog($A$) fire simultaneously, so that one fire-recording function, say $\texttt{Fire}^3$, suffices. But even that one ancillary function is unnecessary because, in the states of $B$, the condition $\texttt{Fire}^3(k) = \top$ is expressed by $m < k \wedge g(k - m - 1) = 1$.

- The ancillary functions $f_1$ and $l_0$ recording the previous value of $l$ are unnecessary because we know that value, it is $l - 1$.

The desired inverse algorithm $C$ may be given by this program:

$\texttt{if } k > 0 \texttt{ then}$
$\quad k := k - 1$
$\quad \| \texttt{ if } k \leq m \texttt{ then } (g(g_1(k)) := 0 \| g_1(k) := \texttt{nil})$
$\quad \| \texttt{ if } m < k \texttt{ and } g(k - m - 1) = 1 \texttt{ then}$
$\qquad f(l) := f_0(k) \quad \| \quad l := l - 1 \quad \| \quad f_0(k) := \texttt{nil}$

Obviously, $A$ is not reversible as is; its final state doesn't have information for reconstructing the initial $f$. But do we need both remaining ancillary functions? Since $f_0$ is obliterated after the first $n$ steps of $C$, $f_0$ seems unlikely on its own to ensure reversibility. But it does. The reason is that, after the first $n$ steps of $C$, the original array $f$ is restored. Recall that $g_1(k)$ records the value $f(k - 1)$ of the original $f$ for each positive $k \leq m$, but we can discard $g_1$ and modify Prog($C$) to

$\texttt{if } k > 0 \texttt{ then } k := k - 1$
$\quad \| \texttt{ if } k \leq m \texttt{ then } g(f(k - 1)) := 0$
$\quad \| \texttt{ if } m < k \texttt{ and } g(k - m - 1) = 1 \texttt{ then}$
$\qquad f(l) := f_0(k) \| l := l - 1 \| f_0(k) := \texttt{nil}$

Alternatively, we can discard $f_0$ but keep $g_1$. Indeed, the purpose of the assignment $f(l) := f_0(k)$ in Prog($C$) is to restore $f(l)$ to its original value. But recall that every $f(l)$ is recorded as $g_1(l + 1)$. So we can modify Prog($C$) to

$\texttt{if } k > 0 \texttt{ then } k := k - 1$
$\quad \| \texttt{ if } k \leq m \texttt{ then } (g(g_1(k)) := 0 \| g_1(k) := \texttt{nil})$
$\quad \| \texttt{ if } m < k \texttt{ and } g(k - m - 1) = 1 \texttt{ then}$
$\qquad f(l) := g_1(l + 1) \quad \| \quad l := l - 1$

## 5.3 External functions and Karger's algorithm

Until now, for simplicity, we restricted attention to algorithms that are isolated in the sense that their computations are not influenced by the environment. Actually, the analysis of sequential algorithms generalizes naturally and easily to the case when the environment can influence the computation of an algorithm [13, §8]. To this end, so-called external functions are used.

Syntactically, the item (V3) in §2.2 should be refined to say that a function symbol $f$ may be dynamic, or static, or external. Semantically, external functions are treated as oracles[5] When an algorithm evaluates an external function $f$ at some input $\bar{x}$, it is the environment (and typically the operating system) that supplies the value $f(\bar{x})$. The value is well defined at any given step of the algorithm; if $f$ is called several times, during the same step, on the same input $\bar{x}$, the same value is given each time. But, at a different step, a different value $f(\bar{x})$ may be given.

To illustrate reversification involving an external function, we turn attention to Karger's algorithm [15]. In graph theory, a minimum cut of a graph is a cut (splitting the vertices into two disjoint subsets) that minimizes the number of edges crossing the cut. Using randomization, Karger's algorithm constructs a cut which is a minimum cut with a certain probability. That probability is small but only polynomially (in the number of vertices) small. Here we are not interested in the minimum cut problem, only in the algorithm itself.

**Terminology 5.1.** Let $G = (V, E)$ be a graph and consider a partition $P$ of the vertex set $V$ into disjoint subsets which we call *cells*; formally $P$ is the set of the cells. The *P-ends* of an edge $\{x, y\}$ are the cells containing the vertices $x$ and $y$. An edge is *inter-cell* (relative to $P$) if its $P$-ends are distinct. ◁

Now we describe a version of Karger's algorithm that we call KA. Given a finite connected graph $(V, E)$, KA works with partitions of the vertex set $V$, one partition at a time, and KA keeps track of the set Inter of the inter-cell edges. KA starts with the finest partition $P = \{\{v\} : v \in V\}$ and Inter $= E$. If the current partition $P$ has $> 2$ cells, then Inter is nonempty because the graph $(V, E)$ is connected. In this case, KA selects a random inter-cell edge $e$, merges the $P$-ends

[5] In that sense, our generalization is similar to the oracle generalization of Turing machines.

$p, q$ of $e$ into one cell, and removes from Inter the edges in $\{\{x, y\} : x \in p \wedge y \in q\}$. The result is a coarser partition and smaller Inter. When the current partition has at most two cells, the algorithm stops.

Next we describe an ASM $A$ representing KA. There are many ways to represent KA as an ASM. Thinking of the convenience of description rather than implementation of KA, we chose to be close to naive set theory. Let $U$ be a set that includes $V$ and all subsets of $V$ and all sets of subsets of $V$ (which is much more than needed but never mind). The relation $\in$ on $U$ has its standard meaning; the vertices are treated as atoms (or urelements), not sets.

The nonobligatory function symbols of Voc($A$) are as follows.

0. Nullary variables $P$ and Inter.

1. Unary static symbols $V, E, |.|$, and a unary external symbol $R$.

2. Binary static symbols $>, -$, Merge, and Intra, where $>$ is relational.

In every initial state of $A$,

- $V$, $U$ and $\in$ are as described above (up to isomorphism). $|s|$ is the cardinality of a set $s$, and $-$ is the set-theoretic difference. The relation $>$ is the standard ordering of natural numbers

- $E$ is a set of unordered pairs $\{x, y\}$ with $x, y \in V$ such that the graph $(V, E)$ is connected. $P$ is the finest partition $\{\{v\} : v \in V\}$ of $V$. Inter = $E$.

- If $e \in E$, $S$ is a partition of $V$, and $p, q$ are the $S$-ends of $e$, then

  - Merge$(e, S) = (S - \{p, q\}) \cup \{p \cup q\}$, and
  - Intra$(e, S) = \{\{x, y\} : x \in p \wedge y \in q\}$.

The external function $R$ takes a nonempty set and returns a member of it. The program of $A$ can be this:

```
if |P| > 2 then
    P := Merge(R(Inter), P)
    ||  Inter := Inter − Intra(R(Inter), P)
```

Now we apply the reversification procedure of Theorem 4.2, taking Remark 4.6 into account. We also take into account that both assignments fire at every step of the algorithm and so there is no need to record the firings. This gives us a faithful reversible expansion $B$ of $A$ with a program

```
if |P| > 2 then
    κ := κ + 1   ||
    P := Merge(R(Inter), P)   ||   P_0(κ + 1) := P   ||
    Inter := Inter − Intra(R(Inter), P)   ||   Inter_0(κ + 1) := Inter
```

The corresponding inverse ASM $C$ may be given by the program

```
if κ > 0 then
    κ := κ − 1   ||
    P := P_0(κ)   ||   P_0(κ) := nil   ||
    Inter := Inter_0(κ)   ||   Inter_0(κ) := nil
```

**Remark 5.2.** A custom crafted faithful expansion may be more efficient in various ways. For example, instead of recording the whole $P$, it may record just one of the two $P$-ends of $R$(Inter). This would require a richer vocabulary.

# 6 Conclusion

We have shown how to reversify an arbitrary sequential algorithm $A$ by gently instrumenting $A$ with bookkeeping machinery. The result is a step-for-step reversible algorithm $B$ whose behavior, as far as the vocabulary of $A$ is concerned, is identical to that of $A$.

We work with an ASM (abstract state machine) representation of the given algorithm which is behaviorally identical to it. The theory of such representation is developed in [13], and the practicality of it has been amply demonstrated.

