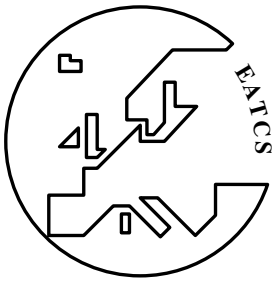
ISSN 0252–9742

Bulletin

of the

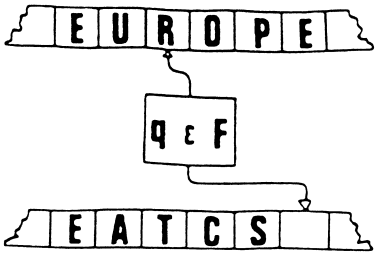
European Association for Theoretical Computer Science

EATCS



Number 137 June 2022

Council of the European Association for Theoretical Computer Science



President:	Artur Czumaj	United Kingdom
Vice Presidents:	Anca Muscholl	France
	Giuseppe F. Italiano	Italy
Treasurer:	Jean-Francois Raskin	Belgium
Bulletin Editor:	Stefan Schmid	Germany

Ivona Bezakova	USA	Anca Muscholl	France
Tiziana Calamoneri	Italy	Luke Ong	UK
Thomas Colcombet	France	Tal Rabin	USA
Artur Czumaj	UK	Eva Rotenberg	Denmark
Javier Esparza	Germany	Maria Serna	Spain
Fabrizio Grandoni	Switzerland	Alexandra Silva	USA
Thore Husfeldt	Sweden, Denmark	Jiri Sgall	Czech Republic
Giuseppe F. Italiano	Italy	Ola Svensson	Switzerland
Fabian Kuhn	Germany	Jukka Suomela	Finland
Slawomir Lasota	Poland	Till Tantau	Germany
Elvira Mayordomo	Spain	Sophie Tison	France
Emanuela Merelli	Italy	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
Tiziana Calamoneri . . . . . calamo@di.uniroma1.it
Thomas Colcombet . . . . . Thomas.Colcombet@irif.fr
Artur Czumaj . . . . . A.Czumaj@warwick.ac.uk
Javier Esparza . . . . . esparza@in.tum.de
Fabrizio Grandoni . . . . . fabrizio@idsia.ch
Thore Husfeldt . . . . . thore@itu.dk
Giuseppe F. Italiano . . . . . giuseppe.italiano@uniroma2.it
Fabian Kuhn . . . . . kuhn@cs.uni-freiburg.de
Slawomir Lasota . . . . . sl@mimuw.edu.pl
Elvira Mayordomo . . . . . elvira@unizar.es
Emanuela Merelli . . . . . emanuela.merelli@unicam.it
Anca Muscholl . . . . . anca@labri.fr
Luke Ong . . . . . luke.Ong@cs.ox.a.uk
Tal Rabin . . . . . chair.sigact@sigact.acm.org
Jean-Francois Raskin . . . . . jraskin@ulb.ac.be
Eva Rotenberg . . . . . eva@rotenberg.dk
Maria Serna . . . . . mjserna@cs.upc.edu
Stefan Schmid . . . . . stefan.schmid@tu-berlin.de
Alexandra Silva . . . . . alexandra.silva@cornell.edu
Jiri Sgall . . . . . sgall@iuuk.mff.cuni.cz
Ola Svensson . . . . . Ola.Svensson@epfl.ch
Jukka Suomela . . . . . jukka.suomela@aalto.fi
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, Berlin, Germany
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 2ε using the class `beatcs.cls` (a version of the standard LaTeX 2ε `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.

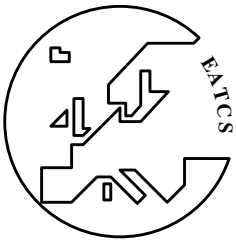
The details can be found at `http://www.eatcs.org/bulletin`, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email `bulletin@eatcs.org`.

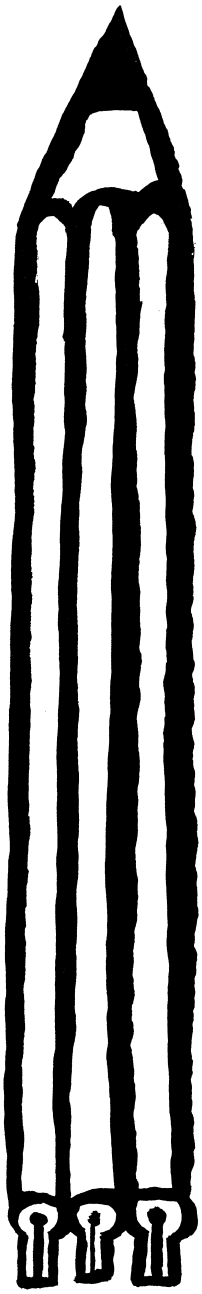
Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

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

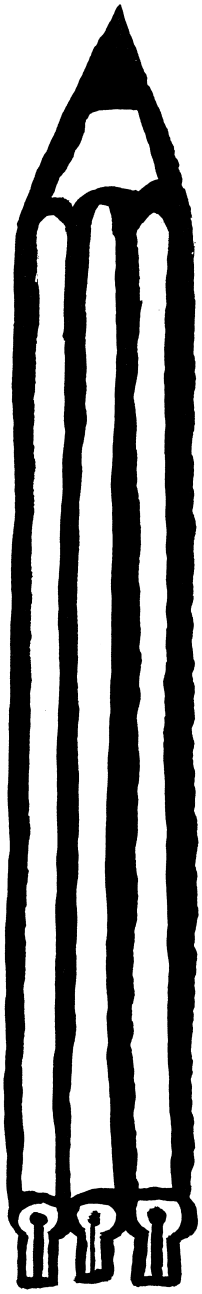




Dear EATCS members,

I hope my letter finds you and your family safe and in good health. While we are still living in the times marked by the global coronavirus pandemic that has a major impact on our lives and our scientific activities, I hope that the situation is improving and we will be able to have a batter balance of online and of networking activities. This spring I have already attended a few international workshops and conferences, and it seems that we may be coming back to more active scientific interaction. 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.

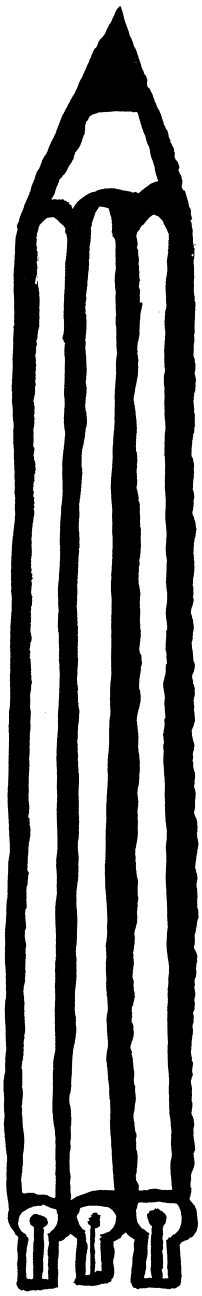
As usual, the June issue of the Bulletin will be available just before ICALP, the flagship conference of the EATCS and an important meeting of the theoretical computer science community world-wide. The 49th ***EATCS International Colloquium on Automata, Languages, and Programming (ICALP 2022)****, will be held in Paris, France, July 4-8, 2022 (URL: https://icalp2022.irif.fr/). The conference will be run in a hybrid mode, with almost all talks delivered in-person, though online participation will be made possible. The conference chair Thomas Colcombet, supported by his colleagues in Paris, promises us an exciting scientific event. I am very grateful to Thomas Colcombet and his team (including Sandrine Cadet, Olivier Carton, Geoffroy Couteau, Hugo Férée, Irène Guessarian, Natalia*



Hacquart, Florian Horn, Maximilien Lesellier, Simon Mauras, Valia Mitsou, Sylvain Perifel, Amaury Pouly, Arnaud Sangnier, Sylvain Schmitz, Mahsa Shirmohammadi, Laurent Viennot) for the extraordinary work they have done in organizing ICALP 2022.

An important part of ICALP 2022 will be the celebration of the ***50th anniversary of both the first ICALP and of EATCS*** *- with year 1972 marking the birth of European theoretical computer science. The first ICALP conference was organized by Maurice Nivat in July 1972 in Rocquencourt, Paris, in the premises of IRIA, now INRIA. At roughly the same time, the efforts to establish a Europe-centered scientific organization in Theoretical Computer Science led to the submission to EEC and to Belgian authorities of legal documents to create EATCS on June 24, 1972; this is the official date of the constitution of EATCS. A royal decree by the King of Belgium from September 4, 1972 officially created EATCS and approved its statute. This has happened thanks to the main founders of EATCS: Giorgio Ausiello (Italy), Jaco de Bakker (The Netherlands), Maurice Nivat (France), Mike Paterson (UK), Manfred Paul (Federal Republic of Germany), Michel Sintzoff (Belgium), and Leo Verbeek (The Netherlands), and with a broad support of the Theory community. I hope to see many of you joining us in these celebrations of the 50th anniversaries of ICALP and of EATCS during the ICALP 2022 conference in Paris!*

In the scientific part of the ICALP program, the Programme Committee chairs David Woodruff (track A) and Mikołaj



Bojańczyk (track B) and their PCs have done fantastic job selecting an impressive collection of papers, 102 accepted papers in track A and 24 in track B out of 429 submissions (342 for track A and 87 for track B). The acceptance rate was 29.4 percent. The programme of ICALP 2022 will highlight research across many areas within theoretical computer science. I invite you to attend and/or watch the talks even outside your own research field.

The ***best paper awards at ICALP 2022*** *will go to the following two articles:*

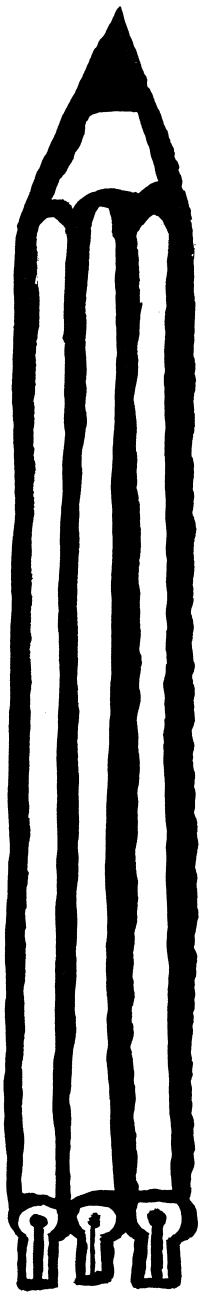
- *Track A: Ilan Newman and Nithin Varma. Strongly Sublinear Algorithms for Testing Pattern Freeness;*
- *Track B: Jakub Gajarský, Marcin Pilipczuk, Michał Pilipczuk, Wojciech Przybyszewski, and Szymon Toruńczyk. Twin-width and Types.*

The ***best student paper award*** *for a paper that is solely authored by a student will go to the following three papers:*

- *Track A: Joakim Blikstad. Sublinear-round Parallel Matroid Intersection;*
- *Track A: Jakub Tětek. Approximate Triangle Counting via Sampling and Fast Matrix Multiplication;*
- *Track B: Gaëtan Douéneau-Tabot. Hiding Pebbles when the Output Alphabet is Unary.*

Congratulations to the authors of the award-receiving papers!

In addition to regular research talks, ICALP 2022 will feature six invited talks delivered by

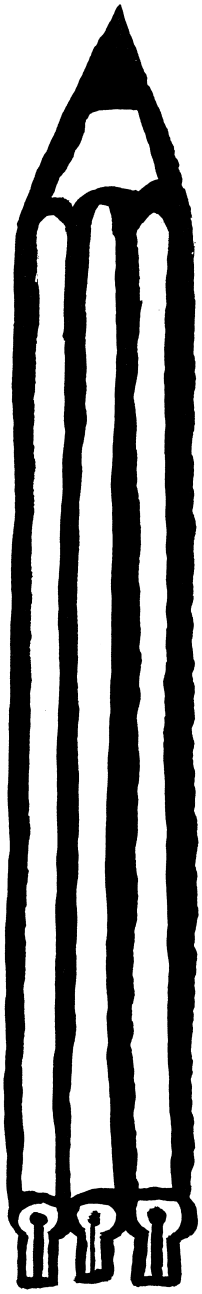


- *Albert Atserias (Universitat Politècnica de Catalunya),*
- *Constantinos Daskalakis (MIT),*
- *Leslie Ann Goldberg (Oxford),*
- *Madhu Sudan (Harvard, USA),*
- *Stéphan Thomassé (l'Ecole Normale Supérieure de Lyon), and*
- *Santosh Vempala (Georgia Tech).*

Apart from the invited and contributed talks, ICALP 2022 will feature three special presentations:

- *of the* ***EATCS Award 2022*** *to* ***Patrick Cousot*** *(New York University), the recipient of the award for introducing and developing the framework of abstract interpretation for program analysis,*
- *of the* ***Presburger Award 2022*** *to* ***Dor Minzer*** *(MIT), the recipient of the award for his deep technical contributions towards resolving the 2-to-2 Games Conjecture, and*
- *of the* ***Gödel Prize 2022*** *(sponsored jointly by the EATCS and the ACM SIGACT) for the seminal work making transformative contributions to cryptography by constructing efficient fully homomorphic encryption schemes to* ***Zvika Brakerski*** *(Weizmann Institute of Science),* ***Craig Gentry*** *(Algorand Foundation), and* ***Vinod Vaikuntanathan*** *(MIT).*

Moreover, during the conference, we will honor the recipients of the 2021 EATCS



Distinguished Dissertation Award and the new group of EATCS Fellows.
The recipients of the ***2021 EATCS Distinguished Dissertation Award*** *are*

- *Alexandros Hollender, University of Oxford (advisor: Paul W. Goldberg),*
- *Jason Li, Carnegie Mellon University (advisors: Anupam Gupta and Bernhard Haeupler),*
- *Jan van den Brand, KTH Royal Institute of Technology (advisor: Danupon Nanongkai).*

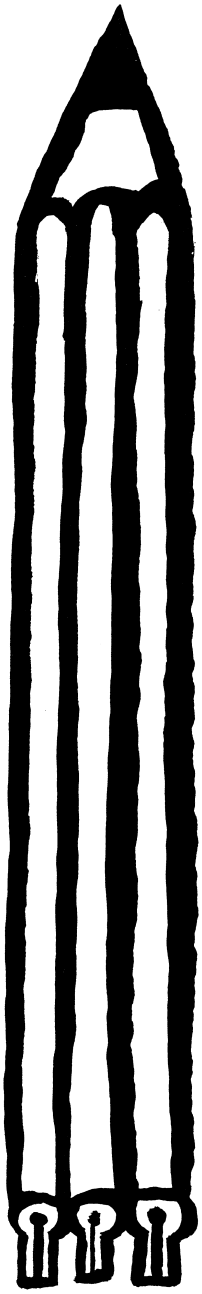
The new group of ***EATCS Fellows (class 2022)*** *recognized for their scientific achievements in the field of Theoretical Computer Science consists of*

- *Samson Abramsky (University College London, United Kingdom), and*
- *Orna Kupferman (Hebrew University, Israel).*

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 2022. (More details about the EATCS Award 2022, the Presburger Award 2022, the 2021 EATCS Distinguished Dissertation Award, and the EATCS Fellows are presented on the later pages of this issue of the Bulletin.)

ICALP 2022 will also have nine satellite workshops co-located with the main



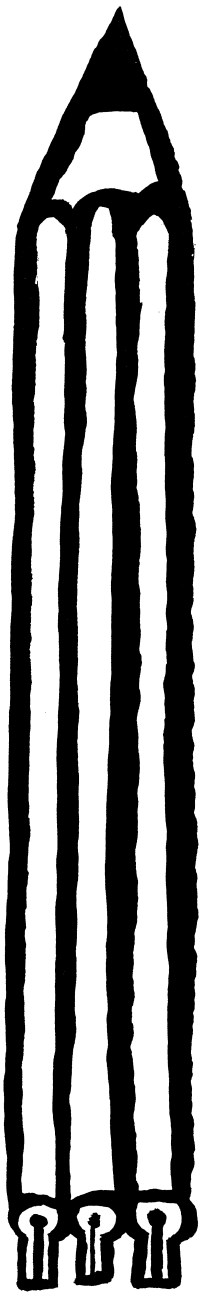
conference, taking place (online) on Sunday and Monday before the main event:

- *Parameterized Approximation Algorithms Workshop*
- *Combinatorial Reconfiguration*
- *Recent Advances on Total Search Problems*
- *LearnAut: 4th edition of the Learning and Automata workshop*
- *Algorithmic Aspects of Temporal Graphs V*
- *Trends in Arithmetic Theories*
- *Structure Meets Power 2022*
- *Straight-Line Programs, Word Equations and their Interplay*
- *Graph Width Parameters: from Structure to Algorithms*

As usual, a more detailed report on the ICALP 2022 conference will be published in the October 2022 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 2022**: the 47th International Symposium on Mathematical Foundations of Computer Science, will be held in Vienna, Austria, August 22--26, 2022 (https://ac.tuwien.ac.at/mfcs2022/).*
- ***ESA 2022**: the 30th Annual European Symposium on Algorithms, will be held in Potsdam, Germany, September 5-9, 2022 (https://algo2022.eu/esa/).*



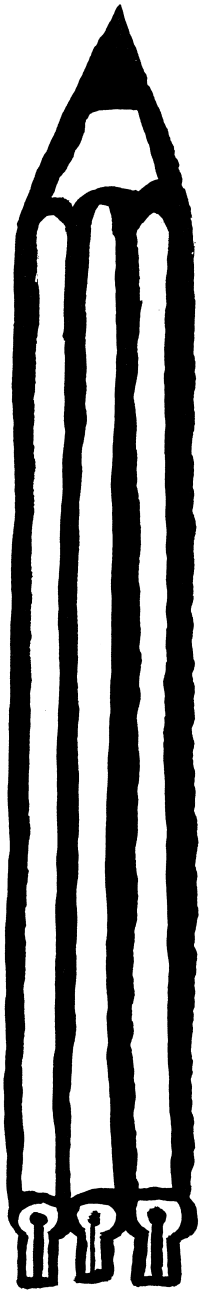
- ***DISC 2022:*** *the 35th International Symposium on Distributed Computing, will be held in Augusta, Georgia, October 25-27, 2022 (http://www.disc-conference.org/wp/disc2022/).*

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 and elswhere, let me list some 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 making transformative contributions to cryptography by constructing efficient fully homomorphic encryption schemes to the following two papers:*

- *Zvika Brakerski and Vinod Vaikuntanathan. Efficient Fully Homomorphic Encryption from (Standard) LWE. SIAM Journal of Computing 43(2): 831-871, 2014.*
- *Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. ACM Transactions on Computation Theory 6(3): 13:1-13:36, 2014.*

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*

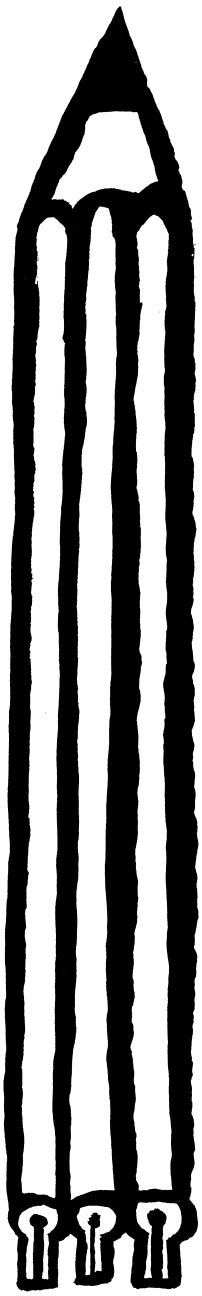


Computing (PODC) and the EATCS Symposium on Distributed Computing (DISC). The 2022 Dijkstra Prize has been awarded to the following two papers for providing the first general approach to memory reclamation in nonblocking data structures, with significant impact both in research and practice

- *Maged M. Michael. Safe Memory Reclamation for Dynamic Lock-Free Objects Using Atomic Reads and Writes. Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC), pages 21-30, Monterey, CA, USA, July 2002.*

- *Maurice Herlihy, Victor Luchangco, and Mark Moir. The Repeat Offender Problem: A Mechanism for Supporting Dynamic-Sized, Lock-Free Data Structures. Proceedings of the 16th International Symposium on Distributed Computing (DISC), pages 339-353, Toulouse, France, October 2002.*

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 2022 Alonzo Church Award has been awarded to Dexter Kozen (Cornell University) for his fundamental work on developing the theory and applications of Kleene algebra with tests, an equational system for reasoning about iterative programs, published in*



- *Dexter Kozen. Kleene Algebra with Tests. ACM Transactions on Programming Languages and Systems 19(3): 427-443, 1997.*

The prize will be formally presented at the Federated Logic Conference 2022 (FLOC2022), held in August in Haifa, Israel.