## Acknowledgment

Many thanks to Andreas Blass for generous sanity check.

# References


[1] Anas N. Al-Rabadi, “Reversible logic synthesis: From fundamentals to quantum computing,” Springer Berlin, 2014

[2] Charles H. Bennett, “Logical reversibility of computation," *IBM Journal of Research and Development* 17:6 (1973), 525–532

[3] Egon Börger and Robert Stärk, “Abstract state machines: A method for high-level system design and analysis," Springer Berlin, 2003

[4] Andreas Blass and Yuri Gurevich, "Abstract state machines capture parallel algorithms," *ACM Transactions on Computational Logic* 4:4 (2003), 578–651. Correction and extension, ibid. 9:3 (2008), Article 19

[5] Andreas Blass, Yuri Gurevich and Benjamin Rossman, “Interactive small-step algorithms," Parts I and II; *Logical Methods in Computer Science* 3:4 (2007), Articles 3 and 4

[6] Alexis De Vos, “Reversible computing: Fundamentals, quantum computing, and applications,” Wiley-VCH Weinheim, 2010

[7] Nachum Dershowitz and Yuri Gurevich, “A natural axiomatization of computability and proof of Church’s thesis,” *Bulletin of Symbolic Logic* 14:3 (2008), 299–350

[8] Edsger W. Dijkstra, “Selected writings on computing: A personal perspective,” Springer New York, 1982

[9] Edward Fredkin and Tommaso Toffoli , “Concervative logic,” International Journal of Theoretical Physics 21:3/4 (1982), 219–253

[10] David Gries, “The science of programming,” Springer New York, 1981

[11] Yuri Gurevich, “Evolving algebras: An introductory tutorial,” The Bulletin of the EATCS 43 (1991), 264–284, and (slightly revised) in “Current Trends in Theoretical Computer Science: Essays and Tutorials” (eds. G. Rozenberg and A. Salomaa), World Scientific, 1993, 266–292. (Abstract state machines used to be called evolving algebras.)

[12] “Evolving Algebra 1993: Lipari Guide,” in Specification and Validation Methods (ed. E. Börger), Oxford University Press 1995, 9–36. Reprinted at arXiv, `https://arxiv.org/abs/1808.06255`. (Abstract state machines used to be called evolving algebras.)

[13] Yuri Gurevich, “Sequential abstract state machines capture sequential algorithms,” ACM Transactions on Computational Logic 1:1 (2000), 77–111

[14] Yuri Gurevich and Jim Huggins, "The semantics of the C programming language," in Proc. CSL'92, Computer Science Logic (eds. E. Börger et al.), Springer Lecture Notes in Computer Science 702 (1993), 274–308

[15] David Karger, "Global min-cuts in RNC and other ramifications of a simple min-cut algorithm," Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, 1993

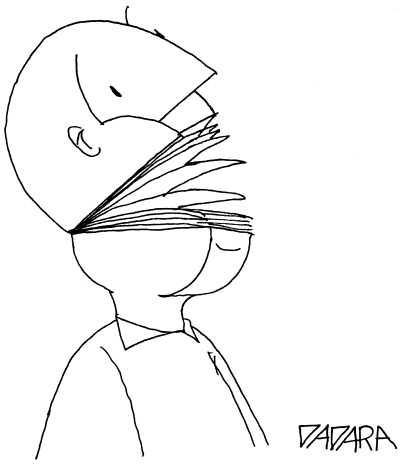
[16] Andrei N. Kolmogorov, "On the concept of algorithm," Uspekhi Matematicheskikh Nauk 8:4 (1953), 175–176 (Russian). English version in Vladimir Uspensky and Alexei Semenov, "Algorithms: Main ideas and applications," Kluwer 1993, 18—19

[17] Kenichi Morita, "Theory of reversible computing," Springer Japan, 2017

[18] Kalyan S. Perumala, "Introduction to reversible computing," CRC Press Boca Raton, 2014

[19] RC2021, 13th International Conference on Reversible Computation, `https://reversible-computation-2021.github.io/`

# Technical Contributions



# Sparse Integer Programming is FPT

Martin Koutecký
Computer Science Institute, Charles University
koutecky@iuuk.mff.cuni.cz

Shmuel Onn
Technion – Israel Institute of Technology
onn@technion.ac.il

**Abstract**

We report on major progress in integer programming in variable dimension, asserting that the problem, with linear or separable-convex objective, is fixed-parameter tractable parameterized by the numeric measure and sparsity measure of the defining matrix.

Integer linear programming, with data $w, l, u \in \mathbb{Z}^n$, $A \in \mathbb{Z}^{m \times n}$, and $b \in \mathbb{Z}^m$, is the problem

$$\min\{wx \ : \ Ax = b, \ \ l \leq x \leq u, \ \ x \in \mathbb{Z}^n\} \, . \tag{1}$$

It has a very broad expressive power and numerous applications, but is generally NP-hard. A well known result [4] asserts that integer linear programming is fixed-parameter tractable (see [1]) when parameterized by the dimension (number of variables) $n$, but this does not help in typical situations where the dimension is large and forms a variable part of the input.

Here we report on a recent powerful result in integer programming in variable dimension, asserting that the problem is fixed-parameter tractable when parameterized by the *numeric measure* $a := \|A\|_\infty := \max_{i,j} |A_{i,j}|$ and the *sparsity measure* $d := \min\{\mathrm{td}(A), \mathrm{td}(A^T)\}$ of $A$. Here $\mathrm{td}(A)$ is the *tree-depth* of $A$, defined below, and $A^T$ is the transpose. The result holds more generally for integer nonlinear programming where the objective function is *separable-convex*, that is, of the form $f(x) = \sum_{i=1}^n f_i(x_i)$ where each $f_i$ is a univariate convex function which takes on integer values on integer arguments and which is given by an evaluation oracle. Below we denote by $L := \log(\|u - l\|_\infty + 1)$ the bit complexity of the lower and upper bounds, and the times are in terms of the number of arithmetic operations and oracle queries.

**Theorem** *The linear or separable-convex program* (2) *is fixed-parameter tractable on $a, d$; and if $d = \mathrm{td}(A^T)$ and is fixed, it is polynomial time solvable even if unary encoded $a$ is variable:*

$$\min\{f(x) \ : \ Ax = b, \ \ l \leq x \leq u, \ \ x \in \mathbb{Z}^n\} \, . \tag{2}$$

*More specifically, there exist computable functions $h_1$ and $h_2$ such that the following hold:*

1. [3] *When $f(x) = wx$ is linear, the problem is solvable in fixed-parameter tractable time*

$$h_1(a,d)\text{poly}(n) \text{ if } d = \text{td}(A) \quad \textit{and} \quad (a+1)^{h_2(d)}\text{poly}(n) \text{ if } d = \text{td}(A^T)\,;$$

2. [2] *When $f(x)$ is separable-convex, it is solvable in fixed-parameter tractable time*

$$h_1(a,d)\text{poly}(n)L \text{ if } d = \text{td}(A) \quad \textit{and} \quad (a+1)^{h_2(d)}\text{poly}(n)L \text{ if } d = \text{td}(A^T)\,.$$

The theorem concerns *sparse integer programming* in the sense that at least one of $A$ and $A^T$ has small tree-depth, a parameter which plays a central role in sparsity, see [5], and which is defined as follows. The *height* of a rooted tree is the maximum number of vertices on a path from the root to a leaf. Given a graph $G = (V, E)$, a rooted tree on $V$ is *valid* for $G$ if for each edge $\{j, k\} \in E$ one of $j, k$ lies on the path from the root to the other. The *tree-depth* $\text{td}(G)$ of $G$ is the smallest height of a rooted tree which is valid for $G$. The graph of an $m \times n$ matrix $A$ is the graph $G(A)$ on $[n]$ where $j, k$ is an edge if and only if there is an $i \in [m]$ such that $A_{i,j}A_{i,k} \neq 0$. The *tree-depth* of $A$ is the tree-depth $\text{td}(A) := \text{td}(G(A))$ of its graph.

Here is a very rough outline of the proof. The complete details are in [2, 3].