EATCS also co-sponsors the ***EATCS-IPEC Nerode Prize*** *for outstanding papers in the area of multivariate algorithmics, which this year was awarded to Bruno Courcelle for the following two papers:*

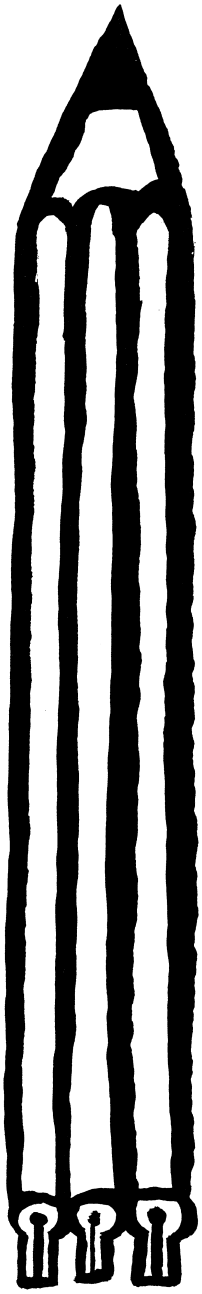
- *Bruno Courcelle. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. Information and Computation 85(1): 12-75, 1990.*

- *Bruno Courcelle. The Monadic Second-Order Logic of Graphs III: Tree-Decompositions, Minors and Complexity Issues. RAIRO - Theoretical Informatics and Applications 26: 257-286, 1992.*

EATCS also sponsors the ***Best ETAPS Paper Award 2022*** *for the best theory paper at ETAPS, which this year was awarded to the following paper:*

- *Emmanuel Hainry, Bruce Kapron, Jean-Yves Marion and Romain Péchoux, Complete and Tractable Machine-independent Characterizations of Second-order Polytime.*

Unfortunately, during the last months, we received the sad news that two active members of our community pass away: Gerhard Woeginger and Rolf Niedermeier. Gerhard Woeginger (RWTH in Aachen) was



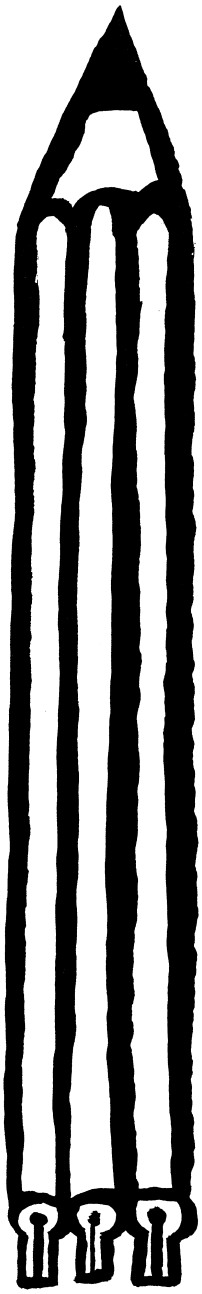
renowned for his research in approximation algorithms, online algorithms, scheduling, and also complexity theory, discrete mathematics, graph drawing, optimization, graph theory, social choice theory, and NP-hard problems (also thanks to his fantastic list of failed proofs of P vs NP); he was a very active member of the TCS community (including being the current EATCS Council member and a member of Academia Europaea). Rolf Niedermeier was a professor of computer science at TU Berlin, known for his research in computational complexity theory, especially in parameterized complexity, graph theory, computational social choice, and social network analysis. The entire Theory community will miss both of them dearly.

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 2022 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 2022



Dear EATCS member,

This issue of the Bulletin of the EATCS is a special one! For the Golden Jubilee, Giorgio Ausiello looks back with us at the 50-year history of the EATCS, and shows why it is still alive and well. The issue further includes a historical article, written by Ute and Wilfried Brauer, for the Silver Jubilee of the EATCS.

We also invited several members of the theoretical computer science community for an interview, and I would like to thank Robert Cori, Leslie Goldberg, Mariangiola Dezani-Ciancaglini, Anne Driemel, Christos H. Papadimitriou, Jean-Eric Pin, Eva Rotenberg, Grzegorz Rozenberg, and Paul Spirakis very much for sharing their perspectives, thoughts and memories with us.

As usual, the Bulletin includes technical columns and other typical contents such as book reviews, conference reports, and award announcements. In particular, check out Luca Trevisan's conversation with Scott Aaronson, the distributed computing column on Redbelly Blockchain, the education column about Tobias Kohn's minimal instruction set for education, or the logics column by Kenichi Morita on making reversible computing machines in reversible cellular space.

This month, I also had the opportunity to meet our logics editor, Yuri Gurevich, in person, in Berlin: thank you very much Yuri for your kind visit and all the memories and stories you shared!

I wish everyone an interesting read and a

look forward to seeing you at the EATCS celebrations in Paris!

Stefan Schmid, Berlin
June 2022

The EATCS Award 2022

Patrick Cousot

The EATCS Award 2022 is awarded to

Patrick Cousot

Patrick Cousot and his late wife Radhia Cousot introduced and developed the framework of abstract interpretation for program analysis. Abstract interpretation formalizes the interplay of static abstraction with dynamic execution for reasoning about the correctness of programs. Since its introduction in 1977, abstract interpretation has become one of the fundamental concepts in programming languages and compiler optimization and has greatly influenced the theory and practice of many related fields, from verification and software engineering to real-time systems and security. Dozens of new instantiations, extensions, and applications of the framework are published every year. Patrick has devoted a lifetime of research to abstract interpretation, having not only formulated the elegant mathematics that lies at the core the framework, but also leading its transition to industrial use, most notably through the Astree project for the analysis of avionics and space software. Recently Patrick summarized his life's work in the textbook "Principles of Abstract Interpretation."

The EATCS Award Committee 2022

- Éva Tardos (chair)
- Johan Håstad
- Thomas Henzinger

The award will be presented at ICALP 2022, in Paris.

The EATCS annually honors a respected scientist from our community with this prestigious EATCS Distinguished Achievements Award, to acknowledge extensive and widely recognized contributions to theoretical computer science over a life long scientific career (see http://eatcs.org/index.php/eatcs-award for more information, including the list of previous recipients)

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

2021 Toniann (Toni) Pitassi	2010 Kurt Mehlhorn
2020 Mihalis Yannakakis	2009 Gérard Huet
2019 Thomas Henzinger	2008 Leslie G. Valiant
2018 Noam Nisan	2007 Dana S. Scott
2017 Éva Tardos	2006 Mike Paterson
2016 Dexter Kozen	2005 Robin Milner
2015 Christos Papadimitriou	2004 Arto Salomaa
2014 Gordon Plotkin	2003 Grzegorz Rozenberg
2013 Martin Dyer	2002 Maurice Nivat
2012 Moshe Y. Vardi	2001 Corrado Böhm
2011 Boris (Boaz) Trakhtenbrot	2000 Richard Karp

The Presburger Award 2022

Laudatio for Dor Minzer

The 2022 Presburger Committee has unanimously selected

Dor Minzer

as the recipient of the 2022 EATCS Presburger Award for Young Scientists for his deep technical contributions towards resolving the 2-to-2 Games Conjecture.

The Unique Games Conjecture, formulated in 2002, is one of the central open questions in theoretical computer science, providing a plausible explanation of the hardness of approximation for a variety of natural problems, and weaving connections between computational complexity, algorithms, analysis, and geometry. While the jury is still out on this conjecture, the closely related variant of the 2-to-2 Games Conjecture was recently resolved by Minzer and his co-authors over a remarkable series of four papers between 2017 and 2018. "Lesser" variant notwithstanding, it has several important consequences, including establishing the hardness of distinguishing between almost-4-colourable graphs from almost-k-colourable graphs for constant k, and ruling out a polynomial-time-vs-truly-exponential-time dichotomy for approximating constraint satisfaction problems.

The proof of the 2-to-2 Games Conjecture involves a complex process of reformulating and reducing it to a concrete combinatorial hypothesis about the expansion properties of Grassmann graphs, and then proving this hypothesis using tools from the analysis of Boolean functions.

Minzer has also to his credit other strong and insightful results, in areas spanning property testing, information complexity, invariance and isoperimetry, noise sensitivity, and more. Minzer's work establishes him as a world leader in Boolean function analysis, an area central to obtaining and understanding many diverse and significant advances in theoretical computer science

The Presburger Committee 2022

- Mikołaj Bojańczyk
- Uriel Feige
- Meena Mahajan (chair)

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:

2021 Shayan Oveis Gharan
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

EATCS Distinguished Dissertation Award for 2022

EATCS is proud to announce that, after examining the nominations received from our research community, the EATCS Distinguished Dissertation Award Committee 2021, consisting of Susanne Albers, Nikhil Bansal, Elvira Mayordomo, Jaroslav Nesetril, Damian Niwinski, David Peleg (chair), Vladimiro Sassone and Alexandra Silva, has selected the following three theses as recipients of the EATCS Distinguished Dissertation Award for 2021:

Alexandros Hollender: Structural Results for Total Search Complexity Classes with Applications to Game Theory and Optimisation (University of Oxford; advisor: Paul W. Goldberg).,

Jason Li: Preconditioning and Locality in Algorithm Design (Carnegie Mellon University; advisors: Anupam Gupta and Bernhard Haeupler).,

Jan van den Brand: Dynamic Matrix Algorithms and Applications in Convex and Combinatorial Optimization (KTH Royal Institute of Technology; advisor: Danupon Nanongkai).

The award certificate will be presented to in the award ceremony of ICALP 2022, to take place in Paris, France, in July 4-8, 2022.

The EATCS Distinguished Dissertation Award Committee 2021 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 list of the previous recipients of the EATCS Distinguished Dissertation Award is available at https://eatcs.org/index.php/dissertation-award.

EATCS-Fellows 2022

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 2022 are:

Samson Abramsky, University College London, United Kingdom For fundamental contributions to logic in computer science, including domain theory in logical form, game semantics, and category-theoretic foundations of quantum computing.

Orna Kupferman, Hebrew University, Israel For fundamental contributions to automata- and game-theoretic techniques aiming at the formal verification and reactive synthesis of computing systems.

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
- Mariangiola Dezani (chair)
- 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.

2022 Alonzo Church Award for Outstanding Contributions to Logic and Computation

The ACM Special Interest Group on Logic (SIGLOG), the European Association for Theoretical Computer Science (EATCS), the European Association for Computer Science Logic (EACSL), and the Kurt Gödel Society (KGS) are pleased to announce that the 2022 Alonzo Church Award for Outstanding Contributions to Logic and Computation is given to

- **Dexter Kozen**

for his fundamental work on developing the theory and applications of Kleene Algebra with Tests, an equational system for reasoning about iterative programs, published in:

Kleene Algebra with Tests. ACM Transactions on Programming Languages and Systems 19(3): 427-443 (1997)

This work on Kleene Algebra with Tests (KAT) is one of the high points among remarkable contributions of Dexter Kozen to logics of programs. It is a culmination of a series of articles by Dexter Kozen that define and apply Kleene Algebra with Tests (KAT), an equational system that combines Kleene Algebra (the algebra of regular expressions) with Boolean Algebra (the tests). Together, the terms of the two algebras are capable of representing while programs, and their combined equational theory is capable of proving a wide range of important properties of programs. Although reasoning in KAT under arbitrary commuting conditions is undecidable, Kozen observes that when the commuting conditions are limited to including tests, it is decidable and in PSPACE. He illustrates the power of KAT with these decidable commuting conditions by proving a well-known folk theorem: Every while program can be simulated by a program with just one loop. KAT has been successfully applied to a variety of problems over the past 25 years, including modeling and reasoning about packet-switched networks.

The 2022 Alonzo Church Award Committee:

- Thomas Colcombet
- Mariangiola Dezani,

- Javier Esparza ,
- Radha Jagadeesan, (chair) 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/`.

2022 Edsger W. Dijkstra Prize in Distributed Computing

The 2022 Dijkstra Prize Award Committee concluded its deliberations and we are happy to announce that the papers

- "Safe Memory Reclamation for Dynamic Lock-Free Objects Using Atomic Reads and Writes," by Maged M. Michael. Proceedings of the 22nd ACM Symposium on Principles of Distributed Computing (PODC), Monterey, CA, USA, July 2002, pages 21–30.

- "The Repeat Offender Problem: A Mechanism for Supporting Dynamic-Sized, Lock-Free Data Structures," by Maurice Herlihy, Victor Luchangco, and Mark Moir. Proceedings of the 16th International Symposium on Distributed Computing (DISC), Toulouse, France, October 2002, pages 339–353.

have been selected by the committee to receive the Dijkstra Prize this year for providing the first general approach to memory reclamation in nonblocking data structures, with significant impact both in research and practice.

The Award Committee 2022:

- Christian Scheideler, Paderborn University (chair)
- Marcos Aguilera, VMware Research
- Alessandro Panconesi, Università La Sapienza, Rome
- Andrea Richa, Arizona State University
- Alexander Schwarzmann, Augusta University
- Philipp Woelfel, University of Calgary

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

Ackermann Award 2022

The EACSL Outstanding Dissertation Award for Logic in Computer Science 2022

Call for Nominations

Deadline: 1 July 2022

Nominations are now invited for the 2022 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 2020 and 31 December 2021 are eligible for nomination for the award. The deadline for submission is 1 July 2022. Submission details follow below.

The 2022 Ackermann Award will be presented to the recipient(s) at CSL 2023, 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);
- Maribel Fernandez (King's College London);

- Delia Kesner (IRIF, U Paris);
- Slawomir Lasota (U 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;
- James Worrell (U Oxford).

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 10 pages) of the thesis in ENGLISH (ps or pdf file); it is recommended to not squeeze as much material as possible into these 10 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 22 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.

Obituary Gerhard Woeginger (by Frits Spieksma) 1964–2022

He was unassuming. He was nice. He could think. He could write. And he knew a lot. He died Friday, April 1, 2022. His name was Gerhard Woeginger, a prolific computer scientist.

Some facts of his life and career can be easily summarized as follows. Born on May 31, 1964 in Graz, he studied at the Graz University of Technology, and got a PhD under Franz Rendl in 1991 with a thesis entitled "Geometric clustering, Reconstruction and Embedding Problems: Combinatorial Properties and Algorithms". He became a professor at the University of Twente (the Netherlands) in 2001, and joined TU/e as a professor in 2004. In 2016 he moved to RWTH Aachen. During his time at TU/e, he supervised around 10 PhD-students.

Describing the impact of his career on the field is not so easily summarized. He was present in almost every field within theoretical computer science. Social choice, bibliometrics, algorithms (especially online), approximability, computational geometry, and of course, one of his prime loves: computational complexity. His talent to see connections between different problems was amazing. His ability to distill the essentials, and then write it up in a way that it all seemed natural was uncanny. And his drive and enthusiasm to distinguish easy from hard, was absolutely infectious. His friendliness combined with a deep mathematical curiosity has been a source of inspiration for all around him.

He was on the program committee of an enormous number of conferences, he was program chair of ESA1997, MAPSP2005, IPCO2011, EURO2009, and he was on the board of a dozen journals among which OR Letters. He set up, and maintained the P-versus-NP page, a vintage Gerhard-style set of webpages that discusses attempts to settle the P=NP question. To say that he contributed to the Christmas puzzle (Advents Kalender) is an understatement, he single-handedly ensured the existence of it. And there is much, much more to be said.

Above all, he could listen - he was able to identify truth in one's unstructured words. And then he'd write the paper, faster than one thought was possible. We will miss his presence at conferences, his revealing questions at presentations, his modest smile when the result was discovered, and his knowledge. He knew

about the origins of the term NP-complete, he knew results from faraway times in obscure journals; he also knew about the ideal composition of a darts board, and he knew his soccer.

I am going to close with a personal anecdote. I visited Gerhard in Aachen, where we had dinner. He said "I feel a bit guilty about something I did while at TU/e - I stole something". I said: "Well, as TU/e is still standing, it can't have been too bad - what did you take?". "When they were cleaning out the library many years ago, I found a copy of a PhD thesis from 1962, written by Duyvesteyn. In it, Duyvesteyn finds a partition of a square into smaller squares. The corresponding picture is still on the cover of the Journal of Combinatorial Theory. I took the book with me, and actually, I have the book with me now. I want you to return it to TU/e." "Are you sure?" I asked. "Yes, I am sure." he said.

Obituary for Rolf Niedermeier

With great sadness, we announce that Rolf Niedermeier recently passed away unexpectedly at the age of 55. Many readers know Rolf from his book "Invitation to Fixed-Parameter Algorithms", which gave a welcoming and smooth introduction to the research area of parameterized algorithms.

Rolf's journey into the world of computer science started at TU Munich in 1986, where he obtained a degree in computer science in 1991. He then continued at the University of Tübingen, where he earned his PhD degree in 1996 and, after spending one year as a postdoc at Charles University in Prague, started an independent research group on parameterized algorithms. He then joined the University of Jena in 2004, where he obtained his first position as a professor and, finally, TU Berlin in 2010 where he established and chaired the research group "Algorithmics and Computational Complexity".

During this time, he achieved several seminal contributions for example on the improvement of depth-bounded search trees, the establishment of kernelization as a framework for parameterized algorithms, and in the quest to turn parameterized algorithms into practical implementations. He was one of the leading researchers in parameterized algorithmics, and worked on the application of parameterized algorithms to problems from various fields such as computational social choice, computational biology, and temporal graph theory with tireless enthusiasm.

Over his academic lifetime, Rolf supervised thirty Doctoral and countless Bachelor and Master students, taught a variety of stimulating courses, and headed more than fifteen different DFG-funded research projects. He was also co-organizer of numerous Dagstuhl workshops, including the first two Dagstuhl seminars on Parameterized Algorithms in 2001 and 2003, "Adaptive, Output Sensitive, Online and Parameterized Algorithms" in 2009, "Application-Oriented Computational Social Choice" and "Algorithms and Complexity in Phylogenetics" in 2019, and "Temporal Graphs: Structure, Algorithms, Applications" in 2021. Moreover, he was the local organizer for the "Workshop on Graph-Theoretic Concepts in Computer Science" (WG 2007), the "Workshop on Challenges in Algorithmic Social Choice" (CASC 2014), and the "Symposium on Theoretical Aspects of Computer Science" (STACS 2019). In addition, he tirelessly contributed to the research community in many different boards and committees (for example in the computer science review board of the DFG and in the review panel for ERC Starting Grants), as member of the steering committee for STACS, as dean of the faculty for Electrical Engineering and Computer Science at TU Berlin, as speaker of the algorithms group of GI, and in various other capacities.

Beyond his unconditional dedication to academia, Rolf was known as a warm-hearted and caring person who was always in the mood for a witticism. He used these to skillfully bridge any distance between him and his students, since they always came with an implicit expectation to

be returned with an equally pointed retort.

Rolf always had a positive attitude and was very approachable. You could always come to his office and ask him about anything. He would probably offer you a cup of tea and definitely give you his full attention—no matter how large the pile of other, more urgent or important work on his desk was.

We mourn for an inspiring mentor, bright scientist, supportive colleague, and good friend. Even though his passing leaves a large void, his influence will live on in all those who worked with him.

Institutional Sponsors

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

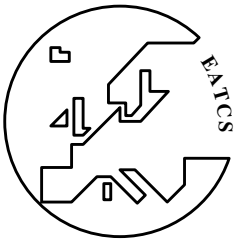
CWI, Centum Wiskunde & Informatica
Amsterdam, The Netherlands

MADALGO, Center for Massive Data Algorithmics
Aarhus, Denmark

Microsoft Research Cambridge
Cambridge, United Kingdom

Springer-Verlag
Heidelberg, Germany

EATCS
Golden Jubilee



EATCS Golden Jubilee: How EATCS was born 50 years ago and why it is still alive and well.

Giorgio Ausiello

Sapienza Università di Roma
ausiello@diag.uniroma1.it

The background

It was the year 1970 when Maurice Nivat, young mathematician, and his mentor Marcel-Paul Schützenberger, recently appointed director at IRIA (now INRIA), announce in a press conference, to about fifteen specialized journalists, the birth of a new scientific discipline that they call 'informatique thèorique' (i.e. theoretical computer science): 'the science which makes use of mathematical and logical tools to clarify and study the notion of computation'. Also the three great domains that formed the core of such new science were identified in the conference: the theory of automata and formal languages, the theory of algorithms and of computational complexity, and the theory of computer programming. "I don't know what happened to us on that day" says Maurice Nivat in 2008, in an interview with the journalist Isabelle Bellin. "The reality is that 40 years later this discipline exists and the number of researchers devoted to this domain of computer science has now increased fifty or may be one hundred times"[1].