**1. Few Graver-best steps suffice.** Define a partial order $\sqsubseteq$ on $\mathbb{R}^n$ by $x \sqsubseteq y$ if $x_i y_i \geq 0$ and $|x_i| \leq |y_i|$ for all $i$. The *Graver basis* of the integer $m \times n$ matrix $A$ is defined to be the finite set $\mathcal{G}(A) \subset \mathbb{Z}^n$ of $\sqsubseteq$-minimal elements in $\{z \in \mathbb{Z}^n : Az = 0,\ z \neq 0\}$. Given a feasible point $x$ in (2), a *Graver-best step* at $x$ is a step $s \in \mathbb{Z}^n$ such that $y := x + s$ is again feasible and has objective value at least as good as any feasible $x + cz$ with $c \in \mathbb{Z}_+$ and $z \in \mathcal{G}(A)$.

It can be shown that, starting from any feasible point, an optimal point can be reached using a suitably bounded number of Graver-best steps. And, an initial feasible point can be found, or infeasibility detected, by a suitable auxiliary integer program. See [6] for details.

**2. Graver norm bounds.** The parametrization by $a = \|A\|_\infty$ and $d = \min\{\text{td}(A), \text{td}(A^T)\}$ of the matrix $A$ enables to bound the norm of elements in its Graver basis $\mathcal{G}(A)$ as follows. It can be shown that there exist functions $g_1$ and $g_2$ such that, if $d = \text{td}(A)$ then $\|x\|_\infty \leq g_1(a, d)$ for all $x \in \mathcal{G}(A)$, whereas if $d = \text{td}(A^T)$ then $\|x\|_1 \leq (a+1)^{g_2(d)}$ for all $x \in \mathcal{G}(A)$.

**3. Finding Graver-best steps.** Let $x$ be a feasible point in (2) and let $c \in \mathbb{Z}_+$ be a given step size. Then a best step with step size $c$ is a solution of one of the following auxiliary integer programs in variables $z$, for each of the cases $d = \text{td}(A)$ and $d = \text{td}(A^T)$ respectively,

$$\min\{f(x+cz) \ :\ Ax = 0,\ \ l \leq x + cz \leq u,\ \ \|z\|_\infty \leq g_1(a,d),\ \ z \in \mathbb{Z}^n\}\,, \tag{3}$$

$$\min\{f(x+cz) \ :\ Ax = 0,\ \ l \leq x + cz \leq u,\ \ \|z\|_1 \leq (a+1)^{g_2(d)},\ \ z \in \mathbb{Z}^n\}\,. \tag{4}$$

Since the variables in these programs are bounded by functions of the parameters only, it can be shown that each of these programs can be solved efficiently by recursion on a suitable tree of small height, which certifies that either $d = \text{td}(A)$ or $d = \text{td}(A^T)$ respectively, is small. It can also be shown that a small list of potential step sizes $c \in \mathbb{Z}_+$ can be produced, and then the suitable program (3) or (4) is repeatedly solved for each step size in the list. Finally, the Graver-best step at $x$ is taken to be that $s := cz$ which gives the best improvement over all.

# References

[1] Cygan, M., Fomin, F.V., Kowalik, Ł., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., Saurabh, S.: Parameterized Algorithms. Springer (2015)

[2] Eisenbrand, F., Hunkenschröder, C., Klein, K.M., Koutecký, M., Levin, A., Onn, S.: An algorithmic theory of integer programming. ArXiv:1904.01361 1–63 (2019)

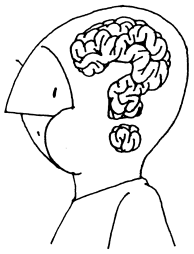
[3] Koutecký, M., Levin, A., Onn, S.: A parameterized strongly polynomial algorithm for block structured integer programs. Proceedings of ICALP 2018, Leibniz International Proceedings in Informatics, 107-85:1–14 (2018)

[4] Lenstra, H.W., Jr.: Integer programming with a fixed number of variables. Mathematics of Operations Research 8:538–548 (1983)

[5] Nešetřil, J., Ossona de Mendez, P.: Sparsity: Graphs, Structures, and Algorithms. Algorithms and Combinatorics, Springer (2012)

[6] Onn, S.: Nonlinear Discrete Optimization. Zurich Lectures in Advanced Mathematics, European Mathematical Society (2010). Available at: `http://ie.technion.ac.il/~onn/Book/NDO.pdf`

# News and Conference Reports



# Report on BCTCS 2021

## The 37th British Colloquium for Theoretical Computer Science

## 29–31 March 2021, University of Liverpool

Patrick Totzke and Michele Zito

The British Colloquium for Theoretical Computer Science (BCTCS) is an annual forum in which researchers in Theoretical Computer Science can meet, present research findings, and discuss developments in the field. It also provides a welcoming environment for PhD students to gain experience in presenting their work to a broader audience, and to benefit from contact with established researchers.

BCTCS 2021 was hosted by the University of Liverpool and held from 29th to 31st March, 2021. Due to the on-going COVID-19 pandemic, all talks were presented over Zoom, and `Gather.town` was used to encourage virtual networking. The meeting was well attended, attracting 128 registered participants, and featured an interesting and wide-ranging programme of five invited talks and 25 contributed talks.

The meeting spanned three days, with each morning session and each afternoon session opening with an invited keynote lecture. The meeting opened on the first morning with an invited talk by Eleni Akrida (Durham) on algorithmic properties of temporal graphs, and the afternoon session opened with an invited talk by Herbert Edelsbrunner (Vienna) on the properties of random polytopes inscribed on the 2-sphere. The second day opened with an invited talk by Alessandra Russo (London) on logic-based machine learning,. and the afternoon session opened with the LMS Keynote Lecture in Discrete Maths delivered by Professor Jan Krajíček (Prague), who gave an engaging presentation on proof complexity. The third and final day of the meeting opened with an invited lecture by Kousha Etessami (Edinburgh) on computing fixed points of monotone functions.

BCTCS 2022 will be hosted by Swansea University. Researchers and PhD students wishing to contribute talks concerning any aspect of Theoretical Computer Science are cordially invited to do so. Further details are available from the BCTCS website at `www.bctcs.ac.uk`.

## Invited Talks

**Eleni Akrida (Durham University)**
***Temporal graphs: Algorithms and complexity***

We discuss here the notion of temporal graphs, which are abstract models of networks that change as time progresses. In particular, a temporal graph on an underlying graph $G$ is such that the edges of $G$ have labels that are positive integers indicating discrete times (moments) of availability. A basic notion that arises from that definition is that of a journey, which extends the notion of a path in static graphs: it is a path in which subsequent edges appear with increasing labels. In this talk, we present recent work on temporal graphs through the lens of algorithms and complexity. We examine foremost journeys, design issues of temporal networks, as well as issues of traversal including temporal variants of the Travelling salesman Problem.

**Herbert Edelsbrunner (IST Vienna)**
***Random triangles and random inscribed polytopes***

Given three random points on a circle, the triangle they form is acute with probability $1/4$. In contrast, the triangle formed by three random points in the 2-sphere is acute with probability $1/2$. Both of these claims have short geometric proofs. We use the latter fact to prove that a triangle in the boundary of a random inscribed 3-polytope is acute with probability $1/2$.

Picking $n$ points uniformly at random on the 2-sphere, we take the convex hull, which is an inscribed 3-polytope. The expected mean width, surface area, and volume of this polytope are $2(n-1)/(n+1)$, $4\pi[(n-1)(n-2)]/[(n+1)(n+2)]$, and $(4\pi/3)[(n-1)(n-2)(n-3)]/[(n+1)(n+2)(n+3)]$. These formulas are not new but our combinatorial proofs are.

This is joint work with Arseniy Akopyan and Anton Nikitenko.

**Kousha Etessami (University of Edinburgh)**
***The complexity of computing a fixed point of a monotone function, and some applications***

The task of computing a fixed point of a monotone function arises in a variety of applications. In this talk I describe recent work in which we have studied the computational complexity of computing a fixed point of a given monotone function that maps a finite $d$-dimensional grid lattice with sides of length $N = 2^n$ to itself, where the monotone function is presented succinctly via a boolean circuit with $d \cdot n$ input gates and $d \cdot n$ output gates, and the underlying ordering is the standard coordinate-wise partial order on $d$-dimensional vectors in $[N]^d$. By Tarski's theorem, any monotone function has a fixed point.

We refer to the search problem of either finding a fixed point or finding a wit-

ness pair for non-monotonicity as the Tarski problem. It turns out that Tarski subsumes a number of important computational problems. In particular, it is essentially computationally equivalent to the task of computing a pure Nash Equilibrium for (succinctly presented) super-modular games, or games with strategic complementarities, which are very widely studied classes of games in economics. Also, computing the value of Condon's turn-based simple stochastic (reachability) games, as well as the more general problem of computing the value of Shapley's original stochastic games to within a given accuracy, is reducible to Tarski. Tarski is contained in both the total search complexity classes PLS and PPAD.

This is joint work with C. Papadimitriou, A. Rubinstein, and M. Yannakakis.

**Jan Krajíček (Charles University, Prague)**
***LMS Keynote Lecture in Discrete Maths***
***Proof complexity***

Proof complexity is an area connecting mathematical logic and computational complexity theory. It has several facets and I try to explain what some of these are. In particular, I discuss a few ways proof complexity relates to the existence of algorithms of various types.

**Alessandra Russo (Imperial College London)**
***Logic-based learning for interpretable AI***

Learning interpretable programs from data is one of the main challenges of AI. Over the last two decades there has been a growing interest in logic-based learning, a field of machine learning aimed at developing algorithms and systems for learning programs that can explain labelled data in the context of given background knowledge. Contrary to the black-box Deep Learning methods, logic-based learning systems learn programs that can be easily expressed into plain English and explained to human users, so facilitating a closer interaction between humans and machines. In this talk, I present recent advances in the field of logic-based learning. I introduce frameworks and algorithms for learning different classes of programs, ranging from definite programs under the Least Herbrand model semantics, to highly expressive non-monotonic programs under the Answer Set semantics. Such programs include normal rules, non-determinism, choice rules, hard and weak constraints, which are particularly useful for modelling human preferences. I discuss the relationship between these frameworks and illustrate a range of real-world problems that have been addressed using these systems, and open challenges in this area.

## Contributed Talks

**Duncan Adamson (University of Liverpool)**
***Ranking bracelets in polynomial time***

Ranking and unranking combinatorial objects is a well studied problem, with classical results for words, trees and partitions. More recently polynomial time algorithms have been provided for necklaces, also known as cyclic words. Despite this, the problem of ranking and unranking bracelets (also known as turnover necklaces) has remained a noted open problem. In this talk the first polynomial time algorithms for ranking and unranking bracelets are presented.

**Benjamin Bumpus (University of Glasgow)**
***Spined categories: generalising tree-width beyond graphs***

Problems that are NP-hard in general are often tractable on inputs that have a recursive structure. For instance consider classes defined in terms of "graph decompositions" such as of bounded tree- or clique-width graphs. Given the algorithmic success of graph decompositions, it is natural to seek analogues of these notions in other settings. What should a "tree-width-$k$" digraph or lattice or temporal graph even look like?

Since most decomposition notions are defined in terms of the internal structure of the decomposed object, generalizing a given notion of decomposition to a larger class of objects tends to be an arduous task. In this talk I show how this difficulty can be reduced significantly by finding a characteristic property formulated purely in terms of the category that the decomposed objects inhabit, which defines the decomposition independently of the internal structure. I introduce an abstract characterisation of tree-width as a vast generalisation of Halin's definition of tree-width as the maximal graph parameter sharing certain properties with the Hadwiger number and chromatic number. Our uniform construction of tree-width provides a roadmap to the discovery of new tree-width-like parameters simply by collecting the relevant objects into our new notion of a spined category.

This is joint work with Zoltan A. Kocsis.

**Marcel Dall'Agnol (University of Warwick)**
***Quantum proofs of proximity***

With the prevalence of massive datasets and small computing devices, offloading computation becomes increasingly important. One may model this problem as follows: a weak device (the *verifier*) holds an input string and communicates with an all-powerful computer (the *prover*) so as to verify the validity of this string. Since the verifier cannot perform the verification on its own, the prover provides a digest of the computational task, allowing the verifier to check that it was performed correctly. In the extreme setting of property testing, the verifier cannot

even read all of its input, and must decide whether it is valid or far from any valid string; in this case, the the digest is called a proof of proximity. We formalise the notion of quantum proofs of proximity, where prover and verifier are quantum computers, and show that a large class of properties admits quantum speedups. We also begin to chart the complexity landscape of the quantum/classical as well as the interactive/non-interactive variants of these proof systems.

This is joint work with Tom Gur and Subhayan Roy Moulik.

**Bhaskar DasGupta (University of Illinois at Chicago)**
***Maximizing coverage while ensuring fairness: a tale of conflicting objectives***

Ensuring fairness in computational problems has emerged as a key topic during recent years, buoyed by considerations for equitable resource distributions and social justice. It is possible to incorporate fairness in computational problems from several perspectives, such as using optimization, game-theoretic or machine learning frameworks. In this talk we address the problem of incorporating fairness from a combinatorial optimization perspective. We formulate a combinatorial optimization framework, suitable for analysis by researchers in approximation algorithms and related areas, that incorporates fairness in maximum coverage problems as an interplay between two conflicting objectives. Fairness is imposed in coverage by using colouring constraints that minimizes the discrepancies between number of elements of different colors covered by selected sets; this is in contrast to the usual discrepancy minimization problems studied extensively in the literature where (usually two) colors are not given a priori but need to be selected to minimize the maximum color discrepancy of each individual set. Our main results are a set of randomized and deterministic approximation algorithms that attempts to simultaneously approximate both fairness and coverage in this framework.

This is joint work with A Asudeh, T Berger-Wolf and A Sidiropoulos.

**Andrei-Cristian Diaconu (University of Oxford)**
***Controlling control: functional language design and implementation***

The design and implementation of functional languages is a problem that has been tackled from many perspectives. Generally, one starts with a formal specification of the semantics, after which this is used to derive an implementation of the language (e.g., an interpreter). The usual (operational) approaches to giving semantics are small-step, big-step and definitional interpreters, the last one capturing both a formal specification of the language and an actual implementation. However, the presentation of a functional language should aim to use a mixture of these (e.g., provide a small-step semantics, and then implement a definitional interpreter). This becomes a problem when languages with more complex control mechanisms such as exception handling and concurrency are considered, because the mentioned approaches generally do not scale well.

In this talk, we present a unified view of semantics and implementation, that scales well with the complexity of the language, whilst still giving one the ability to reason about it. We first present a simple and powerful way to specify small-step operational semantics using evaluation contexts, which allow us to succinctly capture complex control mechanisms. We then show how the idea of evaluation contexts is tightly linked to delimited continuations. Furthermore, delimited continuations can then be used to implement a definitional interpreter, which is both concise and highly extensible. To achieve this, we make use of various control operators and employ multi-prompt delimited continuations. Finally, we explore how the delimited continuation approach relates to other similar approaches of implementing interpreters, such as effects-and-handlers and monadic reflection.

**Alejandro Flores-Lamas (Royal Holloway, University of London)**
***Solving the distance k-clique problem in 1-outerplanar graphs***

A clique is a subset of vertices of a simple graph $G$ in which each vertex in the set is adjacent to all other vertices in it; in other words, any pair of such vertices are connected. Similarly, a distance $k$-clique is a subset of vertices in which any pair of vertices in the set are connected by at most $k$-edges; thus, a clique is a distance 1-clique set. A common formulation of these sets as a problem asks to find a Maximum Distance $k$-Clique, MD$k$C, from a given graph; i.e., the largest cardinality set. This problem is NP-hard in arbitrary graphs. In this work, we address the MD$k$C problem in 1-outerplanar graphs; i.e., graphs that admit a drawing on the plane such that its edges only intersect at the vertices, and each vertex lies on the outer face of the drawing. Our approach to solving this problem was first finding an MD$k$C set in a tree. Then, we adapted this algorithm to run in 1-outerplanar graphs. The proposed algorithm uses dynamic programming, and it goes through two phases. In the first phase, the algorithm follows a postorder traversal on a tree decomposition of $G$ to compute the size of an MD$k$C set. Then, during a second traversal (top-bottom) it identifies the vertices that belong to the set. The running time of this algorithm is $O\left((k+1)\cdot 2^{\tau+1}\cdot n\right)$, where $k$ comes from the MD$k$C problem, while $\tau$ and $n$ are the treewidth and order of $G$, respectively.