Actually, as we all know, the study of mathematical foundations of computer science was not born in 1970; it had already a long history at that time. Theoretical issues in computing had been a research subject since the years Thirties, when the work of Alan Turing, Alonzo Church, Emil Post, and other logicians, had been focused on the characterization of computability and on the discovery of non computable functions and undecidable problems. Indeed these studies had been

[1] I. Bellin, Maurice Nivat: une vision à long terme de la recherche en informatique, https://hal.inria.fr/hal-01350155/

sources of inspiration for the creation of the first electronic computers. Subsequently, after the creation of the first generation of electronic computers technology again passed the floor to theory. Throughout the years Fifties and Sixties, a large amount of concepts and results regarding foundational aspects of computing devices and programming had been achieved and in 1970 such concepts and such results were already considered cornerstones of computer science. Just to make a few examples, in the area of models of computation we can remember the concepts of neural networks (McCulloch and Pitts, 1955) and the first examples of non deterministic processes (Rabin and Scott, 1959); regarding program syntax, we may remember the complete characterization of Chomsky languages in terms of generating grammars and of recognizing automata (Oettinger, Schützenberger, Myhill, Kuroda, Greibach, 1960-1965); regarding control structures in programs we have the results of Böhm and Jacopini (1966) that are at the base of structured programming; in program semantics we may consider the first notions of operational and denotational semantics (McCarthy, 1960, and Strachey, 1964), and the notion of axiomatic semantics (Floyd and Hoare, 1969); finally it is worth remembering the first results addressing the notion of computational complexity (Rabin, 1960, Hartmanis and Stearns, 1965, and Blum, 1966).

Two textbooks, appeared respectively in 1967 and in 1969, “Computation. Finite and Infinite Machines" by Minsky, and "Formal Languages and Their Relation to Automata” by Hopcroft and Ullman show that theoretical issues in computer science were already well defined and present in university curricula when the press conference of Nivat and Schützenberger took place.

So, what was the reason that moved Nivat and Schützenberger to announce the birth of a new scientifique domain? Actually, if we look back at those times we realize that there were many reasons to publicize and try to identify the new scientific domain. First of all the relentless growth of computer applications in all fields of human activity, banks, industry, public administration, that was taking place in those years was characterizing informatics essentially from the technological point of view, leaving in the back the foundational studies that were necessary to improve correctness and efficiency of applications. This was leading, at least in Europe, to lack of recognition in the academic world and inadequate funding for theoretical computer science with respect to other established domains like mathematics, physics, electrical engineering. In Italy mathematicians viewed computers just as tools for numerical computing, totally ignoring that, behind the instrument, another new mathematical discipline, the science of computing, was being shaped. This was not only happening in Italy but was a general attitude of mathematicians toward computer science. One of the leading figures in theoretical computer science, Michael Rabin, recalls: “There was absolutely no appreciation of the work

on the issues of computing. Mathematicians did not recognize the emerging new field", and from this point of view it is also interesting what Edsger Dijkstra said about his appointment at the Department of Mathematics at Eindhoven University of Technology: "Later I learned that I had been the department's third choice, after two numerical analysts had turned the invitation down" [2]. In any case, despite important achievements in some fields such as, for example, programming language design (think of the Algol 60 Report, mostly based on European contributions, and of the creation of the language Simula 67, a prototype of object oriented languages, implemented by Dahl and Nygaard) or software engineering (the first conference in this field took place in Garmish in 1968 thanks to the energetic role of various European scientists) European computer scientists were perceiving the existence of a large gap between the great development of US research in all aspects of computer science, and in theory in particular, and the European situation.

Also from the organizational point of view the European situation was lagging behind that in US. In 1968 ACM had created the Special Interest Group on Automata and Computability Theory (SIGACT, now Special Interest Group on Algorithms and Computation Theory) and in 1969 Patrick Fisher, the first Chair of SIGACT, professor at Waterloo, had started the Symposium on Theory of Computing (STOC). This was the second important US conference entirely devoted to theoretical computer science, the oldest being the IEEE Annual Symposium on Switching Circuit Theory and Logical Design (created in 1959 and in 1966 renamed Switching and Automata Theory [3]). No such scientific environment existed at European scale in the Sixties. Due to such situation European researchers did not have suitable occasions to meet. It was more customary for an Italian, French, British researcher to be in contact with US correspondents than with other European colleagues.

With all this in mind Maurice Nivat, Louis Nolin and Marcel-Paul Schützenberger moved in 1971 a second step that has become crucial in the development of the European theoretical computer science community. On June 25 a letter, prompted by Maurice Nivat and his colleagues, was sent by the Director of the Mathematics Department of the University of Paris VII François Bruhat to the European Commission. The letter recommended a cooperation concerning education and research in theoretical computer science to be established among European universities. The objective of the cooperation would be "to promote the exchange of information and of scientific results in the field and to organize specialized

[2] In this article various excerpts are taken from: G. Ausiello, The Making of a New Science, Springer 2018.

[3] In 1975 the conference received the current title: Foundations of Computer Science (FOCS).

schools and conferences, and visits of young researchers in laboratories of other countries" (a kind of ante litteram Marie Curie fellowship program). Besides Paris VII, the sites indicated as possible partners in the initiative were: Saarbrücken and Munich in Germany, Rome, Pisa and Turin in Italy, Amsterdam and Brussels for The Netherlands and Belgium, Paris VII and Toulouse for France. Furthermore, in the United Kingdom (at the time not a member of the EEC) the universities of Warwick, Edinburgh and Colchester would be invited to join the project.

According to Nivat's memories: "It was an extraordinary moment for computer science. University courses were started in various countries in Europe and elsewhere, computers had allowed humanity to reach the moon but informatics was not yet perceived as a real science. At the two Software Engineering conferences in Garmisch (1968) and Rome (1969) it had been realized that informatics could not be seriously developed without a solid methodological approach. But at the same time it was made clear that programming techniques and methods conceived by 'software engineers' could not reach their aims unless the notions of semantics, and complexity of computation could be analyzed on the basis of mathematical and logical rigorous foundations".

Indeed Nivat remembers that already in 1968 he had discussed with Alfonso Caracciolo, professor in Pisa, the possibility to create a European coperation in theoretical informatics. "It was 1968 when in a meeting I met an Italian fellow, Alfonso Caracciolo di Forino, who impressed me a lot for many reasons: he belonged to the noble Neapolitan family of the admiral Caracciolo who was hanged in 1799, ... chatting after the meeting he told Schützenberger and myself that, maybe, it would be possible to create a European entity dealing with the theoretical aspects of computer science since theory can be pursued with little money and does not raise the unsolvable financial and economic problems that were at stake when talking about cooperation in the software industry. He suggested that we create a European association" [4].

Four years later it was thanks to the support of Alfonso Caracciolo, at that time involved in the European PREST and COST initiatives for technological cooperation, that the proposal of the French colleagues successfully led to the meeting in Brussels that saw the birth of EATCS.

[4] M. Nivat, The True Story of TCS, Theoretical Computer Science, Special Issue for TCS 40th Anniversary (2015).

The Brussels meeting

The meeting took place on January 27th and 28th, 1972 under the title "Cooperation in the Field of Theoretical Data Processing" and it was chaired by Alfonso Caracciolo. The documents presented at the meeting contained three position papers on theoretical computer science.

The first, prepared by Nivat, Nolin and Schützenberger had been circulated since June 1971 as an attachment to Bruhat's letter. It was a 12 pages document containing, for the first time, the definition of the field of 'informatique théorique' (and also of the mathematical domains that, although related to computing, could not be classified theoretical computer science, such as numerical analysis and operations research). The field was articulated in three main chapters, theory of automata and formal languages, theory of algorithms and computational complexity, programming theory and for each chapter examples of the main results achieved until now were exposed. The second document had been prepared by Jaco de Bakker. Here it is claimed that the most important applications of theoretical computer science should be oriented toward i) improvements of 'individual programs' in terms of efficiency and correctness, ii) to the study of 'classes of programs' such as those for non-numerical applications and those devoted to data management, iii) to the formal definition of 'programming languages' both in terms of syntax and semantics, and finally iv) to the design of 'operating systems'. The third document had been prepared by Corrado Böhm. It starts by saying that 'theoretical informatics' was at that time not yet settled and well understood but the results of its development might have been 'explosive' with influence on all aspects of computing: hardware, basic software (operating systems, networks, programming languages), applications (efficiency of algorithms, correctness, software certification etc.). Overall we can observe that the vision of the founding fathers was definitely wide and farsighted. Besides in Nivat's document it was clearly stated that it was impossible to circumscribe the areas of a newly born science whose results might reach unforseeable goals.

Researchers from six countries plus several officers of the European Commission took part in the meeting. France was represented by Maurice Nivat, Louis Nolin, and the linguist Maurice Gross; Germany by Hans Langmaack and Karl Heinz Böhling, The Netherlands by Leo Verbeek and Jaco de Bakker, the UK by Mike Paterson, Belgium by Michael Sintzoff, Italy by Corrado Böhm, Ugo Montanari and Giorgio Ausiello. The agenda of the meeting included a survey of the activity of the Group PREST, the presentation by Louis Nolin of the French document containing the definition of the field of 'informatique théorique' and an overview of the importance of the field, an examination of the situation in the different

countries, and finally the presentation, discussion and, possibly, approval of the proposal prepared by Maurice Nivat on the cooperation among universities. This was indeed the hottest issue. Finally the general idea of the creation of a European organization for theoretical computer science was approved.

At first the idea was to follow the EMBO (European Molecular Biology Organization) example and to ask for a substantial funding from European institutions (80.000 US dollars for the first year supposed to grow to 200.000 US dollars in the subsequent years). In fact a few years before, in 1966, EMBO had been created with the aim to promote such discipline and to support scientific cooperation at European level among research institutions and universities, and had been assigned a relevant dowry that allowed to develop an ambitious program foreseeing organization of conferences, fellowships, exchange of visits, etc. After not so long it became clear that the EMBO experience could not be followed because to obtain funds from EEC was becoming hard. This is also reflected in the changes in the title of the initiative and in the articles of the proposed statute. At the beginning a document dated February 7th has the title 'Institut Européen d'Informatique Théorique' and considers that the institute should consist of 'associated institutes' and of simple 'members'. Subsequently the proposal had been transformed into Association Européenne d'Informatique Théorique – AEIT (in English European Association for Theoretical Computer Science – EATCS), and while still open to the participation of institutional members it had been conceived essentially as an association of 'individual members'.

Various provisional drafts of the statutes of the association prepared by EEC legal experts were circulated. On June 24th (official date of the constitution of EATCS) Michael Sintzoff met with de Bakker, Nivat and Paterson and they prepared the final documents for the creation of EATCS to be submitted to EEC and to Belgian authorities. The document would carry the names of the founders: Giorgio Ausiello (Italy), Jaco de Bakker (The Netherlands), Maurice Nivat (France), Mike Paterson (UK), Manfred Paul (Germany, rather, to be more precise, the Federal Republic of Germany), Michel Sintzoff (Belgium), and Leo Verbeek (The Netherlands). In the same meeting it was decided to nominate Leo Verbeek as president, Manfred Paul and Mike Paterson as vice-presidents, Maurice Nivat as secretary and Michael Sintzoff as treasurer. The process had its conclusion on September 4th, 1972 with the signature of a royal decree by the King of Belgium which set forth the creation of the 'Association Européenne d'Informatique Théorique' and the approval of its statute.

So, formally the association was there but how to give it life? After the creation of the scientific organization it was necessary to create the scientific community.

The first steps

The first step in the creation of the European theoretical computer science community was indeed already moved in July 1972 with the conference that Maurice Nivat organized in Paris (in the premises of IRIA). The title of the conference reflected the taxonomy of theoretical computer science that Maurice and his French colleagues had sketched in the charter of 'informatique théorique': Colloque sur la Théorie des Automates, des Langages et de la Programmation. The Program Committee was composed by outstanding computer scientists: Corrado Böhm, Samuel Eilenberg, Pat Fischer, Seymour Ginzburg, Gunther Hotz, Michael Rabin, Arto Salomaa, Adriaan van Wijngaarden. It was chaired by the father of the French school of theoretical computer science: Marcel-Paul Schützenberger. The conference had a big success and was attended, either as speakers or as participants, by a large group of young scientists that would make the history of theoretical computer science in the future years. The proceedings were published by North Holland and contained 49 papers. It is interesting to know that 34 papers were in English, 14 in French, and 1 in German. ICALP (the International Colloquium on Automata, Languages and Programming) was born. While the first edition of the conference was sponsored by ACM-SIGACT since the edition of 1974 the conference will be sponsored by EATCS and since 1976 will become the big annual event that we all know, one of the major world events in theoretical computer science.

The next important step that was taken to make EATCS grow was to get involved in the project a large number of colleagues from all over the world. In September 1972 Maurice Nivat sent an invitation letter to a long list of scientists in Europe, US and Israel (but he already had in mind the involvement of computer scientists also from Soviet Union, India and Japan). Nivat was soliciting our colleagues to join EATCS "not only to give it life but also to contribute to define aims, scope and activities". As we will see the need to open the scope of EATCS beyond the research fields defined in the first documents has been a constant objective of the founders. One of the characters that make EATCS alive and well after so many years derives from the effort that, throughout these years, the scientists that have led the Association have put to constantly update the scientific horizon of our research domain and to open the annual conference to new hot and stimulating subjects.

March 24th-25th, 1973 marks another important date: the first General Assembly of EATCS took place in Warwick, hosted by Mike Paterson. During the meeting the first Council of EATCS was elected by adding to the founding members six leading figures of theory of computing from Europe and US (C. Böhm, W. Brauer,

B. Mayoh, R. Milner, J.F. Perrot, D. Scott).

Then the Council appointed Maurice Nivat President of EATCS and de Bakker and Paterson Vice-Presidents. In order to circulate information and promote the exchange of ideas in the community it was decided that beside organizing an annual conference and supporting specific informal events EATCS would have initiated to edit a scientific bulletin containing information regarding open problems, events and activity of research groups. A second General Assembly took place in Hamburg on October 7, 1973 and in that occasion the Council was further enlarged to include members from Israel (Z. Manna), Switzerland (E. Engeler) and Finland (A. Salomaa).

In December 1973 Maurice Nivat edited the first EATCS Bulletin. Again the issue of broadening the scope of the Association is addressed by Nivat in the Editorial when he says that the initial list of topics identifying the scope of the Association is by no means limitative: "defining the limits of theoretical computer science is at least as difficult as defining the limits of computer science itself. And we strongly believe that a science is what the scientists at work make it: certainly new areas of computer science will be open to theory in the near future. Let us start small and grow: we are sure many of you will help to achieve this necessary growth up to the point where our Association will be a natural link between all European theoretical computer scientists". The first issue of the Bulletin also contained reports from Universities and research centers that help to understand what was the scientific atmosphere in Europe in those years: abstract computational complexity, Lindenmayer systems, grammatical inference, reproduction of automata, correctness of recursive programs, recursive and iterative program schemes, program semantics, combinatory logic and lambda-calculus.

Two more steps have contributed to establish the role of EATCS already in the years Seventies and Eighties of the last century: the creation by Maurice Nivat of the journal 'Theoretical Computer Science' in 1975 and the launch in 1981 of other important editorial initiatives: the 'EATCS Monographs on Theoretical Computer Science' and the 'EATCS Texts on Theoretical Computer Science'. All such initiatives were not autonomous but were based on the collaboration with commercial publishers (North Holland - Elsevier in the first case and Springer in the second and third case); what matters is that although started in the restricted EATCS circle they were directed to, and reached, the entire world community of theoreticians. Initially the project to start a journal was communicated by Maurice Nivat to the EATCS Council as an initiative involving the EATCS community. While the journal was not formally linked to the Association it had seven members of the EATCS Council in its Editorial Board and (as Maurice said in a letter

to EATCS members) the creation of the journal had been 'inspired' by the Association. Indeed, since volume 12, in 1980, to volume 80, in 1991, the front page of the journal carried the words: 'The journal of the EATCS'. But at beginning of the year Nineties the roads of the Association and of the journal started to diverge mostly because EATCS felt the need to adopt an open attitude towards all journals devoted to theory of computing.

As far as the EATCS Monographs and Texts book series are concerned, this might again be considered a success story in the life of the Association. When the initiatives started in 1983 the Editors of the series were Wilfried Brauer, Grzegorz Rozenberg and the EATCS President Arto Salomaa. Many of the first volumes rapidly became fundamental textbooks in the field, reaching a public much wider than the EATCS members and carrying the EATCS logo on the desk of students worldwide.

The life of EATCS

In the paper Silver Jubilee of EATCS (1997) Ute and Wilfried Brauer say: "That EATCS members identify themselves with their association is mainly due to the ICALP series and to the Bulletin". Twentyfive years later we can undoubtedly confirm that ICALP and the Bulletin still are the pillars of the life of the Association but to them we should add several other initiatives that today make our Association so dynamic and vital.

I do not think it is worth now to enumerate all the developments that underwent along the life of the Association so let me just outline what I think are the most fundamental aspects that show why EATCS is 'well and alive' 50 years after its creation.

ICALP. The success of ICALP and its standing among the top world conferences in theory of computing is well known. Every year it is confirmed by about 500 papers submitted by excellent researchers, from 40-50 countries, mostly from Europe, America and Asia. The conservative selection rate (ranging from 28.2 and 29.8 in the last 10 years) is a guarantee of the quality of the conference and all years several good papers cannot be accepted due to the tough competition. Throughout the years EATCS has made an effort to update format and content of the conference in view of the scientific evolution of theoretical computer science. In the year 1997 the decision was taken to divide the conference in two tracks: Track A, Algorithms, Complexity and Games, and Track B, Automata, Logic, Semantics and Theory of Programming, somewhat corresponding to the two tracks

in which the journal Theoretical Computer Science had been split in 1991. The field was growing both in terms of number of papers and number of results published in the world (in particular the field of algorithms and complexity was exploding and attracting a growing number of researchers), but also, most important, in terms of subfields that needed to be addressed and understood and were getting the theoreticians involved. From the original subjects that dominated the scene in 1972 the years Eighties and Nineties saw a shifting interest toward new topics (just to make a few examples: approximation algorithms, on-line algorithms, dynamic data structures, parameterized complexity, algorithmic game theory, parallel and distributed systems, database theory, a variety of approaches to semantics of programs, program logics, etc.). Always with this spirit in mind in 2005 the program was split in three tracks by adding a Track C to the traditional two tracks A and B. The idea was that Track C might have been devoted, from time to time, to some hot topic that could not be completely hosted in the other two tracks. Since 2005 till 2008 Track C has been devoted to Foundations of Security and Cryptography. In subsequent years, since 2009 till 2019 Track C has been devoted to Foundations of Networked Computation: Models, Algorithms and Information Management. Other reasons that had great impact on the success of ICALP has been the periodic co-location with other top conferences (e.g. LICS) and the organization of satellite events, in some cases, at least at the beginning, intended to present results of research projects (e.g. ALGOSENSORS) in other cases devoted to emerging research topics (e.g. temporal graphs, real-time systems, quantum computational complexity, etc.). Finally it is important to remember that since 2016 the ICALP Proceedings, that previously were published by Springer in the ARCoSS subline of Lecture Notes in Computer Science, are published open access in the Leibniz International Proceedings in Informatics (LIPIcs) series in cooperation with Schloss Dagstuhl – Leibniz Center for Informatics. This is a very important contribution that not only provides a service to the international theory of computing community but also gives ICALP (and EATCS) worldwide visibility.

Bulletin. From the first two issues (edited by Maurice Nivat and by Giorgio Ausiello respectively) that were only twenty or thirty pages long, with the energetic role of the subsequent editors Hermann Maurer and Grzegorz Rozenberg the Bulletin became a fundamental vehicle of information for the community, reaching often the size of 400-500 pages full of informal notes, open problems, news from research centers, conference reports and especially the 'columns'. In the last thirty years the enrichment and updating of the Bulletin with new features (from abstracts of PhD theses, book reviews, and entertaining contributions, down to the most recent 'Interview', 'Viewpoint', 'Theory Blogs' Columns) has been constantly pursued. At the same time the most important step to strengthen the role of the Bulletin has been the decision taken by Luca Aceto (President) and Kazuo

Iwama (Bulletin Editor) about eight years ago to give open access to the Bulletin to the entire world theory community. This farsighted decision is again an important scientific service provided free of charge by EATCS. From this point of view it is paradigmatic the success of some Bulletin issues that have been downloaded 10.000 or even 17.000 times like issues 81 of 2003 and issue 84 of 2004.

Awards. Finally the Awards. EATCS members can be proud to contribute to an impressive series of prizes and recognitions with which the Association rewards excellence of research work of theoretical computer scientists from all around the world in a wide variety of fields. The first initiative of this kind has been taken jointly with the ACM Special Interest Group on Algorithms and Computation Theory (SIGACT) almost 30 years ago, in 1993, the Gödel Prize for 'outstanding papers in the area of theoretical computer science' (in particular "recent' results that have not appeared more than 13 years before the year of the award). The Prize is presented alternately at ICALP and at STOC. In 2022 it will be delivered at ICALP in Paris and the Committee, chaired by Samson Abramsky, consists, as usual, of three members indicated by SIGACT and three indicated by EATCS. Until now 77 scientists have received the **Gödel Prize** for the relevance and excellence of their papers. Taking into consideration the need to broaden the fields of theoretical computer science that deserved a recognition for relevant theoretical results, since the year 2000 EATCS decided to collaborate again with ACM to reward with the **Dijkstra Prize** outstanding papers on principles of distributed computing, 'with evident significance and impact on the theory and/or practice of distributed computing'. The Prize, sponsored jointly by the ACM Symposium on Principles of Distributed Computing (PODC) and the EATCS Symposium on Distributed Computing (DISC) is presented alternately at each of the two conferences. Until now 53 scientists have received the prize, sometimes repeatedly (Leslie Lamport has been awarded three times with this prize). For the same reason in 2015 EATCS and the ACM Special Interest Group for Logic and Computation (SIGLOG) established an annual award, called the **Alonzo Church Award** for 'outstanding contributions to logic and computation', in collaboration with the European Association for Computer Science Logic (EACSL), and the Kurt Gödel Society (KGS). In this case the contribution is required to have established evidence of lasting impact and depth over a time span of 25 years. To a more focused, but not less important, research field EATCS has decided to devote the **IPEC Nerode Prize** 'for outstanding papers in the area of multivariate algorithmics'. Since 2013 the prize is presented annually at IPEC (International Symposium on Parameterized and Exact Computation) and until now it has been awarded to 35 scientists. Beside all the prizes that we have introduced until now, that as we saw are meant to reward specific papers and specific results achieved in various domains, we have now to mention, of course, what we might call the flagship of

the EATCS award initiatives, the **EATCS Distinguished Achievements Award** that was presented for the first time in the year 2000 and that is aimed at acknowledging 'extensive and widely recognized contributions to theoretical computer science over a life long scientific career'. The name of the award reflects that it represents the highest recognition that the EATCS community expresses with respect to outstanding figures whose vision, work, and scientific results have inspired and shaped European and worldwide theoretical computer science. Finally, since all the mentioned awards tend to recognize the work of mature researchers, another important decision was taken by EATCS in 2009: to create an award aimed at providing a recognition for the research work of young scientists: the **Presburger Award** 'for outstanding contributions in theoretical computer science, documented by a published paper or a series of published papers'. The name of the award refers to Mojzesz Presburger who accomplished his fundamental research work on decidability of the theory of addition as a student in 1929. Since 2010 till today the prize has been awarded to 14 young researchers. To conclude the list of these important contributions of EATCS to reward excellence of research work we have to mention the **Best Paper Awards** and the **Best Student Paper Awards** that are annually presented at conferences sponsored by EATCS (ICALP in primis, of course, but also ESA, ETAPS, MFCS etc.). As most scientific associations EATCS has also started, in 2014, the **EATCS Fellows Program** to recognize 'outstanding EATCS Members for their scientific achievements in the field of theoretical computer science and for their intellectual and organizational leadership within the EATCS community'. Until now the status of EATCS Fellow has been awarded to about 40 EATCS members.

* * * * *

In conclusion we think that in order to celebrate EATCS 50th Anniversary the most important aspect to underline is how, during its life, thanks to the constant participation and contribution of its members, the Association has operated to promote the relevance of foundations of computer science, to support excellence of research work, and to disseminate scientific results to a very broad community of researchers. As we have noted, by means of a variety of initiatives EATCS has played and continues to play a fundamental role not only for European theoreticians but for the whole theoretical computer science world community.

Long life to EATCS!

Silver Jubilee of EATCS

Call for Nominations

Deadline: 15 February 2022

The Foundation

25 years ago ended an intensive discussion among some theoritical computer scientists from several West European Countries.The main persons inmolved (who later became the foundation members) were

- Giorgio Ausiello / Italy
- Jacobus de Bakker / Netherlands
- Maurice Nivat / France
- Michael Paterson / Great Britain
- Manfred Paul / Federal Republic of Germany Michel Sintzoff / Belgium
- Leo Verbeek / Netherlands.

Their essential concerns were concentrated on the question how to support Theoretical Computer Science with respect to research and education, which means

- how to improve and to accelerate the exchange of ideas and results in research,
- how to establish a closer cooperation between scientists interested in theoretical informatics,
- how to influence research programs and education curricula.

It was clear that the creation of a European association of theoretical computer scientists could be very helpful to reach these goals. but how should this association look like? Should it become an umbrella organization to the already existing national computer science societies with a limited number of representatives, or would it be more effective to have individual membership. But the main question was how to get financial support for the aims of this European association. The first address to ask for money seemed to be the Commission of the European Communities in Brussels.

One question concerning the association did obviously never arise, namely which language to use officially - from the very beginning on the association was bilingual with English and French. The end of these discussions was the beginning of the Association Européenne d'Informatique Théorique (AEIT) / European Association for Theoretical Computer Science (EATCS).

An authentic description of this very beginning was given by Maurice Nivat in 1972; here are some parts of it:

„un point d'histoire: c'est en Janvier 1972 à Bruxelles, dans les bâtiments de la Commission des Communautés Européennes, et à l'instigation de celle-ci que se sont réunis un certain nombre d'informaticiens, connus pour la nature assez théorique de leurs travaux. M. Caracciolo présidait cette réunion. Un texte élaboré par les membres de l'Université de Paris VII présents à cette réunion fut présenté à cette occasion pour tenter de définir ce que l'on peut appeler l'Informatique Théorique, ... Les principaux sous-chapitres, mais cette liste n'est pas limitative au contraire, en seraient
- la théorie des algorithmes et de leur complexité
- la théorie des automates et des langages formels
- la théorie de la programmation (sémantique formelle des langages de programmation).
En même temps fut proposé la création d'une Association Européenne qui permettrait de favoriser le développement de cette discipline ... Le cadre juridique permettant un tel regroupement a été trouvé dans la loi belge régissant les associatons scientifiques et ceci explique comment l'accord entre les partipants de la réunion initiale a abouti après quelques mois d'efforts de notre premier Président provisoire, Michel Sintzoff, au dépôt, auprès des autorités belges, des statuts d'une assocation".

An English version might be:
A bit of history: It was in January 1972 in Brussels, in the buildings of the Commission of the European Communities, and instigated by it, that some informaticians, known for the rather theoretical nature of their work, got together. Mister Caracciolo di Forino presided the meeting. A text, elaborated by the members of the University Paris VII which were present at the meeting, was presented at this

occasion in order to try to define what one could call Theoretical Informatics... The principal parts of it, but this list is not limitative, just the opposite, would be - the theory of algorithms and their complexity - the theory of automata and of formal languages - the theory of programming (formal semantics of programming languages).

At the same time the creation of a European Association was proposed which would allow to favour the development of this discipline... The legal frame for such a grouping had been found in the Belgian law regulating the scientific associations, and this explains how the agreement between the participants of the initial meeting has finally lead, after several month of efforts of our first provisional President, Michel Sintzoff, to the submission of the statutes of an association to the Belgian authorities".

The submission date, June 24, 1972, is the date of constitution of EATCS. On that day the foundation members agreed on the statutes and rules and elected the first officers of the association. Leo Verbeek became President, Michel Sintzoff treasurer; and it was fixed that the legal seat of EATCS, which had to be a place in Belgium is the home of the treasurer.

The statutes of EATCS, originally written in French, were approved by the Belgian Minister of Justice, and then the Belgian King Baudoin accorded the status of legal person to EATCS on September 4, 1972.

The Rise

The first, but clandestine, appearance of AEIT / EATCS took place from July 3 to 7, 1972 - i.e. immediately after its constitution and even before its official approvement - it was at Paris/Rocquencourt by a colloquium on „Automata, Languages and Programming" organized by IRIA (Institut de Recherche d'Informatique et d'Automatique). The idea of this symposium was originated by M. Nivat, L. Nolin and M.P. Schützenberger. This was the first ICALP! It was sponsored by SIGACT, the proceedings were published in 1973 by North-Holland Publishing Company. It was sponsored by SIGACT, the proceedings were published in 1973 by North-Holland Publishing Company.On September 6, 1972 - two days after AEIT / EATCS became a legal person - Maurice Nivat wrote a letter to several persons whom the founders wanted to become members. Already at that early date it was decided to invite Israeli to this European association, but it was only discussed whether to solicite from the beginning on persons from Eastern European countries. It was quite a long list of potential members: 8 from the UK, 7 from West Germany, 6 from France, from Israel and from Italy, 3 from the Netherlands and from Scandinavia, 2 from Switzerland and 1 from Austria.

Many of these persons came to Warwick on March 24-25, 1973, where the first general assembly and the first council together with an informal scientific meeting took place, organized by Mike Paterson at his university. Main topics

were the aims of AEIT / EATCS, the relations to existing computer societies, and how far membership should be open. It was decided to

- have an informal news bulletin
- support informal working conferences
- organize regular formal conferences, like that at IRIA in 1972.

Finally there were elections: the council, consisting, up to now, of the foundation members only, was enlarged by C. Böhm (I), W. Brauer (D), B. Mayoh (DK), R. Milner (UK), J.F. Perrot (F), D. Scott (UK) while M. Paul retired. Maurice Nivat became president, J. de Bakker and M. Paterson vice presidents, M. Sintzoff stayed as treasurer and B. Mayoh became secretary. (Bulletin no. 1).

The second council meeting was at Hamburg University, it had been organized by W. Brauer on Sunday, October 7, 1973 the day before and at the place where the 3rd annual conference of the German informatics society (GI) took place, such that council members were able to give talks at the conference or to participate in a panel discussion on „What point is there to formal semantics?".

The main topics, discussions and decisions taken at the council were (Bulletin no. 1)

- to accept members from other non-European countries,
- that Maurice Nivat took the responsibility of editing and publishing the first issue of a Bulletin with the support of IRIA; it appeared under the date of December 1973,
- that a presentation of AEIT / EATCS should be sent with a covering letter to the science
- academies of the East European countries to enlist their cooperation.
- that SIGACT and GI should be informed about the activities of AEIT / EATCS.

Three new council members were elected: E. Engeler (CH), Z. Manna (IL), A. Salomaa (SF).

From the beginning of 1974 till the end of 1976 AEIT / EATCS was visible mainly through conference activities; be it by sponsoring the Advanced Course on the Foundation of Computer Science in Amsterdam 1974, the conference on l- calculus in Rome 1975 or its own International Colloquia on Automata, Languages and Programming. The second ICALP took place in Saarbrücken in summer 1974 with the proceedings published for the first time in the series Lecture

Notes in Computer Science by Springer-Verlag; the third one in Edinburgh in July 1976, its proceedings appeared at the Edinburgh University Press.
During this ICALP an informal EATCS meeting was arranged; president M. Nivat had had an unfortunate car accident such that vice president M. Paterson had to chair. It was suggested that there should be a reduced ICALP registration fee for members, the amount of the reduction being comparable to the membership subscription; further it was decided to solicit membership among colloquium attendees (and not only by invitation), the annual dues could be paid at the colloquium registration desk. G. Ausiello agreed to edit the EATCS Bulletin for at least one year. (Bulletin no. 2)

With the help of the Instituto di Automatica, Rome the second issue of the bulletin appeared in December 1976. From that time on the great annual events of EATCS are the ICALPs and the Bulletins.

In 1977, again in July, there was the 4th ICALP, again in another country (this alternation became an aim of EATCS), namely in Turku, Finland. From now on, according to a formal agreement, the Springer-Verlag, Heidelberg published the ICALP proceedings as volumes of the Lecture Notes in Computer Science.

The second general assembly was run during ICALP'77. Mike Paterson became president, M. Nivat and J. de Bakker, vice presidents and the new council members G. Rozenberg (then in Antwerp) and H. Maurer (Karlsruhe, FRG) agreed to serve as treasurer and as secretary plus bulletin editor, respectively. From now on an EATCS general assembly took place at each ICALP, and almost always on the Tuesday of the ICALP week.

In 1977 the 3rd bulletin appeared in October; its production was supported by the university of Karlsruhe; this was the first of the many „Maurer Bulletins".

One may say that in 1977 the character of EATCS was determined and became visible; or to cite president M. Paterson. (Bulletin no. 3, p.1):

„What is accomplished so far? EATCS is a substantial international body of academics and researchers which can already claim to represent those in Europe with a declared interest in theoretical computer science. Independent of any massive parent organization, it has reached a stature adequate to attain many of our original goals. One realized aim, with benefits now taken for granted, is the establishment of an orderly sequence of international colloquia in Europe. By the formal sponsorship of other meetings we may avoid the unforeseen clash of independently arranged events, too similar in time and subject."

Structure and Management

Now, at the age of five, EATCS had got a personality - to a certain extend at least.

It is common human attitude to look at and to treat a child differently before and after entering school. Little children are carefully watched more or less permanently and their development is recorded in detail. School kids, however, are

more autonomous, have a more complex behaviour, and a much broader spectrum of interests, therefore one studies their different types of activities and aspects of their personalities.

Similarly we now consider specific features of EATCS.

Statutes and Rules

The first statutes and rules in their original French version were never published; only in January 1980 a little sloppy English translation was printed in EATCS Bulletin no.10 in order to modify them. They originally defined an association which is quite different from the actual EATCS, although the general goals, expressed in article 2 of the statutes, seem to resemble the current ones. The main differences lie in the membership concept. The AEIT / EATCS was conceived as a very small group of scientists, who could be individual members or people representing institutional members (article 3). New members could only be invited by the council (article 4), and according to a complicated procedure specified by the rules; for example an individual member could only propose one new member per year and had to find two supporting members for such a proposal; per year not more than 10 individual members could be accepted; all members had to be consulted about a proposal, and it was accepted only if at least 50 % of the answers were positive (articles 1 and 2 of the rules). Also, the statutes set the quorum for the general assembly to 50 % of all indivial member votes and 50 % of the institutional member votes where one member could only represent up to two other members (articles 8 and 11).

The general assembly had to take place only every three years; in the mean time the council had to direct and manage the association (articles 12 and 14). The council (7 to 15 members) had to be elected by the general assembly on the basis of candidatures signed by at least 6 members, subject to the restriction that one member could sign only for one candidature (article 7 of the rules). On the other hand nothing was said about the procedure for electing the board. It was stated however, that the president could be reelected only once (article 13 of the statutes).

Moreover it was obviously envisaged to have a secretariat with staff members, since article 14.2 of the statutes reads: „Le secrétaire général assure la gestion courante... Il nomme et révoque le personnel de secrétariat"; i.e. „The secretary general is charged with the daily management.... he appoints and dismisses the staff of the secretariat". EATCS developed rather soon differently from what the founders had envisaged: on the one hand the close contact with and the money from the European Commission did not come, on the other hand the new members invited after the installment of EATCS were not aware of the statutes and rules and therefore from the first general assembly on drove the association into the direction of an open society of individual members.

That the gap between the theoretical concept and the practical realization of EATCS was very large and could create real problems was not seen or ignored for some time. It was Arto Salomaa who immediately after being elected as president stated in his first and second Letter from the President (Bulletin no. 9, 10): „it is now literally impossible to run things exactly according to the original statutes" and even more „ If you read carefully through the old Statutes and Rules, you must realize that practically everything we have been doing has been illegal". And he acted consequently. Together with H. Maurer, M. Paterson and G. Rozenberg he proposed a drastically modified version of the statutes in Bulletin no. 10, with the following commentary:

„Basically, the modified version proposed below has been obtained by
(i) getting rid of the obsolete apparatus but still preserving the things necessary for the „Belgian" aspect;
(ii) removing restrictions on membership;
(iii) implementing explicitly the central functions of EATCS: ICALP and the Bulletin;
(iv) reducing certain things, such as the quorum for the General Assembly, to a practical level;
(v) giving more explicit rules for the election of the council"

There was no modified version of the rules because the essentail points of the „old" rules had been implemented in the „new" statutes.

With minor changes this proposal was accepted „without dissentient votes" by the general assembly 1980 (see Bulletin no. 12), and in 1986, in Bulletin no. 30 (p.4), A. Salomaa could state: „Perhaps a future historian can find out ... whether or not the ratification of the new statutes at the ICALP'80 was done legally. Anyway, after that nobody questioned the legal status of EATCS". Nevertheless it was felt in 1988 that the statutes should be streamlined even more. Burkhard Monien, Secretary of EATCS wrote in Bulletin no. 40: „On its meeting during ICALP'88 in Tampere, Finland, the council formed the statutes committee consisting of W. Brauer, B. Monien, A. Salomaa and P. Turakainen (under the chairmanship of A. Salomaa) whose task was to propose a revision of the statutes. The statutes committee has presented its suggestion at the concil meeting during ICALP'89 in Stresa, Italy. The council formulated then a proposal for the new statutes which is given below. The main changes consist of stating the main functions of EATCS more explicitly in Article 2, and removing unrealistic requirements and unnecessary complications from the statutes.

EATCS members are asked to vote on the statutes by sending back the voting form enclosed in this issue...".

In Bulletin no. 41 B. Monien reported that 166 of the 172 postal votes obtained were in favor such that the new statutes are in force since June 1990. Important new aspects of the statutes are the introduction of a postal referendum for statutes modification purposes, the restriction to 5 of the number of proxies a general assembly member may use in voting, the dropping of the upper bound for the number of council members and of the terms of office of the president, the denial of EATCS's „liability for any activity carried out partly or wholely on its behalf" and the legalization of the enlargement of the board by the editors of the TCS journal and the EATCS Monograph series (in addition to the editor of the EATCS Bulletin who legally belongs to the board since 1980). That the past presidents since 1973 are also board members is covered by the statement that „other members, at the discretion of the council" may be board members (article 10f).

Due to the continuous efforts of A. Salomaa EATCS now has statutes which proved to be reasonable and practical. EATCS owes a lot to Arto Salomaa.

EATCS officers

EATCS as a non-profit organization is run by scientists only on a honorary basis, they all do voluntary work. EATCS is directed and managed by the council, which itself is controlled by the general assembly, i.e. the GA has to ratify the decissions of the council.

Major concerns of the council are to discuss proposals, initiate activities (like the Monograph series) and to decide about cooperations with other organizations (i.e. SIGACT). Its main activity consists in planning and monitoring the „International Colloquia on Automata, Languages and Programming" (ICALP):

- to give (or to change) guidelines for the organization of an ICALP (about structure, finances, program committee work)
- to get proposals for hosting ICALPs and to preselect the site.

The council prepares the GA, it meets before and in case there had been elections, after the GA.

According to the statutes, the term of office of a council member is three years. The members are proposed by the council to the general assembly, thereby it is taken into account that the composition of the council mirrors the geographical and scientific distribution of the members of EATCS.

From its beginning on EATCS treated Israel like a European country, therefore it was natural to have an Israelian council member as soon as possible: it was Zohar Manna in 1976. But already in 1977 the first „non-European" became council

member: Ron Book from Santa Barbara, USA, followed in 1979 by the first Canadian: Derick Wood, and in 1982 with Masako Takahashi the first Japanse joined the council. East Europeans however were elected to the council only in 1985: F. Gécseg from Szeged and J. Gruska from Bratislava.

Besides the intention of having an international council there is the need to have a Belgian member in the council because EATCS has its official seat in Belgium.

Additionally, since 1989 the chairperson of SIGACT is council member, vice versa the EATCS president is council member of SIGACT.

The council started in 1972 with seven members, the founders of EATCS. It was in 1973 enlarged to 15 people (the maximum allowed by the statutes). From 1977 on there were always 20 or more members (this was legalized by the 1980 statutes). The maximum number of 29 members was reached in 1991.

According to the first and second statutes the council had to elect the board out of its members - This was changed in 1990: Now the board members need not be council members. The board originally consisted of the president, two vice presidents, the treasurer and the secretary; since 1976 the bulletin editor was considered member of the board.

Obviously the council had decided, maybe in 1977, to have the past presidents as ex-officio members of the board. But astonishingly L. Verbeek has never been listet as past president; Verbeek served as council member until 1979. Other ex-officio members of the board are the TCS editor (listed for the first time in June 1983), who from the beginning on was and still is Maurice Nivat, and since 1991 the Monographs editors W. Brauer, G. Rozenberg and A. Salomaa. The members of the board are responsible for the day-today management of ETACS. It seems that they have never met and decided as a seperate organ; the board obviously conceived itself always as part of the council.