**Arved Friedemann (Swansea University)**
***Propagator networks for unification and proof search***

Logical languages have often shown to be useful as they give compact code for a huge variety of search problems. However, their performance highly depends on the underlying search engine. Propagator networks have been developed that combine the principles of SAT , SMT and CSP solvers all in one formalism. These networks can be used to get a general notion of what unit propagation means for more than just boolean functions and they can be easily parallelised. Here, we present a framework for propagator networks that can be used to easily model

unification and branching and create a logical language to perform proof search for PROLOG-like languages.

This is joint work with Philipp Dargel.

**Cameron Hargreaves (University of Liverpool)**
***The earth movers' distance as a metric for inorganic compositions***

Computational chemists have an intuitive grasp of what makes two compositions "chemically similar", but this concept is surprisingly difficult to capture using standard numeric techniques. Here we present the earth movers' distance (EMD) as a measure of similarity between two compounds, which operates by optimally pairing elements in a source composition to their most similar elements in a target composition and scoring the resultant similarities. This is akin to the cognitive process by which humans judge chemical similarity and as such, the resultant distances consistently align with human understanding. The chemical formula gives a mathematically abstract representation of the inorganic material which is typically difficult to work with. By contrast, we can use a well-formed distance in a wide range of analytical techniques, allowing us to cement these abstractions into tangible information. This is demonstrated effectively on the 12,567 binary structures of the ICSD, where we use this distance to plot detailed chemical maps which separate compositions into families of clear similarity both on a global and local scale. These maps can be clustered using unsupervised machine learning techniques to automatically partition our compounds into digestible subgroups which enables us to identify and distil critical chemical trends that would otherwise have been overlooked. We can additionally use these distances for the automated retrieval of structures from chemical databases, where an exact formulation may never have been reported but a closely related structure provides the reference needed. This metric is fast to compute between two compositions in practice, making it a strong candidate for many other applications in data driven materials discovery. We discuss how the EMD is calculated between two compositions, and demonstrate strengths against the standard metric.

**Alexandros Hollender (University of Oxford)**
***The complexity of gradient descent: CLS = PPAD ∩ PLS***

We study search problems that can be solved by performing gradient descent on a bounded convex polytopal domain and show that this class is equal to the intersection of two well-known classes: PPAD and PLS. As our main underlying technical contribution, we show that computing a Karush-Kuhn-Tucker (KKT) point of a continuously differentiable function over the domain $[0,1]^2$ is PPAD $\cap$ PLS-complete. This is the first natural problem to be shown complete for this class. Our results also imply that the class CLS (Continuous Local Search) – which was defined by Daskalakis and Papadimitriou as a more "natural" counter-

part to PPAD ∩ PLS and contains many interesting problems – is itself equal to PPAD ∩ PLS.

This is joint work with John Fearnley, Paul Goldberg and Rahul Savani.

**Finnbar Keating (University of Warwick)**
***Resolving data flow dependencies of concurrent programs with a graded monad***

In concurrent programs that interact with memory, it is important to know that variables are read from and written to in the correct order to prevent race conditions or inconsistent data. To this end, we introduce a graded monad for identifying the exact memory locations read and written by a given function. With this we can replace Haskell's IO monad with a variant indexed by a representation of the operations performed. This new information allows us to identify data-flow dependencies between functions at the type-level; that is, if one function reads a memory cell that is written to by another function we might wish to evaluate the latter before the former or vice-versa depending on the context.

With these dependencies known at compile-time, we then provide a function that establishes the correct order of execution for these program parts at compile-time with respect to their data-flow dependencies. This compile-time sort even allows us to identify functions that can be run in parallel, and can skip evaluation of code that relies on state that has not changed. We also prevent compilation in the presence of two errors: race conditions via multiple functions writing to the same memory cell, and dependency loops from functions writing to cells that other functions read from.

**Peter Kiss (University of Warwick)**
***Dynamic matchings***

We present a framework for deterministically rounding a dynamic fractional matching. Applying our framework in a black-box manner on top of existing fractional matching algorithms, we derive the following new results: the first deterministic algorithm for maintaining a $(2-\delta)$-approximate maximum matching in a fully dynamic bipartite graph, in arbitrarily small polynomial update time; the first deterministic algorithm for maintaining a $(1+\delta)$-approximate maximum matching in a decremental bipartite graph, in polylogarithmic update time; and the first deterministic algorithm for maintaining a $(2+\delta)$-approximate maximum matching in a fully dynamic general graph, in *small* polylogarithmic (specifically, $O(\log^4 n)$) update time.

Our rounding scheme works by maintaining a good *matching-sparsifier* with bounded arboricity, and then applying the algorithm of Peleg and Solomon to maintain a near-optimal matching in this low arboricity graph. To the best of our knowledge, this is the first dynamic matching algorithm that works on general graphs by using an algorithm for low-arboricity graphs as a black-box subroutine.

**Nina Klobas (Durham University)**
***Fast recognition of some parametric graph families***

Understanding the cycle (or anticycle) structure of a graph is fundamentally related to graph families such as trees, perfect graphs, bipartite graphs, chordal graphs, pancyclic graphs, and many others. A particularly strong cycle-related property is the notion of cycle-regularity, introduced by Mollard, which has been used to better understand the structure of graph families such as hypercubes or generalized Petersen graphs. In this talk we present three graph families, namely I-graphs, double generalized Petersen graphs and folded cubes and show how their cyclic structure helped us devise linear time recognition algorithms for them.

**Joe Livesey (University of Liverpool)**
***Propositional gossip protocols***

Gossip protocols are programs that can be used by a group of agents to synchronise what they know. Assuming each agent holds a secret, the goal of a protocol is to reach a situation in which all agents know all secrets. Distributed epistemic gossip protocols use epistemic formulas in the component programs for the agents.

We investigate open problems regarding propositional gossip protocols, in which whether an agent wants to make a call depends only on their currently known set of secrets. Specifically, we show that all correct propositional gossip protocols, i.e., ones that always terminate in a situation where all agents know all secrets, require the communication graph to be complete, whilst investigating the minimum number of calls required. We also investigate the complexity of the problem of checking correctness of a given propositional gossip protocol, before discussing implementing such a check with model checker NuSMV.

**Lily Major (Aberyswyth University)**
***Developments with manipulating Lyndon factorizations using evolutionary computation methods***

String factorization is an important tool for partitioning data for parallel processing and other algorithmic techniques often found in the context of big data applications such as bioinformatics or compression. Duval's well-known linear algorithm uniquely factors a string over an ordered alphabet into Lyndon words, patterned strings which are strictly smaller than all of their cyclic rotations. While Duval's algorithm produces a pre-determined factorization, modern applications motivate the demand for factorizations with specific properties, e.g., those that minimize/maximize the number of factors, or consist of factors with similar lengths.

We considered the problem of finding an alphabet ordering that yields a Lyndon factorization with such properties. For problems with no known heuristics, evolutionary computation methods may be useful, as such, we produced a flexible

evolutionary framework and evaluated it on biological sequence data. For the minimization case, we proposed a new problem-specific heuristic, Flexi-Duval, and a problem-specific mutation operator for Lyndon factorization. We have shown that for individual amino acid sequences, very long (often single) Lyndon factors can be produced using Flexi-Duval, and in comparison with Duval's algorithm, a much more balanced length for each Lyndon factor is achievable for specific texts.

We also consider the balancing problem for a much larger range of input texts, including random sequences, repetitions with some introduced noise, and natural language text corpora. We consider improvements to the Flexi-Duval algorithm and evaluate it with the broader input texts, and consider the problem of maximizing the number of Lyndon factors via evolutionary computation methods.