Changes in the board of EATCS

Time	President	Vice President(s)	Treasurer	Secretary	Bulletin Editor
till June 72	(provisional) Sintzoff				
since June 72	Verbeek	Paterson, M. Paul	Sintzoff	Nivat	
March 73	Nivat	de Bakker, Paterson	Sintzoff	Mayoh	Nivat
July 76	Nivat	de Bakker, Paterson	Sintzoff	Mayoh	Ausiello
July 77	Paterson	de Bakker, Nivat	Rozenberg	Maurer	Maurer
July 78	Paterson	de Bakker, Nivat, Sintzoff	Rozenberg	Maurer	Maurer
July 79	Salomaa	de Bakker, Nivat	Rozenberg	Ottmann	Maurer
July 81	Salomaa	Nivat	Paredaens	Ottmann	Rozenberg
July 82	Salomaa	Rozenberg	Paredaens	Ottmann	Rozenberg
July 85	Rozenberg	Brauer	Paredaens	Ottmann	Rozenberg
July 86	Rozenberg	Brauer	Paredaens	Monien	Rozenberg
July 88	Rozenberg	Brauer	Janssens	Monien	Rozenberg
July 91	Rozenberg	Brauer, Gécseg	Janssens	Monien	Rozenberg
July 94	Brauer	Díaz, Gécseg	Janssens	Monien	Rozenberg
July 95	Brauer	Díaz, Gécseg	Janssens	Rovan	Rozenberg
July 97 New elections will take place and will lead to further changes					

For more information see the statistics by M. Kudlek in Bulletin no. 52 (pp. 116, 117).

Membership

Created by European scientists as a European association AEIT / EATCS almost immediately became a worldwide organization; there has obviously been a big need all over in the informatics community to have such an association, which not only enables the European theoreticians to inform each other more quickly and through this to have more effective cooperations but also serves as a partner for non-Europeans who want to know more about and to get into contacts with Europeans working in the same field. EATCS had the right aims at the right time and realized them with the right means, namely with a series of colloquia , the ICALPs, and with a bulletin. After the EATCS Bulletin started to appear regularly three times per year, the membership nearly increased by a factor of 4 within a year.

Membership Development

Year	Total	Europe (West/East)	Israel	USA	Canada	Latin America	Japan	Far and Near East	Australia New Zealand	Africa
1972	7									
1973	ca 50									
1976	100	92	3	4	1					
1978	343									
Feb. 1979	458			109						
Oct. 1979	526	354	17	111	16	5	11	2	9	1
July 1980	590			153			(+Israel) 45		9	1
July 1981	603	357/28		148		6	(+Israel) 53		9	2
July 1982	538	303/28	17	114	14	6	38	8	9	1
July 1983	618	336/61	18	120	17	8	40	7	9	2
July 1984	617	386	19	131	21	7	33	8	10	2
July 1985	642	387	17	155	17	6	37	9	13	1
July 1986	709	433	20	173	17	6	36	10	12	2
July 1987	881	529	18	237	21	4	46	8	16	2
July 1988	1240	829	17	253	36	6	56	23	18	2
July 1989	1538	1086	18	295	41	6	55	19	16	2
July 1990	1692	1199	14	322	44	6	54	35	16	2
July 1991	1879	1352	18	324	56	7	57	42	17	6
July 1992	1826	1329	17	299	56	7	57	40	17	4
July 1993	1730	1273	15	250	56	7	63	39	19	8
July 1994	1608	1194	12	210	50	9	70	40	17	6
July 1995	1609	1210	31	178	42	9	80	45	8	6
July 1996	1488	1174	11	136	34	6	73	35	13	6

Until 1991 EATCS was a fast growing society. In 1992 a decrease in membership began, which unfortunately continues and seems to have the same speed as the growth had. This development has to be seen in the frame of the general decline in economy. In many countries it became much harder for scientists to get their expenses (or at least part of it) for participation in an ICALP payed; and even to get the permission to leave the job or project for a stay at ICALP is not always easy. As we know, it is not only EATCS which suffers from the harder constraints scientists are faced with. It is a particular problem that the membership from the US has drastically declined to now less than half of the number of 1991 - this happens after the USA members had for a long time formed the largest national group in EATCS. From July 85 till July 90 they were outdoing the Germans who from an early time on were by far the largest member contingent.

Membership Fee and Sponsoring

An association like EATCS is not only proud of its many members, it really needs

them because of the fees which at the beginning constitute its only income.

In 1986 president G. Rozenberg succeded in his efforts to get institutional sponsors for EATCS; the first two sponsors were: BULL France and the Danish Datamatics Centre, Lyngby (Bulletin no. 28, p. 2), and by the end of the same year joined IBM Belgium and Philips Research Lab., Eindhoven, The Netherlands, of which still Philips continues as sponsor - for 12 years already! Other long lasting sponsors have been BULL (9 years) and still are Siemens ZTI, München, Germany, Nixdorf Computer AG (now Siemens-Nixdorf) Paderborn, Germany (both 10 years) and IASI- GNR, Rome (9 years). At the moment EATCS has nine institutional sponsors, some even from outside Europe: PWS Publishing Company, Boston, USA and UNU / IIST UN University, International Institute for Software Technology, Macau. Institutional sponsors may present themselves, their activities, aims and scopes in the Bulletin, where their names are listed regularly. EATCS owes not only a strong financial support to its institutional sponsors but also gained recognition as an association worth to be sustained.

At the beginning of AEIT / EATCS members were often urged to pay their fee. In the first Bulletin (p. 4) under the date of November 12, 1973 secretary Brian Mayoh complained - in a very polite way: „As the list of those who have returned the application forms to the secretary does not coincide with the list of those who have paid their fee to the treasurer, there is some confusion". And this confusion continued. The fact that the annual fees were not regularly payed was neither due to their exorbitant amount - from 1972 till June 1976 they were only 100 Belgium Francs (which was equivalent to 2,70 US$ or 7 DM), in July 76 they were raised to 2 British Pounds (or 3,30 US$ or 8 DM) and from July 77 till July 81 they amounted to 5 US$ per year (i.e. 11,50 DM in 1977 and only 9 DM in 1980) - nor to the lazynes of the EATCS members but it was due to the time- consuming and costly procedure to send the money from one country to another, be it by cash, cheque or money order.

The fee increased slowly from US$ 5 via 8 (from July 1981 to July 1982) to 10 (until March 1990). And the problems with sending the money to the Belgium bank account increased also: In June 1984 treasurer Paredaeans warned that fees can be paid by bank cheques or cash but „anyhow a member cannot pay by International Post Money Order" (Bulletin no. 23, p.2). Little later, in October 1985, it became more complicated, treasurer Paredaens wrote in Bulletin no. 27, p.3: „If the transfer is in US$ then the annual membership payment equals US$ 10. If the transfer is in a currency other than US$ then the annual membership payment must be equivalent to US$ 12 (the difference is used to cover the bank charges) ...".

In 1990 the last but one change with respect to membership fee was decided, treasurer D. Janssens wrote in Bulletin no. 40, p. 8: „The agreed amount was the equivalent (at that time) of 30 DM (German marks): US$ 10. However, since 1972

the conversion rate of the US$ towards the DM has decreased quite considerably, in particular during the last five years. In the meantime, the cost of printing and mailing the Bulletin has increased. For these reasons the council of EATCS has decided at its meeting of July 13, 1989 in Stresa, Italy, that from March 1, 1990 the EATCS membership fee will be DM 30. .... Those members, in particular outside Europe, who wish to continue paying their fee in US$, can evidently do so; hence they should pay the amount equivalent to DM 30, which at present is US$ 18". Since 1996 the fee had to be increased to 45 DM, but now the membership dues can be payed by using credit cards.

Before this very elegant solution could be achieved the EATCS council had tried to facilitate the payment for the members in various ways. At first the members were asked to pay their fees for some years in advance to lower the total bank charges; this was not unfair because of the small amount of the annual fee. The second possibility to avoid high bank charges was to have the EATCS membership fee payed together with the ICALP conference fee. This was very helpful for the quantity of members; but through this procedure the number of EATCS members became strongly dependent on the number of ICALP participants. The third idea was to build national groups which collect the EATCS fees of members of one country and transfer the total sum to the EATCS treasurer. It seems that this gave an impetus to set up the Chapters.

Another challenge to EATCS was that theoretical computer scientists from East European countries did not only have problems with bank charges but that most of them were unable to get hard currency for paying the membership fee, but they would have very much liked to have access to the Bulletin or even to become member of EATCS. Therefore it was a good idea by G. Rozenberg and excellent initiative by Dines Bjorner, Denmark when he wrote on August 12, 1987 to the EATCS members: „With this letter I would like to invite you to become a sponsor of 'The EATCS- Membership and - Bulletin Subscription Fund'. The purpose of this fund is to support a number of colleagues in „foreign currency constrained" countries, either in their becoming members of the EATCS, or in obtaining a subscription of the EATCS Bulletin. Sponsorship will cost you any multiple of the EATCS membership fee (currently 10 US$).

... Some colleagues may be able to receive a subscription, but not become a member of the EATCS - oftentimes the latter requires cumbersome and problematic approval procedures from public authorities, usually ending with refusals..."

It seems that this letter was quite successful, however the list of donors and receptors has been and still is kept confidential. In March 1994 Dan Simpson, Brighton, UK who had succeeded D. Bjorner who had moved to Macau, informed EATCS members that „Last year, thanks to the generosity of EATCS members we were able to offer well over 100 individuals in currency constrained countries full membership benefits of EATCS. I can assure you that your efforts are fully

appreciated. ..." (Bulletin no. 53, p. 35).

The fund still exists because some former East European countries continue to suffer from financial difficulties.

The Chapters

Mainly due to the initiative of president G. Rozenberg two chapters have been founded within EATCS, the Italian and the French chapter. Both of them have in common, that

- they share the same aims with EATCS,
- they operate within the same field of activity,
- a member of a chapter is automatically member of EATCS,
- a chapter cooperates with other organizations of it own country,
- a chapter runs national conferences, workshops and schools on theoretical computer science,
- reports and contributions of the chapters are regularly published in the EATCS Bulletin.

The main difference between the two chapers consists in their statutes. While the EATCS statutes are binding for the Italian chapter, the French chapter has modeled its statutes according to French law. It therefore is an autonomous organization with its own name: Association Francaise d'Informatique Théorique (AFIT); that it is mainly the French Chapter of EATC is due to an extra agreement between EATCS and AFIT. The Italian Chapter of EATCS was founded on July 3, 1987 after preparatory work had been done by Bruno Apolloni, Aldo De Luca and one of the founders of EATCS, Giorgio Ausiello. Alberto Bertoni, Milano became the first president; the actual president is Giancarlo Mauri, Milano.

AFIT, was founded, on March 14, 1988 here, too a founder of EATCS was involved: Maurice Nivat who was chosen as first president of AFIT - and still is. The first secretary was Brigitte Rozoy, Universite de Caen followed in 1995 by Ph. Schoebelen, ENS Cachan.

EATCS has profited in two ways from the creation of the chapters: it got more members and its European bases became stronger.

Cooperations

Informatics has two roots - the theory of computing (initiated mainly by Herbrand, Gödel, Kleene, Church, Post and Markov) and the construction and programming of computers. Although the theory of computing is not only older than the first

functioning computers but also has influenced the development of hardware and of programming tools from the very beginning on, in the public opinion (i.e. by most people who are not theoretical informaticians) theoretical informatics is considered as a secondary issue, usually as an a posteriori reflection on the inventions of hard- and software engineers. What people see and use are hard- and software and not the theory behind.

This situation is mirrored within the spectrum of computer and informatics societies. In many countries theoretical computer scientists were in former times not very well represented in the computing-related societies. This has for sure been a main reason for EATCS to find such a positive acceptance. Notable exceptions - and therefore potential cooperation partners - were ACM, IEEE Computer Society and GI (the West German informatics society) which, already rather early had special committees or interest groups on theory. In addition to these, some later established groupings like BCS-FACS, IFIP-SGFCS and EACSL became „sister organizations". Basically the benefit of these cooperations consists in a systematic and detailed information about each other and the respective activities, in particular conference coordination and sponsoring.

Unfortunately no formal relationships to the organizing bodies of the East European conference series MFCS and FCT were possible because of political reasons - via informal and personal contacts however informations and reports about these conferences were published regularly in the Bulletin.

ACM-SIGACT

SIGACT, the ACM Special Interest Group for Automata and Computability Theory, which was founded in 1969 has been a model for EATCS. Their aims and activities are rather similar. A decisive diffference is that EATCS is not an offspring of a mother organization like ACM, thus it did and does not get such a basic support as SIGACT obtains from ACM.

The major difference, however is due to the geographical region in which their main conference activities take place. To cite D.S. Johnson (from the first SIGACT Chairman's Column in the EATCS Bulletin, no. 34, p. 27/28, February 1988): „SIGACT is an international organization, with roughly a third of its 2000 members coming from Europe and the Far East. The bulk of its members and most of its activities, however, are centered in North America ... SIGACT is best known for the conferences it sponsors, in particular the annual „ACM Symposium on Theory of Computing" or „STOC" conference ... This and the mirror-image „FOCS" conference („Symposium on Foundations of Computer Science") held in the fall, are the two main general-interest theory conferences in North America analogous to EATC's ICALP" (Indeed, they were initially models for the creation of ICALP).

„SIGACT sponsors only STOC, not FOCS. The latter job is taken by a second

organization, the IEEE Computer Society Technical Committee on Mathematical Foundations of Computing, or simply „TCMFC" ... Although SIGACT and TCMFC share the same „turf" so to speak, they are not rivals. In practise they act mostly like one organization with two overlapping heads. These two „heads", the Executive committees of the two organizations, provide a useful method for dividing the responsibility for putting on conferences. Their intersection includes the Chairs of the two organizations as well as a Conference Site Coordinator and, to improve cooperation with Europe, the EATCS chair. TCMFC is the older of the two organizations, having been in existence under various names since 1960, when the first FOCS conference was held, although it was then called a „Symposium on Switching Circuit Theory and Logical Design" ... Besides the conferences we sponsor, our most visible product is SIGACT News. This publication in effect serves as a joint SIGACT / TCMFS newsletter... It is not presently, however, as substantial and useful a publication as is the EATCS Bulletin, and we are working hard to improve it". The contents lists of the SIGACT News are published in the EATCS Bulletin since 1988.

The cooperation between the three organizations which began with the first ICALP and was intensified by a proposal of Ron Book (Bulletin no. 6, p. 3) for exchanging more information and mutual reduction of conference fees as well as by a call for contributions by the SIGACT News editor M. Blattner in Bulletin no. 18 (p. 8), includes in particular also the coordination or joint sponsoring of conferences in Europe and North America.

The most important outcome of the cooperation between SIGACT and EATCS, however, is the installment and continuous sponsoring of the Gödel Prize.

The Gödel Prize

Theoretical informaticians in Europe, or at least in Germany are more reluctant than in North America to award prizes. This may be a reason why only in June 1992 (in Bulletin no. 37) EATCS together with SIGACT announced the sponsoring of a new theory award, the Gödel Prize. The first ideas concerning it were already published by D.S. Johnson in the SIGACT Chairman's Column of Bulletin no. 37 (February 1989). He gave several good arguments for prizes: „A well-positioned prize does far more than allow an occasional researcher to pocket a check or hang a medallion on the wall: it draws positive attention to the field" and „they help certify a body of potential spokespersons for the field".

The prize is for outstanding papers on theoretical computer science in a broad sense, published by a single author or a team of authors in the recent six years. Beginning in 1994 the Gödel Prize has been accompanied by a US 5.000 award which is provided by a grant from PWS Publishers in cooperation with International Thompson Publishing (ITP).

The selection committee consists of 6 well-known scientists representing a

diversity of areas, appointed for a term of 3 years by EATCS and SIGACT - at the beginning each organization appointed 3 members, from then on each year two are leaving the committee and each organization appoints a new member.The chair is always coming from the region where the award is presented.

- The first committee, for the 1993 prize consisted of St. A. Cook, Toronto; R. A. Karp, Berkeley, R. Milner, Edinburgh, B. Monien, Paderborn, A. K. Salomaa, Turku, A. C. Yao, Princeton (chair).
- For 1994 B. Monien and A. C. Yao left and J. v. Leeuwen, Utrecht and M. Rabin, Harvard and Jerusalem came in, A. Salomaa was chairman.
- In 1995 J. Hartmanis and G. Plotkin replaced St. A. Cook and R. Milner, R. A. Karp was chairman.
- For 1996 committee was M. Rabin, D.S. Johnson and G. Rozenberg had replaced R. A. Karp and A. Salomaa; chairman was M. Rabin.

The prize is given annually at ICALP or STOC alternatingly. The first prize, for 1993, has been presented during the Federated Computing Research Conference 1993 (FCRC'93) at STOC'93 in San Diego, the following prizes were given at ICALP'94 in Jerusalem, at STOC'95 in Las Vegas and during FCRC'96 at STOC'96 in Philadelphia.And, naturally all past winners are listed in the Bulletin; for the newest list see Bulletin no. 60, p.19 (October 1996).

BCS - FACS

The „British Computer Society Specialist Group - Formal Aspects of Computing Science" (BCS-FAC) was inaugurated on March 16, 1978 with five closely related aims of which the central one seems to be: „to bring together groups from both industrial and academic departments who have interests in the formal aspects of computing science" (Bulletin no. 23, p.117). The formation of FACS was prepared by a meeting on November 30,1977, where a preparatory committee lead by Dan Simpson and John Cooke was formed.

„One of the major services offered to members of FACS is the ability to easily join EATCS in conjunction with FACS membership. About 80 % of FACS members take advantage of the scheme" (D.J.Cooke, D.Simpson, Bulletin no. 37, p.52, February 1989).

In Bulletin no. 23 and from Bulletin no. 37 on rather regularly, Dan Simpson and since Bulletin no. 58 Ann Wrightson (sometimes with the help of colleagues) present a rather broad spectrum of informations on FACS; they report on internal FACS matters and developments and on workshops, foreign guests, publications,

government research programs etc. In a way FACS can be seen as a kind of chapter of EATCS.

IFIP - SGFCS

IFIP, the International Federation for Information Processing, is a multinational federation of professional and technical organizations (or national or regional groupings) concerned with information processing and computer science. The aims of IFIP are to promote informatics and information technology by fostering international cooperation, stimulating research, development and application of informatics, furthering dissemination and exchange of information, encouraging education. IFIP was formally established in January 1960 after (similar to EATCS) the first International Conference on Information Processing (now counted as the first IFIP World Congress) which was held in Paris in June 1959 under the sponsorship of UNESCO. Its technical work is managed by 12 Technical Committees (TCs), each of which has several Working Groups. Although several of its TCs, in particular TC 2 (Software: Theory and Practise) and TC 7 (Computer System Modelling), comprise theoretical aspects the field of theoretical computer science in its entirety was not represented adequately in IFIP. Therefore, in 1989, the IFIP president-elect, B. Sendov (Bulgaria), and W. Brauer (West German General Assembly member and TC 3 chairman) invited a number of well-known theoretical computer scientists from around the world (in particular many EATCS council members) to participate in the creation of a Specialist Group on Foundations of Computer Science (SCFCS) in order to give theory a stronger position within IFIP. At the IFIP World Congress 1989 in San Francisco a well-attended preparatory meeting took place and right after that the IFIP General Assembly inaugurated the group; J. Gruska (then chairman of the theory track of the 1989 World Congress) was appointed chairman (Bulletin no. 40, p.330).

The first meeting of SGFCS took place during ICALP'90 in Warwick (Bulletin no. 42, p.1). A rather extensive report on SGFCS is given by J. Gruska in the section on sister organizations of Bulletin no. 52 (pp. 155-162).

In September 1996 the IFIP General Assembly transformed SGFCS into Technical Committee 1 (Foundations of Computer Science) - this was a great success for theory and for SGFCS chairman J. Gruska. Since January 1997 the TC 1 chairman is Giorgio Ausiello - one of the founders of EATCS! This gives a strong hope that the close cooperation between EATCS and IFIP TC 1 will continue successfully.

EACSL

The European Association for Computer Science Logic (EACSL) was founded on July 14, 1992, at the conference centre Schloss Dagstuhl, Germany „by computer scientists and logicians from 14 countries" (Bulletin no. 48, p.449, October

1992). Two of the Executive Members of EACSL are closely related to EATCS: C. Boehm (early council member from 1973 till 1982) and Y. Gurevich (current council member since 1991). EACSL promotes CSL as „an interdisciplinary field between mathematical logic and computer science... in the areas of scientific research and education", in particular by organizing the annual international conference on CSL whose proceedings (i.e. a selection of the papers presented) are published in the Springer LNCS series.

EACSL president E. Börger reports regularly in the Bulletin since 1992.

ESPRIT - Basic Research Actions

Originally, in 1982, the ESPRIT (European Strategic Programme for Research and Development in Information Technology) initiative of the European Community had the objectives: „1. To promote European industrial cooperation in precompetitive Research and Development in Information Technology. 2. To provide European IT industry with the basic technologies it needs in the early 90's. 3. To pave the way for standards." (G. Metakides, Bulletin no. 37, p.59).

Five to six years later this very strong industry-oriented position had slightly changed as G. Metakides stated in Bulletin no. 37, p.61: „The second half of this century is replete with examples of research ideas which, although conceived without even thought to applications, led to technological developments with major industrial and social impact". A result „is a growing consensus that supporting basic fundamental research in IT is a solid investment whose payback, even if it does not come in the form of short term industrial applications, will be great. Basic Research performed at universities and research institutes serves the dual role of providing new knowledge and helping to ensure the future availability of high-calibre scientists and engineers. Both of these are key elements in the long-term ability of Europe to compete in global markets."

These considerations seem to have led to the establishment of the ESPRIT Basic Research Actions. The first call for proposals was published in the Journal of the European Communities in March 1988. It is clear that now EATCS got very much interested in the topics and results of this program. On the other hand the project officers of the ESPRIT Basic Research Actions wished to have their projects as widely known as possible. Therefore at ICALP'89 in Stresa descriptions of five ESPRIT-BRA projects were presented, which then were published in EATCS Bulletin no. 39 (October 1989). From that time on reports on ESPRI-BRA projects are published in a special section of the Bulletin, and in addition to these, general reports about the ongoing of the ESPRIT-BRA program and on calls for proposals as well as their results can be found regularly in the Bulletin.

In 1994 the ESPRIT-BRA program was replaced by a „Long Term Research" concept (Bulletin no. 53, p.74, June 1994) which looks for projects „motivated by industrial needs", which „combine the long term characteristics of research with

high industrial relevance" (Bulletin no. 57, p.79, October 1995). The last report on a BRA-project appeared in Bulletin no. 59.

Scientific Publications

From its very beginning on it was one of the aims of EATCS to promote the publication of research results (article 2 of the first statutes). But it was a long and cumbersome way to reach the current position of EATC as the sponsor of four important scientific publication means: the EATCS journal „Theoretical Computer Science" published by North-Holland, the EATCS book series Monographs on Theoretical Computer Science, and the EATCS book series Texts on Theoretical Computer Science, both published by Springer, the ICALP proceedings series, as a subseries of the LNCS series of Springer.

The comparatively easiest thing was the establishment of the agreement with Springer-Verlag on the ICALP proceedings: Springer liked their quality, EATCS the ease of production and the financial support by Springer (to the proceedings editor and the EATCS treasurer); from June 1980 (Bulletin no. 11) on Springer gave also an 25 % discount on all ICALP proceedings volumes for each EATCS member.

The good cooperation with Springer also resulted, in spring 1983, in the establishment of the „EATCS Monographs in Theoretical Computer Science", which was announced by the editors (W. Brauer, G. Rozenberg, A. Salomaa) in Bulletin no. 20 (June 1983) and presented at ICALP'84 in Antwerp - already with the first 3 volumes published (and 10 more in preparation) and „quite a pleasant surprise ... the cocktail party given by Springer-Verlag before the conference dinner on July 19, on the occasion of launching the new monograph series". (cited from Jozef Gruska's report on ICALP'84 in Bulletin no. 24, p. 157).

For the first time the idea to create an EATCS Monograph series was mentioned at the council meeting at ICALP'80 in Nordwijkerhout, Netherlands (see Bulletin no. 12, p. 1) where „Ron Book presented a proposal about such a series" (Bulletin no. 20, p. 3). In the 1981 general assembly R. Book's proposal „was considered to be preferable" (Bulletin no. 15, p. 6) and he „was asked to negotiate a contract with the publisher proposed by him". It was thought, that „the final decisions on this" could „be taken by the monographs series committee within the next few months" (Bulletin no. 15, p. 7) - the committee consisted of R. Book, H. Maurer, G. Rozenberg, A. Salomaa. Unfortunately the envisaged publisher (Academic Press) seemed „to be no longer seriously interested. Thus it was decided to consult many scientific publishers in order to find an outlet for a monograph series" (Bulletin no. 18, p. 5, October 1982). Also Jan van Leeuwen was very active in this matter.

That the monographs series was a great success right from the beginning was also due to an intensive advertisement campaign by the Bulletin editor G.Rozenberg

and the EATCS president A. Salomaa, who in each Bulletin from no. 20 till no 24 mentioned in their „letters from the ..." this series. The regular announcement of the book series in the Bulletin started however only in June 1987 (Bulletin no. 32), although it was announced (and tried) already in Bulletin no. 28 (February 1986). After 10 years of success, the monographs series was split into two series, The Monographs and the Texts (see Bulletin no. 54), since it had turned out, that some of the monographs were, by the time, really used in graduate classes, and that there is also a need for advanced texts covering the field between the usual textbooks and research monographs. Needless to say, that EATCS members get a 25% discount on all books of the two series.

The EATCS journal TCS really is the child of Maurice Nivat - in spite of many difficulties he brought it up in a process of several years. Already by a letter of June 18, 1973 to the EATCS council he proposed the creation of the journal and presented a draft of an agreement with North Holland Publishing Company. In his letter he writes: „This proposal was made after discussions I had with Mr. E. Fredriksson in which we asked ourselves whether it would be possible to have our European Association play a role in this creation.

The idea we came to was that the Association could appoint say 5 members of the future Editorial Board among whom hopefully one would be the managing editor and that these 5 people would discuss with North Holland the numerous questions to be solved: other members of the editorial board, scope of the journal, etc...".

It was envisaged that the first issue of journal should appear in March 1974. In September 1973 M. Nivat reported to the council: „ The project I submitted to you has been shaped during the past months and I shall present it to you in Hamburg: a tentative list of 18 members of the editorial board has been made, seven at least of which are members of our association. The idea is now to create a truly international journal, which would not be linked with our association by any other link than the fact that some of us will sit in the editorial board".

The council in Hamburg was not so happy about the dissociation of EATCS and TCS. The minutes of the meeting say: „The report of M. Nivat on his discussions with N. Holland about a new journal in theoretical computer science was approved. This question no longer concerns the association". (Bulletin no. 1, p. 5).

But M. Nivat did not give up; in early summer 1976 he wrote to the EATCS council members: „... and 7 members of the counicl of EATCS stand on the editorial board of TCS. And TCS works well: a bad choice of the first printer delayed the publication, but a new printer has been found so that we hope to be on schedule at the end of 1976". He also announced that he would propose in the general assembly at ICALP'76 in Edinburgh „to tie together the journal TCS and the association", he had in mind that EATCS members should get a large discount on the

journal price. Because of his car accident there was no regular general assembly and in the informal meeting the discussion on his written proposal did not lead to any decision. And also in the 1977 general assembly it was decided that EATCS „should concentrate - on the publication of a Bulletin and - on sponsoring the conferences of the ICALP series". (Bulletin no. 3, p.3). The break-through came with the agreement between North-Holland and EATCS „...signed by a representative of North-Holland, the President of EATCS (M. Paterson) and the Editor-in-chief of TCS (M. Nivat)". (Bulletin no. 5, p.2,3). Main parts of this agreement (which was ratified by the EATCS general assembly in July 1978, see Bulletin no. 6, p. 3) are: „The journal Theoretical Computer Science will be published from now on as „The journal of the European Association for Theoretical Computer Science". Precisely this is what will be printed on the front cover of TCS below the title of TCS which remains Theoretical Computer Science". „As an effect of this agreement, and that was the main purpose in signing it, a personal subscription rate will be offered to EATCS members. ... Otherwise there will be no change in the editorial policy of TCS which remains an international journal open to authors from all over the world. It should be clear that no „preference" will be given to European authors, all papers being refereed according to the usual international rules. No change in the Editorial Board is planned for the time being. .... the link between TCS and the EATCS is now ensured by the fact that 9 members of the Editorial Board of TCS are council members of the EATCS. (In fact the Association and the Journal were created a few years ago by the same group of people). It seems reasonable that this proportion of half the members of the Editional Board of TCS being members of EATCS, agreed upon by the general assembly of EATCS, be respected in the future. And also that when the Editor in Chief resigns he be replaced by someone who is agreed upon by the EATCS general assembly." And M. Nivat continues:

„This means that the Editor in Chief of TCS takes the engagement (even if this is not written in the above mentioned agreement) to keep the EATCS informed to every change in the Editorial Board, or in the editorial policy, to send his resignation to the president of EATCS at the same time as to North Publishing Cy, to resign immediately if the general assembly of EATCS emits a note towards this effect. He also takes the engagement to give periodically informations about the status of TCS in this Bulletin, including titles of forthcoming papers, length of the backlog and whatever seems proper to be brought to the knowledge of EATCS members. This will start in the next issue of the Bulletin. Right now the Editor in Chief of TCS would be very happy to receive all comments, complaints suggestions ... of EATCS members as concerns TCS". Since that time M. Nivat, who is still Editor-in-Chief has reported regularly to council and general assembly and occasionally in the Bulletin - e.g. when the split into sections A (automata, algebra und algorithms) and B (logic, semantics and related topics) was decided upon

(Bulletin no. 45, p.2,3, October 1991); the contents lists of the current TCS issues are printed regularly in the Bulletin since 1983).

The Bulletin

The best at the end.

That EATCS members identify themselves with their association is mainly due to the ICALP series and to the Bulletin. The success of the Bulletin - i.e. that people really like to read it - has several reasons: the high level and broad scope of its scientific contributions, the many informations about conferences, activities in institutes and organisations, etc., and as a very important part the fotos and the DADARA cartoons. Therefore it is not simply the EATCS Bulletin but it is our Bulletin.

One of the supports the Bulletin had given to EATCS and which should not be undervalued are the logos. The first logo of AEIT / EATCS can be found in the first Bulletin, edited by M.Nivat and printed at IRIA (Institut de Recherche d'Informatique et d'Automatique); it vanished rather soon, maybe because EATCS developed into a purely English speaking society.

Already on the cover of Bulletin no. 3 the now famous EATCS emblem appeared. It had been designed by M. Hennemann, Karlsruhe. The Bulletin editor H. Maurer also asked for further ideas. And there was a further idea by G. Rozenberg; his symbol is used in combination with the list of council members since Bulletin no. 4.

According to an agreement with Springer-Verlag the EATCS symbol is printed on the spine of each ICALP proceedings since the 5th ICALP as distinguishing mark from the other LNCS volumes. Moreover in 1984 the „European map" button was created by M. Kudlek and awarded to persons who, according to strong rules, did major contributions to ICALP. In Bulletin no. 5 (p. 115) from June 1978 the „European map" logo of EAT CS developed its own life with the help of M. Jantzen, Hamburg.

Textual Contributions

The EATCS Bulletin really is a wide-spectrum journal where each theoretical informatician may find useful informations from almost each aspect of scientific activities; typically an issue contains over 20 (even up to 27) sections. Different from the periodicals of other organizations the Bulletin is purely science-oriented, for example, there are no business advertisements - neither payed nor indirect ones - there are no stories about persons, instead the concentration is on scientific work and scientific results.

This has been the scope of the Bulletin from its first issue on. No. 1, the Nivat-Bulletin, from December 1973, contains already in addition to what is now called EATCS matters, some reports on computer science departments and insti-

tutes (from 3 places in the Netherlands, from the university of Torino, and from the university Paris VI), a report on the second MFCS conference, and some conference announcements".

In the Ausiello-Bulletin (no. 2, December 1976) additionally the first two „Technical Contributions" can be found. This Bulletin is the first one in which all (exactly 100) EATCS members are listed; in Bulletin no. 9 (October 1979) again such a list was published, it took 30 pages and contained 526 names.

At the beginning of the regular publication of the Bulletin in October 1979 (no. 3) the Bulletin editor H. Maurer refined and augmented the section structure of the Bulletin - which had been introduced by G. Ausiello. Moreover he made an explicit call for contributions: „Contributions of all kind for the Bulletin are solicited now. Information on the following items would be particularly useful: short technical contributions and announcements of new results, titles of new research reports, thesis etc. from your institution, information on the structure, aims and on-going research of your institution, information on conferences, working groups etc. you are planning, any other new items of interest".

No. 4 is remarkable not only for the first fotos but also because of its 116 pages, containing 6 „Technical Contributions" and 8 „Reports on Conferences". From that time on the Maurer-Bulletins found a strong resonance, a growing number of people did not only read them but also sent contributions - the success of the Bulletin became obvious. In 1981 the era of the Maurer-Bulletins ended, and G. Rozenberg startet with Bulletin no. 15 (October 1981). In his next Bulletin (February 1982) he already introduced two new sections „Reports on computer science organizations" and „Book reviews", and had announced that from then on he wanted to have „Reports on Computer Science Departments and Institutes" on a regular basis. In June 1982 G. Rozenberg came up with „Problems and Solutions", which for quite a while was a very popular topic of the Bulletin series. The section „Abstracts of Ph.D. Theses" was started in no. 19 (February 1983), followed in no. 20 (June 1983) by „Contents of TCS". In this way, by continuously producing new topics, the Bulletin editor strongly motivated people to read and to contribute to the Bulletin; the sizes of the Bulletins became bigger and bigger.

Although G. Rozenberg has been taken up since 1983 additionally with the EATCS Monograph series together with W. Brauer and A. Salomaa - his creativity for the Bulletin continued: in no. 25 (February 1985) „Surveys and Tutorials" started, in no. 28 (February 1986) „EATCS Monographs"; since February 1987 exists the „Columns", since June 1988 the series „News from ...." starting with Australia and Japan, later in February 1991 Latin America followed and in October 1993 New Zealand. Furthermore should be mentioned the sections „Sister Organizations" (since October 1988) and „ESPRIT - Basic Research Actions" (since February 1989). This list of sections is by no means a complete one, and but as a continuation of what has happened up to now Bulletin readers can be sure to find

pretty soon again a new topic - but what will come next?

Picture Contributions

Each family starts sooner or later to collect fotos of its members, taken in particular on special events. Vice versa if fotos of a community are taken maybe at special events, this community feels like a family. That's what happened to the EATCS members.

In Bulletin no. 4 from January 1978 the first fotos were reproduced, they documented ICALP'77 in Turku and MFCS'77 in Tatranská Lomnica, Czechoslovakia. Already since June 1979 the EATCS Bulletin has its official picture editor: Peter van Emde Boas, Amsterdam, The Netherlands had agreed not only to take pictures, to urge other members to send pictures to him, but also to compile and print them at the Mathematical Centre in Amsterdam; it was a hard job for him but very enjoyable for the Bulletin readers. Since October 1989 M. Kudlek, Hamburg, Germany is picture editor. He is wellknown for his systematic fotodocumentation of the speakers of ICALPs and other conferences although only few of them can be reproduced in the Bulletin. Let us hope that he will, besides his many jobs in ICALP, continue to take care of the fotos.

The other pictures which catch the attention as soon as a new Bulletin has arrived are the DADARA cartoons. In Bulletin no. 27, October 1985 they appeared for the first time, signed with „DR"" which is still used in no. 28; from no. 29 one the excellent DADARA cartoons can be found in each Bulletin. Who is DADARA? It is not a secret: he is Daniel Rozenberg, the son of Grzegorz Rozenberg; meanwhile he is quite a famous artist with TV presentations and art exhibitions in different parts of Europe and in the US. EATCS can be very proud to have DADARA's contributions in its Bulletin.

Printing and Mailing

Bulletin readers might think that printing and mailing of the Bulletin is of minor interest to them since it deals only with organization and finances. But from the very beginning on this has been the major obstacle for the installation of the Bulletin. In a letter from January 17, 1975 M. Nivat wrote to EATCS council members with regard to Bulletin no. 1, which was to appear in December 1973: „The shipment of the first issue of the Bulletin to all the members who payed 100 belgian francs, which I asked my secretary in IRIA to make in October was impossible due to a major mail strike in France. This issue of the Bulletin is very obsolete". M. Nivat obviously meant October 1974; no. 1 presumely reached its readers only in 1975.

It was mainly because of lack of money that the series of Bulletins could not start till October 1977 when H. Maurer got „the support of the University of Karlsruhe and of the Institute für Angew. Inf. u. Form. Beschreibungsverfahren of

that University" (see cover of Bulletin no. 3). This subsidy lasted only till H. Maurer left Karlsruhe and moved to Graz in 1978. Bulletin no. 6 got „the support of the Technical University of Graz and of the Research Centre Graz". But because of high mailing costs from Austria to overseas members surface mail had to be used. D. Wood, McMaster University, Hamilton, Ontario, Canada offered his help, he even had the volumes of Bulletin no. 8 for the 45 North American members printed in Hamilton and distributed from there. Unfortunately this procedure could not be continued.

Bulletin no. 11 was the first issue for which more material had been sent to H. Maurer than he could include since for financial reasons the weight of a volume should not exceed 250g i.e. some 180 pages. But the problems became even greater; in no. 12 (October 1980) secretary Th. Ottmann reported that H. Maurer: „... explained that not the editing but the actual production has become a big problem for him. To print about 120.000 pages and to mail over 600 Bulletins three times a year consumes already more time of his staff than he can justify. Latest after one further year, another solution for producing the Bulletin must be found".

Therefore from no. 13 on printing and mailing of the Bulletin became an additional duty for the secretary. Consequently it was again the University of Karlsruhe which supported the Bulletin. In spite of this help by Th. Ottmann, H. Maurer was unable to continue the job as Bulletin editor and from October 1981 on the new Bulletin team consisted of: G. Rozenberg (editor), P. van Emde Boas (picture editor) and secretary Th. Ottmann (production and distribution). Some problems had been solved, i.e. overseas members had to pay an extra fee to get the Bulletin by airmail but in general the financial shortage persisted. From 1983 on there was a little hope, because in October 1983 (no. 21) treasurer J. Paredaens announced: „Until now the only source of funds has been the membership fees, which are used exclusively for the Bulletin. In near future some other resources, e.g. publications, will be available". This was mainly due to G. Rozenberg. The excellent Bulletin team Rozenberg, van Emde Boas, Ottmann edited and managed the Bulletin during 5 years.

At the council meeting at ICALP'86 in Rennes B. Monien from Paderborn, West Germany was approved as secretary and consequently from no. 31 (February 1987) on the Bulletin was produced in and mailed from Paderborn. B. Monien and his staff members, in particular Walter Unger at the University of Paderborn were more and more confronted with the growth of the Bulletins, the volumes became fatter and fatter and heavier: from 298 pages (no. 31) to 323 pages (no. 34), 423 pages (no. 38) to 536 pages (no. 40); since then the number of pages varied between about 400 and 565 (no. 50), i.e. between 550g and 770g. In 1989 the team changed, M. Kudlek, Hamburg, Germany succeeded P. van Emde Boas.

At the council meeting at ICALP'95 in Szeged secretary B. Monien resigned

and B. Rovan, from Bratislava, Slovakia became his follower. The team: Rozenberg, van Emde Boas / M. Kudlek, Monien had published 27 issues of the Bulletin. During nine years the university of Paderborn, B. Monien, W. Unger and several other people in Paderborn had strongly supported the production and distribution of the Bulletin. Bulletin no. 58 was the first produced in Bratislava; there not only the printing but also the mailing costs are fortunately a little less than in Germany. The actual Bulletin team: Rozenberg, Kudlek, Rovan will hopefully act at least as long as the previous did for the benefit of the EATCS Bulletin.

Using the Bulletin

It is worthwhile to contribute to the Bulletin - at each ICALP the contributers to the last three issues are awarded with a little present by G. Rozenberg; this dates back to ICALP'84 in Antwerp. It is worthwhile to bring your own June issue of the Bulletin to the General Assembly at ICALP to get a nice prize from G. Rozenberg. It is worthwhile to read old Bulletins again! Looking on the Bulletin fotos it is a real surprise to realize how much or how less some persons have changed during the last 20 years.

There are a few Bulletins which contain texts which do not exactly meet the scope of the Bulletin but instead give a more private flavour and therefore should not be forgotten: one is the Bulletin no. 15 where on pages 7 and 8 a letter from J. W. Thatcher, IBM Research Center, Yorktown Heights is printed who after having met Ivan M. Havel in Prague in 1981 gives a very touching report on the situation of Havel and his family as consequence of their connection with the Charter 77 movement. And also in no. 15, pp. 8-21 A. Salomaa explained in a purely scientific manner „What computer scientists should know about Sauna" a reprint can be found in no. 35, pp. 15-26. Bulletin no. 20 is remarkable for a contribution in French „Vive ICALP!" by M. Nivat - it has been the last non-English text. The translation into English which is given together with Nivats text caused some problems; the translator team consisted of a native Dutch, Polish and American. Finally the no. 50 has to be mentioned as the thickest Bulletin up to now (565 pages), containing a history of the EATCS Bulletin series demonstrated by clippings by Lila Kari and Arto Salomaa.

Surely each Bulletin reader will find her or his favoured volumes but whatever volume is chosen, very likely it will be due to Grzegorz Rozenberg. You read right now the 48th Rozenberg-Bulletin. He very much improved and determined the character of the Bulletin, moreover he is the heart of the Bulletin series - let us hope that his heart will continue to beat for the EATCS Bulletin.

For the Future

To complete our personal view on the history of EATCS, an appreciation of ICALP, the best known activity of EATCS, should be indispensable. However, next year

EATCS will celebrate its 25th ICALP in Aalborg, Denmark and this merits an extra and a special report which will be given by the ICALP expert, M. Kudlek; he is the only person in the world who had been to all ICALPs, therefore who else could do this job better than he!