**Diptapriyo Majumdar (Royal Holloway, University of London)**
***Tractability of Konig edge deletion problems***

A graph is said to be a Konig graph if the size of its maximum matching is equal to the size of its minimum vertex cover. The Konig edge deletion problem asks if a graph has a set of at most k edges whose deletion results in a Konig graph. While the vertex deletion version of this problem was shown to be fixed-parameter tractable (FPT) more than a decade ago, the FPT status of Konig edge deletion has remained open since then. It has been conjectured to be W[1]-hard in several papers. In this paper, we settle the conjecture by proving it W[1]-hard. In addition, we prove that a variant of this problem, where we are given a graph G and a maximum matching M, and we want a Konig edge deletion set of size at most k that is disjoint from M, is fixed-parameter tractable.

This is joint work with Rian Neogi, Venkatesh Raman, and S. Vaishali.

**Michael McKay (University of Glasgow)**
***A non-trivial polynomial time algorithm for a 3D stable roommates problem***

In this talk we consider possible formalisations of the three-dimensional stable roommates problem (3D-SR). Players specify preference lists over their peers, and the task is to partition the players into triples based on their preferences. A number of hardness results exist under various schemes of preference representation. We consider a formalisation of 3D-SR in which agents' preferences are additively separable and binary, named 3D-SR-AS-BIN. The decision problem then asks whether we can partition the players into triples so that no three players would prefer to be in a set together than remain in their current triples. We show that 3D-SR-AS-BIN is NP-complete and consider its restriction in which preferences are symmetric, named 3D-SR-SAS-BIN. We show that every instance of 3D-SR-SAS-BIN contains a stable matching that can be found using a non-trivial algorithm in polynomial time. These results help us explore the boundary between NP-hardness and polynomial-time solvability in problems of coalition formation.

**Jay Morgan (Swansea University)**
***Trustable machine learning systems***

Machine Learning (ML) has had a remarkable impact on society. Everything from the phones in our pockets, to the cars that we drive, are being increasingly outfitted with this progressively sophisticated suite of algorithms. But while many of the most basic and fundamental algorithms from ML can be formally verified and tested for safety without much trouble, the same may not be said for Deep Learning (DL) – a prominent forerunner in the state-of-the-art for ML research. These DL models, while performing simple matrix-to-matrix operations at a micro-level, have evolved in scale far past what is tractable for current formal verification methods – all in the pursuit of improving accuracy and performance. This issue of tractability is unsettling considering that the existence of adversarial examples is well known in the ML community. These adversarial examples occur when very small changes to the input space result in a large change in the output space and cause a misclassification made by the DL model. In the context of self-driving vehicles, small defects and visual artefacts in the sensor input of the DL model, could lead the vehicle to wrongly conclude a stop sign indicates to continue driving where it should have stopped. While the manufacturers will need to put safeguards in place to prevent this from happening, we should formally prove the (non)-existence of these adversarial examples in the DL model itself. In this presentation, I present the foundational knowledge for understanding adversarial examples, how we can use the input space to dictate the search space for the existence of these examples, and demonstrate their presence with the use of SAT-solving. This work, as a free and open-source project, provides a framework for ML practitioners to verify their own architectures.

**Eric Munday (University of Edinburgh)**
***Forcing infinite memory for lim inf total payoff objectives in countable MDPs***

We look at an example of a countable MDP with integer-valued transition rewards. Every infinite run induces a sequence of total payoffs (the sequence of sums of all rewards seen so far). The objective is to maximise the probability that the lim inf is non negative. We present a counterexample to show that infinite memory is required in order to satisfy the lim inf objective for epsilon-optimal strategies. Furthermore, this holds even if the step-counter is implicit in the state and the MDP is finitely branching.

**Kheeran Naidu (University of Bristol)**
***A unifying class of algorithms for semi streaming bipartite maximum matching***

In the semi-streaming model of Feigenbaum et al., a graph with $n$ vertices is presented to an algorithm as a stream of edges where the storage space of the algo-

rithm is bounded by $O(n \text{ polylog } n)$. It provides an efficient model for processing massive graphs which have quickly become widespread. An algorithm in this model typically takes anywhere from one pass up to logarithmically many passes of the stream, in the same order.

A long-standing open problem is to improve upon a $\frac{1}{2}$-approximation for maximum cardinality matching (MCM) in one pass of the stream (in adversarial order). Currently, a simple greedy algorithm achieves the state-of-the-art approximation factor. Improving upon a $\frac{1}{2}$-approximation in two passes for bipartite MCM, however, was achieved by Konrad et al. with a $(\frac{1}{2}+0.0192)$-approximation. Kale and Tirodkar, and Esfandiari et al. independently improved these bounds to a $(\frac{1}{2}+0.0625)$- and $(\frac{1}{2}+0.0833)$-approximation, respectively. Most recently, Konrad set the state-of-the-art for bipartite MCM at a $(\frac{1}{2}+0.0858)$-approximation in two passes of the stream.

We present a wider class of two-pass bipartite MCM algorithms of which the above algorithms are special cases. We show that there are two optimal algorithms in this class which achieve the state-of-the-art, one of which is a novel algorithm. Finally, we construct a worst-case example which achieves the analytical lower bound, proving that: the analysis is indeed tight; a $(\frac{1}{2}+0.0858)$-approximation is best for this class of algorithms; and a completely different algorithmic approach will be needed to further improve this bound.

This is joint work with Christian Konrad.

**Adam Ó Conghaile (University of Cambridge)**
***Partition games, compositionally: Structure and power of linear algebraic games***

Spoiler-duplicator games have been a crucial tool for proving expressibility upper bounds on logical languages. Recently, a large body of work on game comonads has exposed a fascinating compositional structure lying beneath the surface of many of these games, relating games to each other and to relevant structural parameters such as treewidth. This has helped to better understand the pebble game, modal bisimulation game and games for generalised quantifiers. One class of games whose category theoretic structure is not yet understood is that of partition games. As explored in the thesis of Bjarki Holm, partitions give a rich grammar for constructing spoiler-duplicator games for various logics including the matrix equivalence and invertible maps games for linear algebraic logics. In this talk, I briefly review recent developments in game comonads before introducing partition games and describing where they might fit into this compositional picture.

This is joint work with Anuj Dawar.

**Bruno Pasqualotto Cavalar (University of Warwick)**
***Monotone circuit lower bounds from robust sunflowers***

Monotone Boolean circuits form one of the largest natural circuit classes for which

we are able to prove exponential size lower bounds. Such lower bounds play a pivotal role in complexity theory, being a proxy for lower bounds on communication complexity, proof complexity and optimization. For over 20 years, the best known lower bound on the size of monotone circuits computing an explicit $n$-bit monotone Boolean function was $\exp(n^{1/3-o(1)})$.

In this talk, we motivate the study of monotone circuits and their applications, and present the first lower bound for monotone circuit size of order $\exp(n^{1/2-o(1)})$. The proof employs the classical approximation method of Razborov and recent robust sunflower bounds of Alweiss, Lovett, Wu and Zhang, and of Rao.

This is joint work with Mrinal Kumar and Benjamin Rossman.

**Nicos Protopapas (University of Liverpool)**
***Impartial selection, additive guarantees, and prior information***

We study the problem of impartial selection, a topic that lies in the intersection of computational social choice and mechanism design. The goal is to select the most popular individual among a set of community members. The input can be modelled as a directed graph, where each node represents an individual, and a directed edge indicates nomination or approval of a community member to another. An impartial mechanism is robust to potential selfish behaviour of the individuals and provides appropriate incentives to voters to report their true preferences by ensuring that the chance of a node to become a winner does not depend on its outgoing edges. Our goal is to design impartial mechanisms that select a node with an in-degree that is as close as possible to the highest in-degree. Recent progress has identified impartial selection rules with optimal approximation ratios. It was noted though that worst-case instances are graphs with few vertices. Motivated by this fact, we propose the study of additive approximation, the difference between the highest number of nominations and the number of nominations of the selected member, as an alternative measure of quality. We present randomized impartial selection mechanisms with additive approximation guarantees of $o(n)$, where $n$ is the number of nodes in the input graph. We furthermore demonstrate that prior information on voters' preferences can be useful in the design of simple (deterministic) impartial selection mechanisms with good additive approximation guarantees. In this direction, we consider different models of prior information and analyze the performance of a natural selection mechanism that we call approval voting with default.

**Benjamin Smith (University of Liverpool)**
***Fixed-parameter tractability of pinwheel scheduling***

The pinwheel scheduling problem asks if it is possible to construct a perpetual schedule for a set of $k$ tasks of unit length where the maximum gap between repetitions of the same task is limited by an ordered multiset $A$ of $n$ constraints, one per

task. We demonstrate that pinwheel scheduling is fixed-parameter tractable with respect to the parameter $k$. We also show that, for any given value of $n$, a finite set of schedules can solve *all solvable* pinwheel scheduling instances. We then introduce exhaustive-search algorithms for both pinwheel scheduling instances and partial pinwheel scheduling instances (where only a prefix of $A$ is known and gaps have to be left in the schedule), and use that to construct the Pareto surfaces over all possible schedules for $k$ tasks, for $k \leq 5$. These results have implications for the bamboo garden trimming problem of Gasieniec et al.

**Siani A. Smith (Durham University)**
***The complexity of acyclic, star and injective colouring for $H$-free graphs***

Colouring is among the best known problems in computer science. One direction of study has been to consider the complexity of colouring for $H$-free graphs, that is graphs which do not contain $H$ as an induced subgraph. A full dichotomy is known in the case where the number of available colours, $k$, is part of the input whereas infinitely many open cases remain where $k$ is fixed.

We study three variants of the colouring problem, acyclic, star and injective colouring. An acyclic colouring of a graph $G$ is a proper colouring in which the union of any two colour classes induces a forest. Similarly a star colouring of $G$ is a proper colouring in which the union of any two colour classes is $P_4$-free, in other words it induces a star forest. Finally, an injective colouring, also known as a distance 2 colouring or an $L(1,1)$-labelling, is a proper colouring in which the union of any two colour classes is $P_3$-free. In each case we combine new and known results to obtain a full complexity dichotomy where $k$ is fixed and classify all but at most finitely many graphs $H$ where $k$ is part of the input. In fact, for both star and injective colouring, we show that only one open case remains.

This is joint work with J Bok, N Jedličková, B Martin and D Paulusma.

**Federico Vastarini (University of York)**
***Random graph generation using hyperedge replacement grammars***

This work addresses the problem to satisfy the necessity of graph based applications when requesting random test data. May they involve classical algorithms, software testing, pointer manipulation, pattern recognition or even complex networks, the proposed solution, allows for the specification of any graph domain. Such approach, that, to the best of our knowledge has never been followed by any pre-existing method, uses context-free hyperedge replacement grammars to define graph classes, over which, elements are sampled uniformly at random.

In order to achieve this, a generalised efficient version of Mairson's algorithms for the sampling of strings over non-ambiguous context-free grammars in Chomsky normal form is adapted to the setting of hyperedge replacement. The presented method is correct in that given a non-ambiguous hyperedge replacement grammar

$G$ in Chomsky normal form and a size $n$, a $n$-hypergraph in the language $L(G)$ is generated uniformly at random.

**Pavel Vesely (University of Warwick)**
***Relative error streaming quantiles***

Approximating distributions and quantiles is a fundamental task in data analysis. We consider streaming algorithms that make just one pass over a massive dataset and must use a small amount of memory, polylogarithmic in the input size. The problem of approximating the median and other quantiles with an additive error is sufficiently well-understood, however, the case of the stronger relative error is still open. Such an error guarantee provides more accurate estimates for extreme quantile queries (such as the 99.5th percentile), thus helping to understand the tails of the distribution. We describe a new streaming algorithm providing the relative error guarantee with a nearly optimal trade-off between space usage and accuracy.

This is joint work with G Cormode, Z Karnin, E Liberty, and J Thaler.

# Report from EATCS Japan Chapter

*Yukiko Yamauchi* (Kyushu University)

## EATCS-JP/LA Workshop on TCS and Presentation Awards

The 19th *EATCS/LA Workshop on Theoretical Computer Science* was held online February 1st to 3rd, 2021. (The details can also be found, although this website is written in Japanese, at
`http://www-ppl.ist.osaka-u.ac.jp/la2020/winter.php`.)

Every year, we choose the best presenter and the best student presenter. This year, we celebrated the following presentation as the 19th LA/EATCS-Japan Presentation Award:

> "*Spanning trees with the maximum number of leaves of grid graphs*", **Masahisa Goto** and Koji M. Kobayashi (University of Tokyo)

We celebrated the following presentation as the 10th LA/EATCS-Japan Student Presentation Award:

> "*NP-completeness of k Generalized Lunar Lockout Variant*", **Kosuke Kagaya**, Masaki Tomisawa, and Hiroaki Tohyama (Maebashi Institute of Technology)

The awards were recognized publicly on the last day, February 3rd, 2021.

**Congratulations!**

This workshop is jointly organized by *LA symposium*, Japanese association of theoretical computer scientists. Its purpose is to give a place to discuss topics on all aspects of theoretical computer science. This workshop is an unrefereed meeting. All submissions are accepted for the presentation. There should be no problem of presenting these papers at refereed conferences and/or journals. This meeting is unofficial, familiar, and widely open for everyone who is interested in theoretical computer science. It is held twice a year (January/February and July/August). If you have a chance, I recommend that you attend it. Check the website `http://www.ecei.tohoku.ac.jp/alg/EATCS-J/` for further details. The list of the presentations is as below; you can see the activity of the Japanese society of theoretical computer science.

**Program of EATCS-JP/LA workshop on TCS (February 1–3, 2021)**

In the following program, "*" indicates ordinary speakers, while "**" indicates student speakers. The number [S*xx*] means it is in student session, namely, it is shorter talk than ordinary one.

[S1] On the strongest die in a set of dice with equal expected values
***Shang Lu, Shuji Kijima (Kyushu University)*

[S2] Computational complexity of partitioning a weighted tree into subtrees of almost equal weights with exceptions
***Masashi Ito (Nagoya University), Shuichi Miyazaki (Kyoto University), Shinsaku Nakajima (Meiji University), Hirotaka Ono, Yota Otachi (Nagoya University)*

[S3] NP-completeness of $k$ Generalized Lunar Lockout Variant
*Kosuke Kagaya , Masaki Tomisawa, Hiroaki Tohyama (Maebashi Institute of Technology)*

[S4] Computer experiments for Single Pile NIM with prohibition rule
***Shota Asaba, Koichi Yamazaki (Gunma University)*

[S5] Max-Min Approximation Algorithms for the 3-Item Bin Packing Problem
***Rina Atsumi, Hiroshi Fujiwara, Hiroaki Yamamoto (Shinshu University)*

[S6] Searching algorithm of quantum phase estimation using optical interferometer
***Takuto Ozu, Ryuhei Mori (Tokyo Institute of Technology)*

[S7] Optimal Online Bin Packing Algorithms for Certain Classes of Two Item Sizes
***Masaya Kawaguchi, Hiroshi Fujiwara, Hiroaki Yamamoto (Shinshu University)*

[S8] Distributed Complexity of $k$-Maximal Independent Set Verification
***Ryosuke Sato, Naoki Kitamura, Ryota Eguchi, Yonghwan Kim (Nagoya Institute of Technology), Taisuke Izumi (Osaka University)*

[S9] Search and evacuation by a modular robotic system in a 3D grid field
***Ryonosuke Yamada, Yukiko Yamauchi (Kyushu University)*

[S10] Compact index for full-text search using factor oracle
***Akihide Sato, Hiroaki Yamamoto, Hiroshi Fujiwara (Shinshu University)*

[S11] Implementation of CA150 on Turing Tumble
***Keigo Shimonozono, Shuichi Inokuchi (Fukuoka Institute of Technology)*

[S12] Containment of a mobile fault by blocking communication links and connectivity of communication graph
***Hinata Hanzawa, Yukiko Yamauchi (Kyushu University)*