For those who think that one year is much too long to wait for the ICALP history, please look into Bulletin no. 52, pp. 32-134 (February 1994) where detailed informations and statistics have already been given by M. Kudlek.

EATCS has reached its goal to be a leading society for theoretical computer science. During the past 25 years EATCS was a valuable, comfortable and well-esteemed scientific home for theoreticians. Let us hope that also in the future scientists shall be willing to do - on a honorary base - hard and continuous work for EATCS. EATCS - that is its members and their activities. Congratulations to EATCS and all the best for the coming 25 years.

Ute Brauer
Wilfried Brauer
Technische Universität München

Know the Person behind the Papers

Today: Christos Papadimitriou

We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Christos: Yannakakis and me is unusual. We went to the same high school in Athens, studied in the same department in Polytechneio, and we both got our PhDs at Princeton (I was 2 years ahead in all this). He does not always admit it, but I recruited him to TCS during his first PhD year, when he was enamored to Communication Theory. When I succeeded, I remember thinking "I have now done enough for TCS". Our CVs intersect only by some small fraction, but this fraction contains some interesting work. The picture is from a time in the 1980s when we had begun to look similar, and people at conferences had started to get confused. Then we diverged a bit. Joining him at Columbia five years ago (our office doors are 3 meters apart) was like a dream come true. When we work together, our bandwidth – the depth of our shared intellectual and life experience – is beyond anything that I have known.
The second picture is recent, by my good friend Muli Safra, who is the most talented photographer that I know.

Can you please tell us something about you that probably most of the readers of your papers don't know?

Christos: My student years in Greece were miserable. The country was under a US-supported fascist dictatorship which lasted roughly throughout my student years, and Polytechneio, my university, was an authoritarian place in a way that was not altogether unrelated. I hated my studies but I was doing okay. Students were trying to resist, but we were getting nowhere, except to jail and abuse. When I graduated I got a job as an engineer, but had to join the army first. My year in khaki was by far the worst of my life. I saw evil in the eye, every day, and I was powerless. My health suffered badly. I decided I had to leave Greece, and graduate school was my only way out. Princeton was one of two schools that admitted me, and they informed me that I was to join their Computer Science group – a category I was completely unaware of. They classified me this way probably because the only phrase in my statement of purpose that they could understand was "IBM 1460". I ran to Princeton in 1973, two weeks after my discharge from the army. I was not serious about my studies, and I was opening my eyes to New York and the new opportunities around. Then I looked at my first homeworks – in Logic and Automata Theory – and I was hooked. Here is an intellectual endeavor that existed only in my dreams and prayers, and now I can explore it, and on top of it all I seem to be good at it! For a few weeks I was in heaven. Then in November, the students at my former school, the Polytechneio, started a rebellion that was crushed by tanks. People I knew were killed. I was a deserter – in fact, one who had fled behind enemy lines.
Such are the contradictions that defined me. In 1980 I left my job at MIT to teach at the same Polytechneio. For a few years I became a European TCS researcher, and I was on the board of EATCS – I even organized the 1985 ICALP at Nafplio. Those who were there have not forgotten: Don Knuth gave a talk on "Theory and Practice" in the ancient theater of Epidavros, and was introduced in fiery left-wing prose by famed actor Melina Merkouri, then minister of culture. Four decades after the fact, confessions come easy: I had written that speech.

Is there a paper which influenced you particularly, and which you recommend other community members to read?

Christos: Alan Turing's two most famous papers – or is this too much of a cliché? I occasionally teach a course on "Classics in CS" where I cover – well, the classics. Besides Turing we go over Gödel, von Neumann's interim report, Shannon, Danzig and Edmonds, Cook-Levin-Karp, Codd, Dijkstra, etc. Even Euler and his bridges. It is good to remind yourself once in a while that there are times when a researcher – in most cases in their 20s – realizes that there is something terribly wrong with the science around them, and takes time to fix it.
But Alan Turing is special to me. A tribute titled "Alan and I" was published in CACM on Turing's centennial in 2012. It narrates how, all of a sudden, in 1997 I started to write fiction, inspired by Turing. Before that I had written nothing

besides math, and yet after this experience I realized that I will never stop writing stories. My first novel "Turing (a novel about computation)," an oblique homage to Alan Turing, led me to a collaboration that ended up with the graphic novel "Logicomix," which became a kind of best seller. Then I wrote "Independence," a story about modern Greece (including the Polytechneio insurrection) narrated by a failed mathematician and backgammon champ named Christos P.

Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

Christos: "A biologically plausible parser" (2022) describes a parser of English that is implemented exclusively by neurons and synapses. It is based on intricate TCS (work with Santosh Vempala), which however is only in the references here. It is the latest step in my quest to understand how the brain works, something I have been thinking about for many years now. Develop a formal, computational understanding of the brain at a level of detail bridging the two extremes: individual neurons and synapses, and cognitive phenomena at the other end. Nothing I have done compares in difficulty or excitement.

What do you do when you get stuck with a research problem? How do you deal with failures?

Christos: Ours is a good life that runs in a very peculiar pattern: months of frustration, then a moment of elation, and then you must write a technical paper. Repeat. There is a second kind of joy and longing: when you have identified a problem worthy of a research life, but feel completely powerless in front of it. And yet another moment of importance: when you turn your back to the problem, but promise yourself to return in good time, and take a fresh look.

Is there a nice anecdote from your career you like to share with our readers?

Christos: There is a picture on the web of me playing piano with Don Knuth, clad in regalia. In 2003 the two of us were about to receive honorary doctorates from the University of Macedonia in Thessaloniki, and the music department asked virtuoso Don to play something at the ceremony. To my horror, he responded: "I will play only if Christos agrees to play à quatre mains with me." I did play rock-and-roll keyboards as a teenager but by then I was very rusty. I spent months practicing the pieces with my daughter, and when the day came I was ready. We practiced with Don in the morning of the ceremony, and he could cover all my inadequacies. That afternoon, after we sat in front of the piano with our heavy academic gowns and started Debussy's "En evoquer Pan," I burst out laughing, and Don noticed the problem at about the same time: with our long-sleeved gowns we could never cross hands as required by the piece! There were two hilarious bars of musical chaos before we ended with gusto.

A couple of years later my musical career took a different turn: I played with a rock band called "The Positive Eigenvalues." Mike Jordan was my drummer, and David Culler my guitarist – not many keyboard players can say this. I also wrote the original songs of the band, and some of them were about TCS – this blogpost contains a couple of examples http://blog.geomblog.org/2013/10/focs-reception-sing-along.html
By the way, you may have noticed a pattern in my life's account: All my closest friends are computer scientists. It's not that I don't have a life, it's that computer scientists are all so amazing!

Do you have any advice for young researchers? In what should they invest time, what should they avoid?

Christos: If you do not feel an irresistible attraction to your research, you are in the wrong place. Don't work on a problem just because somebody else could not solve it. No math you learn is useless. Even the hardest problems are likely to crumble eventually, make sure to be there when they do. Create your own problems as often as you can; your advisor is there to advise you, not to dictate problems to you. Read Manuel Blum's advice to young researchers.

What are the most important features you look for when searching for graduate students?

Christos: What are you looking for in your new-born child? A huge moment, a life-long relationship is starting, the future is present, words fail you.

Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

Christos: Modern TCS started around 1970 with the realization – through the papers of Cook and Strassen, to pick only two – that elegant mathematical work can inform computation in a direct and consequential way. In its first quarter century it bloomed: it articulated one of the greatest questions in all of science ("can exhaustive search always be avoided?"), and provided mathematical help for the solution of a few epic engineering problems: compilers, databases, security, chips, networks, more. Then the Internet came, applied CS fields developed home-grown theories and theoreticians, and TCS changed course. It started on its new dual mission: develop even more sophisticated math for the big problems of algorithms and complexity, and use the profound insights into computation we have absorbed over the decades to illuminate other sciences and make progress on their important problems. This is where we are now. With luck, there will be even more exciting progress on both of these research modes and fronts. And the interplay between the two will grow as well.

How was your research affected by the pandemic? How do you think it will affect us as a community?

Christos: Research happens in the world, and the world changed these two years. We may be wiser, more humble, well versed in zoom, more in touch with the planet's unity. And yet we are also more unequal: if you were rich or poor, powerful or powerless, you are probably more so now. I fear that wealth inequality is a timer of planetary disaster that may be even more ominous than climate.

Know the Person behind the Papers

Today: Anne Driemel

Bio: Anne Driemel received her doctorate in 2013 from Utrecht University, after studies of computer science at the Free University of Berlin and the University of Pennsylvania. Following stations at the TU Dortmund and the TU Eindhoven, she moved to the University of Bonn in 2018, where she is now a professor at the Institute of Computer Science and a member of the Hausdorff Center for Mathematics. Since 2022 she is an elected member of the Computational Geometry Steering Committee (term 2022-2026).

We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Anne: This photo was taken in our home in Eindhoven. My husband is also a computer scientist and when we first moved in together, we wanted to have a proper chalk board in the home office. It turned out later that we did not use it much for work, but our children liked to draw on it. The second picture shows what is going on behind the scenes.

Can you please tell us something about you that probably most of the readers of your papers don't know?

Anne: Before I discovered theoretical computer science as a subject, I often wanted to become a writer. However, there was no clear career path. Besides, I didn't know what to write about. I thought I first had to get some life experience before I could write a book. I guess you could say that I am an author now, although it is not quite how I envisioned it.

Is there a paper which influenced you particularly, and which you recommend other community members to read?

Anne: I was intrigued by "The nature of statistical learning theory" by Vladimir Vapnik, where he explains central ideas and the historical context of the VC-theory, which is important for PAC-learning. More generally, I recommend reading older concept papers in artificial intelligence. I think those are highly relevant to the field of computer science as a whole today, because many of these ideas have shaped the field from the beginning.

Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

Anne: This would be my paper with Peyman Afshani on the complexity of range searching among curves. The story behind it is that Peyman was looking for a range searching problem that is strictly harder than simplex range searching. I was interested in the problem of range searching under the Fréchet distance and whether multi-level partition trees could be used for that. Peyman had developed a technique for proving lower bounds that he wanted to apply. There is also a deeper question involved, namely whether the additional logarithmic factors, that

are caused by extra levels in a multi-level partition tree, are somehow artifacts of the construction, or if we can find a range searching application that observes these extra factors in a lower bound. It turns out that the Fréchet distance has this property.

When (or where) is your most productive working time (or place)?

Anne: This used to be in the evenings. Now that I have a family, the productive times come and go and I don't see a clear pattern. I am probably most productive when I can forget everything around me. This has less to do with a certain time or place, but with the circumstances that allow me to let go of things (e.g. reliable good quality child care).

What do you do when you get stuck with a research problem? How do you deal with failures?

Anne: I learned that it is good to cycle through a small stack of problems. Whenever you get stuck on one problem, you move to the next in the stack. Ideally, by the time you reach the problem again, where you got stuck, you have cleared your mind of the stuckness. Also, it really helps to collaborate with others on research problems, not only to share expertises, but simply because talking about the problem can help tremendously to sharpen the focus and to distill the questions that are important.

Is there a nice anecdote from your career you like to share with our readers?

Anne: I can't think of anything interesting right now.

Do you have any advice for young researchers? In what should they invest time, what should they avoid?

Anne: Find a topic area or application that you care about and try to formulate a new concept or fundamental problem within that area that has not been studied before. Avoid chasing after low-hanging fruits in competition with others. Collaborate, share your problems with others.

What are the most important features you look for when searching for graduate students?

Anne: Ideally, I would like to know if the candidate knows how to write and in particular if they are able to develop their own ideas and thoughts in writing. Another question is how well the candidate is motivated for an academic career. Both things are often difficult to assess, but sometimes the application letter and the interview give away enough hints.

Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

Anne: Climate change.

What kind of opportunities should EATCS offer to researchers, and especially to young researchers?

Anne: I think we could do more to support young researchers with young children. Often, these different challenges come together at a time when an academic career is the most volatile, namely during the postdoc phase. Travel support for conference trips with young children would be great, e.g., to support travel costs of a care-taker person that travels with you.

What can be the role of EATCS in solving the challenges of our society?

Anne: I think we should not underestimate the role of theoretical computer science in society. Sometimes, good theoretical work needs some time to flourish and establish itself, but eventually it may have an impact in unforeseen ways.

Please complete the following sentences:

- *My favorite movie is the Three Colours Trilogy by Krzysztof Kieślowski.*
- *Being a researcher is being free to decide which great problems to spend your time and energy on.*
- *My first research discovery is the notion of c-packedness that characterizes a realistic class of curves for which the Fréchet distance can be approximated in near-linear time (together with Sariel Har-Peled and Carola Wenk).*
- *Theoretical computer science in 100 years from now will be based on the same mathematical foundations.*
- *EATCS in 50 years from now will look very different (but I don't know how).*
- *Being respected is key to being a happy academic.*

KNOW THE PERSON BEHIND THE PAPERS

Today: Paul Spirakis

Bio: *Paul Spirakis was the director of the Greek Computer Technology Institute (CTI) from 1996 to 2016. He is now in the leadership team of the Leverhulme Research Center for new materials design of the University of Liverpool , a faculty member in U. Liverpool and also in U. Patras. He graduated from the National Technical University of Athens and earned his Ph.D. from Harvard University.Among other, Spirakis is known for his contributions in probabilistic techniques in algorithms , foundations of distributed computing and algorithmic game theory. He co-established the present form of the Computer Technology Institute and Press "Diophantus" in Greece. He played an important role in the establishment of several Conferences such as the European Symposium on Algorithms , the Conference on Distributed Computing (DISC) and the Conference on the Internet and Network Economics (WINE). He was the President of EATCS from 2016 to 2020 and he is a Member of Academia Europaea and also the Editor-in-Chief of the TCS-A journal. He served in several research related EU bodies and chaired the ERC Informatics Panel.*

We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Paul:

I share two photos. The first is in Boston on 1980 with my advisor John Reif. My wife Asimina is at the right corner (not so visible).The second is me and my youngest student (my grandson Marios) in Liverpool on 2019.This photo shows that I can teach theory to very young students, even in the street, buying food.

Can you please tell us something about you that probably most of the readers of your papers don't know?

Paul: When I started my undergraduate studies in 1973 in Greece (in Electrical Engineering) I had never used a computer. In 1975 I started to interact with a mainframe by punching and submitting cards to a person who was responsible for compiling and delivering the output to us in printed form. The first computer language I used was FORTRAN. In 1977 I started my undergraduate Diploma Thesis

on the subject "Information processing in the nervous system of man". I learned then about neurons and I wanted to do graduate work in biomedical engineering. At the same time I started learning about Markov Processes and I proposed a Markov Chain for the information processing in the cerebellum ! As an Electrical Engineer I was facsinated from the use of stochastic processes.

Is there a paper which influenced you particularly, and which you recommend other community members to read?

Paul: A year after starting my graduate work at Harvard I read the paper of Dana Angluin and Leslie Valiant on Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings (appeared in 1977 , I read it in 1979). This was the first time I saw the Chernoff bounds. I used them later in my Ph.D. Thesis. At that time almost nobody was using tail bounds on sums of random variables.

Very young I read the book of Karl Marx "Das Kapital". I suggest it to everybody independently of political beliefs.

Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

Paul: Read the paper "An Optimization Approach for Approximate Nash Equilibria" (Internet Mathematics 2008) by Haralampos Tsaknakis and myself. Haralampos was a mathematical genius. He is no longer in life and I want to point out this paper so that we all remember him. At that time Haralampos was working in a managerial post in some firm in Greece. I met him first when we were both high-school students in Thessaloniki in 1972. We met again around 2005 and he

was working on some security problems in his spare time. Thn I told him "do you want to work on a nice problem in Game Theory ?" . And this paper (whose achieved approximation ratio is not beaten till now) came out afterwards.

When (or where) is your most productive working time (or place)?

Paul: I am an evening person. I start working in the morning but not very

early. During the day I interact with students , go to meetings, teach and administrate , write some reports etc. My evenings are usually devoted to research , some time until late at night.

The schedule is different when I travel but nowadays this is not a problem.

My favorite work place is my office (and my office at home) but I can think and work almost everywhere (even in train or plane)

What do you do when you get stuck with a research problem? How do you deal with failures?

Paul: Failures are much more frequent than successes. When I am stuck I keep trying and I try to understand the problem better. Also I discuss the problem with colleagues and students. Hard problems need lots of trying. Sometimes I become disappointed and wnat to forget the problem but it comes again in my mind. It helps to work on more than one problem at the same time. When you are tired of one you can switch to the other. My experience is that when I insist on a problem , something comes out eventually , maybe not on the original problem but on a related one.

Is there a nice anecdote from your career you like to share with our readers?

Paul: When I was applying for graduate studies I was mostly applying to the U.S. (Greece had not graduate studies at that time in 1978. And I was mostly applying for biomedical engineering or cybernetics. Then one day I got a phone call from Christos Papadimitriou. He was visiting Harvard and my application to Harvard somehow arrived to him. He told me if I knew a lot about the subjects I was looking for. I honestly told him that I had no idea but I was fascinated by subject names. He told me "here we have some different topics , such as algorithms , automata etc.". I took the risk to accept the offer (was coming with a fellowship and my parents could not pay for graduate studies). I was lucky in my decision then.The funny thing is that in my first year at Harvard I worked on performance evaluation of computer systems (not on algorithms) Was natural , it involved some queuing theory and I knew a bit of that from my undergraduate studies. After a year I switched subject (and advisor !) and I started working on algorithms with John Reif. John influenced me a lot and inspired me to work on those , new for me then , topics. I owe a lot to John for his direction and for helping me to start a career.

Do you have any advice for young researchers? In what should they invest time, what should they avoid?

Paul: They should work on subject that psyche them ! They should avoid current fashion (unless they really want to work on topics related to current fashion). Think deeply , make research your way of life.

What are the most important features you look for when searching for graduate students?

Paul: Mathematical talent and desire to work even on subjects they are not very familiar with.

Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

Paul: Theoretical computer science has a great future. New topics and questions arise all the time : population protocols in distributed computing , economics and algorithms , new models of dynamic and random graphs , theory of learning and big data , new complexity classes , are some examples.

In my career I switched topics several times starting to work on topics that were not so known (or even did not exist) before.

What kind of opportunities should EATCS offer to researchers, and especially to young researchers?

Paul: EATCS is a scientific association. It offers already many opportunities , ranging from prizes and awards to schools and to job openings announcements. Most importantly it supports several theory conferences (including its flagship conference ICALP) and thus offers to researchers (especially to young ones) the opportunity to meet peers and older scientists and gain in wisdom and cooperation. EATCS is a big family of european theorists. The bulletin itself highlights important topics and discoveries. Hard to find out if EATCS misses to offer some opportunities to researchers. If there is any I am sure that the EATCS bodies will spot it and act on it also via continuous interaction with european researchers.

What can be the role of EATCS in solving the challenges of our society?

Paul: Well , it could motivate the relation of such challenges to theory ! For example , I think we need a theory of humane algorithms. When theorists (especially young ones) become aware that their skills and scientific goals can also help society then an additional quite strong team of thinkers will be added to the people that try to solve the challenges of our society.

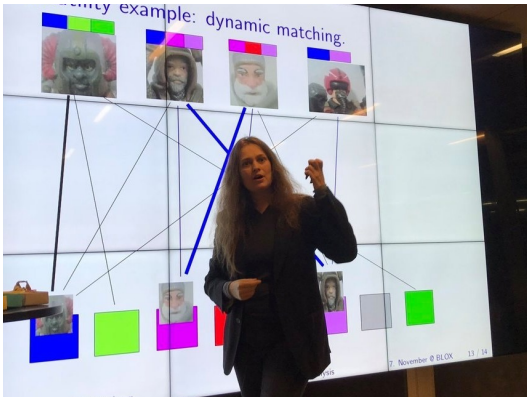
Please complete the following sentences?

- *My favorite movie is...Matrix (the first one).*
- *Being a researcher...is big pleasure and a way of life.*
- *My first research discovery...was not on theory but on performance analysis of computer systems.*
- *Theoretical computer science in 100 years from now...will be even stronger but perhaps very different from now.*
- *EATCS in 50 years from now...will have many more members and will be be key in relating theory to societal challenges.*
- *Love for research and for teaching ... is key to being a happy academic.*

KNOW THE PERSON BEHIND THE PAPERS

Today: Eva Rotenberg

Bio: *Motivated by curiosity and the search for mathematical elegance, Eva Rotenberg's research focuses on graph algorithms, particularly dynamic graphs, and touches upon other areas within algorithms. Eva Rotenberg is an associate professor at the Technical University of Denmark, and holds a PhD from the University of Copenhagen from 2017. Eva Rotenberg currently holds a grant from Independent Research Fund Denmark, and starting grants from the Villum Foundation and the Carlsberg Foundation.*



We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Eva: In this picture, I am trying to illustrate the problem of *online matching with recourse* to a small general public audience, using toy figurines and origami boxes of different colours. Everyone could sympathise that having to move figurines between boxes all the time was really cumbersome, and were relieved to hear that a very simple algorithm was mathematically guaranteed to make very few moves in total. (See *Online Bipartite Matching with Amortized* $O(log^2 n)$ *Replacements* [**?**].)