[S13] Distributed Complexity of Minimum Spanning Tree in Unit Disk Graphs with Euclidean Edge Weights
***Aru Hokodate, Naoki Kitamura, Ryota Eguchi, Yonghwan Kim (Nagoya Institute of Technology), Taisuke Izumi (Osaka University)*

[S14] Physical Zero-Knowledge Proof for Kurodoko
***Ryo Itoyama (Kumamoto University), Yota Otachi (Nagoya University)*

[S15] A Match-Three Game with Continuous Moves is NP-Complete
***Keitaro Kawagoe, Yasuhiko Takenaga (The University of Electro-Communications)*

[S16] Mathematical Studies of Deadlock in Adaptive Match Scheduling

***Yuta Tanaka, Hiroshi Fujiwara, Hiroaki Yamamoto (Shinshu University)*

[S17] Spanning spiders in bubble-sort graphs

*Yosuke Kikuchi, **Chinatsu Sakamoto (NIT, Tsuyama College)*

[1] Cake Cutting: A Simple Envy-Free and Truthful Mechanism with a Small Number of Cuts

**Takao Asano (Chuo University)*

[2] A graphical generalization of SEVENS

**Hironori Kiya (Nagoya University), Koki Suetsugu (National Institute of Informatics), Hirotaka Ono (Nagoya University)*

[3] Structural properties of graphs containing many minimal ab separators

**Koichi Yamazaki (Gunma University)*

[4] Spanning trees with the maximum number of leaves of grid graphs

**Masahisa Goto, Koji M. Kobayashi (University of Tokyo)*

[5] Reconfiguration problem based on re-unfolding of polyhedra

**Tonan Kamata, Ryuhei Uehara (Japan Advanced Institute of Science and Technology)*

[6] On particle complexity of number conserving cellular automata

**Gil-Tak Kong, Katsunobu Imai (Hiroshima University)*

[7] A reduction of the DTW distance to the LIS length

**Yoshifumi Sakai (Tohoku University), Shunsuke Inenaga (Kyushu University)*

[8] Tighter lower bounds for the error probability of quantum algorithms discriminating multiple quantum channels

**Ryo Ito, Ryuhei Mori (Tokyo Institute of Technology)*

[9] Communication Complexity of Quantum Private Simultaneous Messages Protocols

**Akinori Kawachi (Mie University), Harumichi Nishimura (Nagoya University)*

[10] Research on Rep-cube — dissection of net of cube to nets

**Tamami Okada, Ryuhei Uehara (Japan Advanced Institute of Science and Technology)*

[11] Indexing structures for labeled trees

**Shunsuke Inenaga (Kyushu University)*

## Forthcoming Events

### ISAAC 2021

International Symposium on Algorithms and Computation (ISAAC) is intended to provide a forum for researchers working in algorithms and theory of computation. The 32nd edition of this symposium will be held in Fukuoka, Japan (+ Online) from December 6 to 8, 2021. See `https://tcs.inf.kyushu-u.ac.jp/isaac2021/` for more information on ISAAC 2021.

### WALCOM 2022

The 16th International Conference and Workshops on Algorithms and Computation (WALCOM 2022) will be held at University of Jember, Indonesia from March 24th to 26th, 2022. This conference has been established to encourage the researchers of theoretical computer science in Asia, especially, India and Bangladesh. Nowadays, there are many

participants not only from a wide range of Asia but also from Europe and North America. See `https://walcom2022.unej.ac.id` for more information on WALCOM 2022.

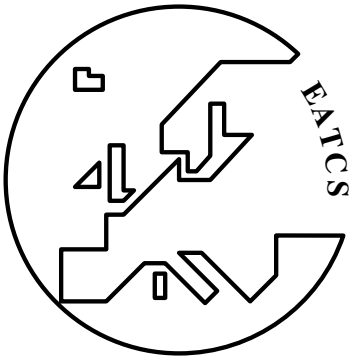
---

## EATCS Japan Chapter


| | |
|---|---|
| Chair: | Ryuhei Uehara |
| Vice Chair: | Takehiro Ito |
| Secretary: | Yukiko Yamauchi |
| email: | eatcs-jp@grp.tohoku.ac.jp |
| URL: | http://www.ecei.tohoku.ac.jp/alg/EATCS-J/index.html |


---

# 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 career prizes, including the EATCS Award, the Gödel Prize (with SIGACT), the Presburger Award, the EATCS Distinguished Dissertation Award, the Nerode Prize (joint with IPEC) 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: CIAC (Conference of Algorithms and Complexity), CiE (Conference of Computer Science Models of Computation in Context), DISC (International Symposium on Distributed Computing), DLT (International Conference on Developments in Language Theory), ESA (European Symposium on Algorithms), ETAPS (The European Joint Conferences on Theory and Practice of Software), LICS (Logic in Computer Science), MFCS (Mathematical Foundations of Computer Science), WADS (Algorithms and Data Structures Symposium), WoLLIC (Workshop on Logic, Language, Information and Computation), WORDS (International Conference on Words).

Benefits offered by EATCS include:

- Subscription to the "Bulletin of the EATCS;"
- Access to the Springer Reading Room;
- 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."

Benefits offered by EATCS to Young Researchers also include:

- Database for Phd/MSc thesis
- Job search/announcements at Young Researchers area

## (1) THE ICALP CONFERENCE

ICALP is an international conference covering all aspects of theoretical computer science and now customarily taking place during the second or third week of July. Typical topics discussed during recent ICALP conferences are: computability, automata theory, formal language theory, analysis of algorithms, computational complexity, mathematical aspects of programming language definition, logic and semantics of programming languages, foundations of logic programming, theorem proving, software specification, computational geometry, data types and data structures, theory of data bases and knowledge based systems, data security, cryptography, VLSI structures, parallel and distributed computing, models of concurrency and robotics.

SITES OF ICALP MEETINGS:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, UK 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, UK 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologne, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006
- Wrocław, Poland 2007
- Reykjavik, Iceland 2008
- Rhodes, Greece 2009
- Bordeaux, France 2010
- Zürich, Switzerland 2011
- Warwick, UK 2012
- Riga, Latvia 2013
- Copenhagen, Denmark 2014
- Kyoto, Japan 2015
- Rome, Italy 2016
- Warsaw, Poland 2017
- Prague, Czech Republic 2018
- Patras, Greece 2019
- Saarbrücken, Germany (virtual conference) 2020

## (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 for-

mats, 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 EATCS Monographs and Texts are now M. Henzinger (Vienna), J. Hromkovič (Zürich), M. Nielsen (Aarhus), G. Rozenberg (Leiden), 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 Editor-in-Chief of the journal currently are D. Sannella (Edinburgh), L. Kari and P.G. Spirakis (Patras).

## ADDITIONAL EATCS INFORMATION

For further information please visit `http://www.eatcs.org`, or contact the President of EATCS:

*Prof. Paul Spirakis,*
*Email:* `president@eatcs.org`

## EATCS MEMBERSHIP

DUES

The dues are € 40 for a period of one year (two years for students / Young Researchers ). Young Researchers, after paying, have to contact `secretary@eatcs.org`, in order to get additional years. 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 € 35 per year. We also offer a five-euro discount on the EATCS membership fee to those who register both to the EATCS and to one of its chapters. Additional € 35 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, contact the Secretary Office of EATCS:

*Mrs. Efi Chita,*
*Computer Technology Institute & Press (CTI)*
*1 N. Kazantzaki Str., University of Patras campus,*
*26504, Rio, Greece*
*Email:* `secretary@eatcs.org`,
*Tel: +30 2610 960333, Fax: +30 2610 960490*

If you are an EATCS member and you wish to prolong your membership or renew the subscription you have to use the Renew Subscription form. The dues can be paid via paypal and all major credit cards are accepted.

For adittional information please contact the Secretary of EATCS:

*Prof. Emanuela Merelli*
*via Madonna delle Carceri, 9*
*Computer Science Build. 1st floor*
*University of Camerino,*
*Camerino 62032, Italy*
*Email:* `secretary@eatcs.org`,
*Tel: +39 0737402567*