Is there a paper which influenced you particularly, and which you recommend other community members to read?

Eva: Let me recommend: *"Dynamic Representations of Sparse Graphs"* by Gerth Stølting Brodal and Rolf Fagerberg [**?**].

When Jacob Holm and I were working on an algorithm for fully-dynamic planarity testing [**?**], we suddenly realised that an analysis similar to that in this paper would help us take a big step towards solving our problem.

Today, I use this paper in our advanced algorithms class when introducing the students to the topic of dynamic graph algorithms.

Is there a paper of your own you like to recommend the readers to read?

Eva: One suggestion could be: *"A Hamiltonian Cycle in the Square of a 2-connected Graph in Linear Time"*, joint work with Stephen Alstrup, Agelos Georgakopoulos, and Carsten Thomassen [**?**]. Working with Carsten Thomassen has taught me a lot about simplicity and elegance in proofs and algorithms.

When (or where) is your most productive working time (or place)?

Eva: The question is hard to answer because there are so many different ways of being productive, that are all necessary in order to solve problems. Different kinds of productivity flourish under different conditions. Sometimes it is even possible to just wake up with a new, vague idea, that gives a new perspective on some problem. One of my favourite ways to work is to stand by a board and discuss ideas with other people.

What do you do when you get stuck with a research problem?

Eva: Regardless of whether a specific idea works and solves the problem, it is interesting to understand the power of that idea: does it work under other assumptions or restrictions? Is there a version of the problem that it solves?

Often when we are stuck, it is because there is this one annoying case, that stands in the way of our original algorithmic idea working. Then, it can be helpful to draw and describe this particular case – I prefer first to draw it in a a very imprecise high-level way. Then starts the work of really, intensely, understanding the nature of this annoying case and why it annoys us. Hopefully, in this process, one gains insights that inspire new algorithmic ideas.

Do you have any advice for young researchers?

Eva: Sometimes, PhD students worry that they are educating themselves too broadly, working on seemingly unrelated problems within theoretical computer science, and publishing papers on very diverse topics. The way I see it, there is nothing wrong with having a broad set of skills and a broad range of interests.

What kind of opportunities should EATCS offer to young researchers?

Eva: We want to have a safe and welcoming environment for students and newcommers. It would be great to offer summer schools and workshops, e.g. in connection with 'our' conferences. We could also consider supplementing our best student paper awards with a best student presentation awards, inspired by the computational geometry community.

Please complete the following sentences?

- *My favorite movies are...* old. Yet, I prefer not to see the same one twice.
- *I like theoretical computer science because...* it is difficult, fun, interesting and aestestically pleasing.
- For me, collaboration ... *is key to being a happy academic.*

KNOW THE PERSON BEHIND THE PAPERS

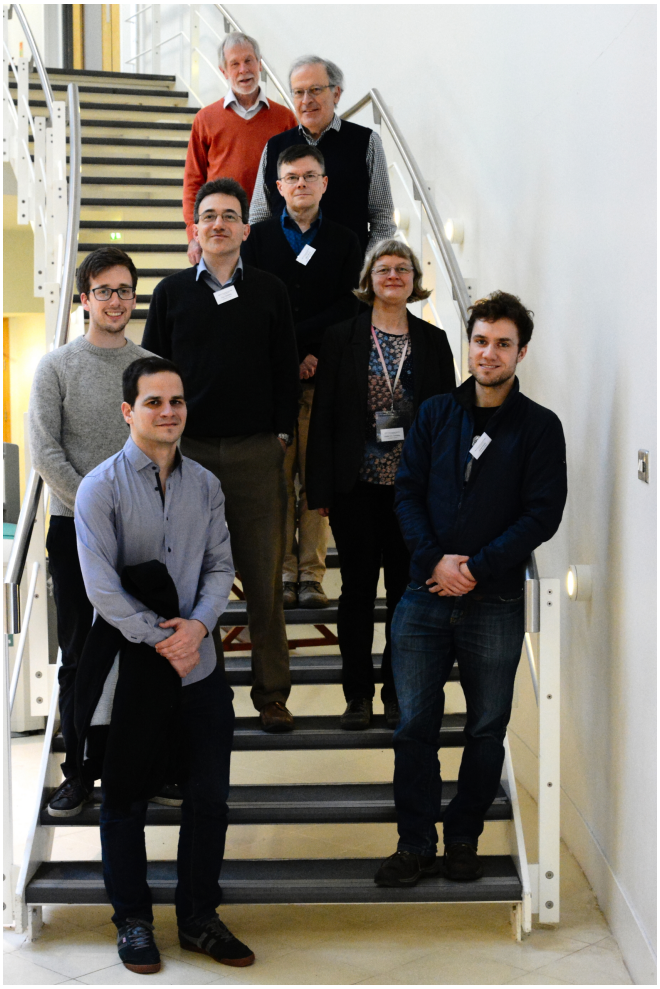
Today: Leslie Ann Goldberg

Bio: *Leslie Ann Goldberg is the Head of the Department of Computer Science at the University of Oxford. Her work is in the area of Randomised Algorithms and in the related area of Approximate Counting. Leslie received her BA in 1987 from Rice University and her PhD in 1992 from the University of Edinburgh. She was awarded an ERC Advanced Grant in 2014, and was elected to Academia Europaea in the same year. In 2016 she received a Suffrage Science Award. She and her co-authors have won four best-paper prizes at ICALP.*

We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Leslie: Like some others, I've included two photos here. The first one was taken in 2016 in the "green room" at the Hay Festival of Literature and Arts. I was there because I was giving a public lecture entitled "Algorithms and their Limitations" about P vs NP. I really like doing this kind of outreach, and I chose

this photo partly for that reason, and partly because my main non-academic hobby is reading fiction. The annual festival has grown from its original literary purpose to now include science, politics, and music. I'm pretty sure that I look both nervous and excited in that photo. Partly nervous about my talk, and partly about being in the presence of literary giants. Salman Rushdie had just walked by!



The second photo is a "context" photo. This is a tiny piece of Mike Paterson's computer science "family tree". Standing below Mike is his once-student Les Valiant, and below Les is his once-student, Mark Jerrum. Below Mark are my husband, Paul Goldberg, and myself, both once-students of Mark. On the next

row are Paul's then-student Edwin Lock and my then-student Jacob Focke. On the bottom row is Andreas Galanis, a really important collaborator for me. Andreas fits into the picture because he is a former student of Eric Vigoda, who is a former student of Alistair Sinclair, who is also a former student of Mark. I chose the photo because of the great TCS context — so many great colleagues to whom I owe so much!

Can you please tell us something about you that probably most of the readers of your papers don't know?

Leslie: I was a latecomer to Maths and Computer Science. My dream, as a young person, was to be a civil rights lawyer. At Rice University, I took a double major in Political Science and Computer Science. Political Science, to prepare for postgraduate study in law, and Computer Science because I mistook it for vocational training which would give me a way of paying for my study in law! Theoretical Computer Science was my first introduction to open problems in mathematics. I was overwhelmed by how fascinating it was.

Is there a paper which influenced you particularly, and which you recommend other community members to read?

Leslie: I think that papers in Theoretical Computer Science do age a bit with time, given the speed of advances in the field. Two papers that were hugely influential on me when I was a PhD student were Valiant's "The Complexity of Computing the Permanent" and "The Complexity of Enumeration and Reliability Problems", which essentially introduced the field of Computational Counting (one of the many fields that Les has initiated!). I was definitely also influenced by Jerrum and Sinclair's "Approximating the Permanent" which introduced some great techniques and ideas. They also write very well.

Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

Leslie: I can't really imagine recommending that anybody should "study" one of my own papers! That feels very arrogant and strange. Let me say instead that I am usually most enthusiastic about some of my recent work. At the moment I am very excited about contention resolution, which is something that I really liked working on a long time ago with Mike Paterson and others, and which I've very recently come back to with John Lapinskas. I'm very excited about our new paper "Instability of backoff protocols with arbitrary arrival rates".

When (or where) is your most productive working time (or place)?

Leslie: I'd love to be one of those people who can work perfectly well in noisy places in odd snatches of 10 minutes, but in fact I work best when I have

long quiet periods alone, especially in the morning. When I was a PhD student I once solved a problem that I'd been stuck on for many months during a walk. I was with others who had much better gear for Scottish peaks (something I fixed later!) and I was too cold to continue to the top with them, so I spent the day with the sheep much lower down. It was a blow to the pride to drop out of the excursion to the top, but it was good compensaton to be rewarded with an idea for my problem!

What do you do when you get stuck with a research problem? How do you deal with failures?

Leslie: I think the right approach to getting stuck is to divide time between (a) persisting and (b) working on something else. You want to do the first because you can't solve a problem if you don't even try. You want to do the second because the original problem might not be solvable! Failures are actually kind of nice because they give you problems that you can "carry around" to come back to later in life. The main open problem from my PhD (the complexity of approximating the cycle index polynomial) is still open. The main problem that I was working on with Mike Paterson at Warwick around 20 years ago is the source for what John Lapinskas and I have picked up recently.

Is there a nice anecdote from your career you like to share with our readers?

Leslie: When I was a young researcher Martin Dyer asked me whether I'd been invited to a certain Oberwolfach meeting and I admitted that I hadn't been invited. He rushed to explain "Oh, don't worry. It isn't what you know. It is *who* you know." I hadn't actually been very upset about not being invited — after all, only a small number of people can attend — but I was very touched by Martin's kind reassurance. He was a good mentor, to me and to others.

Do you have any advice for young researchers? In what should they invest time, what should they avoid?

Leslie: The following advice is something that I learned from Robin Milner. At the time I was struggling to find a PhD topic and my supervisor, Mark Jerrum, was also pretty young, so we went to Robin for advice. He advised me to stop thinking so much about the "big picture" and instead to focus on a small problem that I enjoyed working on. His comment was that the small problem would always lead to something else, so there was no need to do so much planning. I think it is good advice.

What are the most important features you look for when searching for graduate students?

Leslie: Enthusiasm for the topic. Problem-solving ability. Being curious about problems. Being hard-working.

Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

Leslie: I see many opportunities for theoretical computer science! Computing is becoming more and more important and I don't expect that to change. But as computing becomes more and more ubiquitous, foundational questions like "how long does it take" and "what is actually possible" become increasingly important. Right now "deep learning" is proving to be useful for many practical problems. But it will have limits and I expect TCS to be at the forefront of figuring out what those limits are, and what can be done instead.

What kind of opportunities should EATCS offer to researchers, and especially to young researchers?

Leslie: In addition to offering outstanding conferences, EATCS offers a sense of community. I think this is particularly important for young researchers.

What can be the role of EATCS in solving the challenges of our society?

Leslie: I'm strongly of the belief that the best way to obtain research that has a societal impact is to support "blue-skies" curiosity-driven research. There are lots of examples of curiosity-driven research that later turned out to have a big "impact" (consider, for example, radar, x-rays, or even calculus). This is true in science generally, and it is also true in the Theory of Computing. Oded Goldreich and Avi Wigderson have written a nice essay about this entitled "The Theory of Computing: A scientific Perspective". My main point is that the EATCS does and should stay focussed on "discovery of truth" and "foundational understanding". This is the best way to contribute to societal challenges.

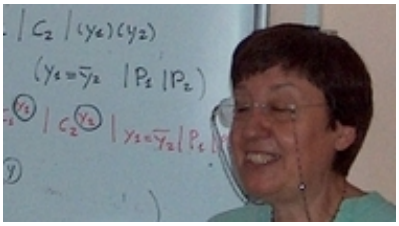
Please complete the following sentences?

- *My favorite movie is...* perhaps "Before Sunrise" — I'm not really the sort of person that has a "favourite movie". I'm not even sure whether there are any movies that I've seen more than twice, apart from maybe "Casablanca".
- *Being a researcher...* is a very rewarding career!
- *My first research discovery...* was probably written on a piece of paper that I don't have anymore!
- *Theoretical computer science in 100 years from now...* will be making lots of fascinating discoveries that we can't predict now.
- *EATCS in 50 years from now...* will hopefully continue to be at the centre of this fascinating area.
- intellectual curiosity ... *is key to being a happy academic.*

KNOW THE PERSON BEHIND THE PAPERS

Today: Mariangiola Dezani-Ciancaglini

Bio: *Mariangiola graduated in Physics the same year when Corrado Böhm joined the University of Torino as the responsible of the new degree in Computer Science. Mariangiola, having an interest both in research and in teaching, was planning to pursue an academic career in Theoretical Physics, but when she meet Corrado by chance she was totally fascinated by his imaginative way of investigating problems. Therefore, Mariangiola started to do research in Computer Science under the superb guidance of Corrado. She was always employed at the University of Torino, but she loved travelling and working with different people, hence she visited many universities and research centres during her career. As a self-present for her 50th birthday, Mariangiola got a PhD at the University of Nijmegen with her friend and co-author Henk Barendregt being her supervisor. At the core of the research activity of Mariangiola there are types, starting with intersection types for building models of the λ-calculus. Later, she devised types for object calculi, biological systems and concurrent processes, in particular session types for web services. Mariangiola became EATCS Fellow in 2015.*



We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Mariangiola: I share two photos which represent my main research interests. The first photo was taken in 1978 at the "Spring School on λ-calculus" and it appears in the book "The Lambda Calculus" by Henk Barendregt. In the second photo I am giving a talk in Novi Sad about session types in 2008.

Can you please tell us something about you that probably most of the readers of your papers don't know?

Mariangiola: I always loved to play with children and this activity frequently had a good influence on my scientific discoveries.

Is there a paper which influenced you particularly, and which you recommend other community members to read?

Mariangiola: One of the first papers Corrado suggested me to read was "Toward a mathematical semantics for computer languages" by Dana Scott and Christopher Strachey. This paper has influenced my love for semantics.

Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

Mariangiola: Somebody interested to intersection types could look at "A tale of intersection types". I wrote this paper with my friend and ex-student Viviana Bono during the confinement for the Covid pandemic. My isolation was softened by this work.

When (or where) is your most productive working time (or place)?

Mariangiola: I can work everywhere, also surrounded by noise, but I like to sleep in the morning, while I can be productive late in the evening.

What do you do when you get stuck with a research problem? How do you deal with failures?

Mariangiola: I always try to work in parallel to more than one paper and with different people. In this way, when I find a difficult problem I can abandon it for some time. This usually helps in gaining a fresh look. Of course this is not a universal recipe, I have papers I will never finish.

Is there a nice anecdote from your career you like to share with our readers?

Mariangiola: My conversations with Corrado were always about science only. The day when I gave birth to my first child, Corrado called me and just after congratulating he asked me something about a joint paper. That was the only time in which I did not try to answer to an interesting question.

Do you have any advice for young researchers? In what should they invest time, what should they avoid?

Mariangiola: To be open in collaborating with people having different backgrounds. This can strongly widen the view.

What are the most important features you look for when searching for graduate students?

Mariangiola: I look for scientific curiosity and open-minded vision.

Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

Mariangiola: In my long career I saw with pleasure many interesting results springing from theoretical computer science. I like to mention the isomorphism between Combinatory Logic terms with intersection types and proofs of the minimal relevant logic ("Intersection types as logical formulas" by Betti Venneri) and a formalisation of communication protocols between distributed peers ("Multiparty asynchronous session types" by Kohei Honda, Nobuko Yoshida, and Marco Carbone).

I must confess I was almost always incapable to predict these developments. Today, being retired and unable to meet colleagues because of the Covid restrictions, it is even more difficult for me. I can only say that I am sure the future of theoretical computer scientists will be full of success.

What kind of opportunities should EATCS offer to researchers, and especially to young researchers?

Mariangiola: The EATCS and also its Italian Chapter played a crucial role in my scientific development. In particular, the EATCS support to the organisation of conferences and workshops allowed my colleagues and myself fundamental scientific exchanges, unfortunately made impossible in the last two years by the pandemic. I am sure EATCS will take again this role as soon as the Covid pandemic will be over.

A big help for all researchers, and in particular for the young ones (usually also those with less research funds), would be a manifest in favour of open access publications without author fees and against the unjustified prices required by publishing houses. Moreover, the authors today prepare their manuscripts in the LaTeX style suggested by the journals, but frequently the typographers use a different style with the result of introducing meaningless breaks inside formulas and sometimes also mistakes to be fixed. This should be avoided with a clear benefit also for the printing costs.

What can be the role of EATCS in solving the challenges of our society?

Mariangiola: One of the main problem today is to fill the gap between the first and the third world. Open access publications without author fees will allow members of all research centres to keep up-to-date and to submit their manuscripts more easily. EATCS can play a fundamental role for the scientists working in the field of theoretical computer science.

Please complete the following sentences?

- *My favorite movie is "The Gospel according to Matthew" by Pier Paolo Pasolini.*
- *Being a researcher made my life full of exciting discoveries.*
- *My first research discovery was looking at the λ-calculus as a paradigm of functional programming languages.*
- *Theoretical computer science in 100 years from now will only appear in open access publications without author fees.*
- *EATCS in 50 years from now will celebrate the 100 year jubilee with many members from all parts of the world.*
- *Love of research and teaching is key to being a happy academic.*

KNOW THE PERSON BEHIND THE PAPERS

Today: Jean-Éric Pin

Bio: *Jean-Éric Pin is a theoretical computer scientist known for his contributions to the algebraic automata theory and semigroup theory. Former head of LITP and LIAFA*[1]*, he is currently an Emeritus CNRS research director at IRIF. Pin is a member of the Academia Europaea (2011) and an EATCS fellow (2014). In 2018, Pin became the first recipient of the Salomaa Prize in automata theory, formal languages, and related topics. He is also the editor of the recent Handbook of Automata Theory.*

We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

J.-É.: I share two photos. The first one was taken in Trois-Rivières (Québec), on Denis Thérien's farm, during the summer of 1980. Howard Straubing is seen

[1]Both research units of CNRS (National Center for Scientific Research), ancestors of IRIF.

standing and giving a presentation. Sitting on the chairs in the foreground, John Brzozowski (right) and myself (left). On the patio, from left to right, Stuart Margolis, Denis Thérien and my daughter Garance, who kindly lent us her children's blackboard. More than forty years later, I have fond memories of that time.

The second photo was taken in 2018 at the Calouste Gulbenkian Museum in Lisbon, where I was trying to convert coffees into new results.

Can you please tell us something about you that probably most of the readers of your papers don't know?

J.-É.: I owe much to Klaus Keimel, a German researcher from Darmstadt who sadly died in 2017. Going back to 1976, I had to attend a series of lectures given by him. By an incredible coincidence, I met him on the suburban train to Paris and we started talking. I had written an article on semigroups, but didn't know what to do next. He not only helped me to ultimately publish this article, but he also directed me to the French School founded by Schützenberger, a decisive input in my career. Later on, in 1996, Klaus would introduce me to Mai Gehrke in Nashville. Years later, this led to a best paper award with Mai and Serge Grigorieff at ICALP 2008.

Is there a paper which influenced you particularly, and which you recommend other community members to read?

J.-É.: Wolfgang Thomas' article *Classifying regular events in symbolic logic* led me to learn enough logic to understand it. Similarly, the articles by Schützenberger and Straubing led me to deepen my knowledge of semigroups and automata. I would also like to recommend two papers: the little known paper by

J. Berstel and L. Boasson, *Towards an algebraic theory of context-free languages* (1996), for its conciseness and elegance and the impressive survey paper by T. Eiter, G. Gottlob, T. Schwentick, *Second-order logic over strings — regular and non-regular fragments* (DLT 2001).

Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

J.-É.: My most recent paper, coauthored with C. Reutenauer, A noncommutative extension of Mahler's interpolation theorem, to be published in the *Journal of Noncommutative Geometry* `https://hal.archives-ouvertes.fr/hal-03579151`. It corresponds quite well to my research in TCS: a combination of algebra, combinatorics, topology and automata theory. It is the culmination of research work begun in 2008 with P. Silva, and it has therefore taken more than twelve years to complete.

When (or where) is your most productive working time (or place)?

J.-É.: It can be at any time, including at night. As for the location, it is rarely in my office, except for collaborative work. I remember having successful ideas on the beach in the isle of Ré, on la Fouly pass in Switzerland while waiting for a bus, or in the kitchen of H. Straubing in Boston.

What do you do when you get stuck with a research problem? How do you deal with failures?

J.-É.: First of all, try to understand the difficulty, and if possible have a battery of examples and counter-examples. Always keep the problem in mind, but do not obsess over it and work on other issues at the same time. Discuss with colleagues, look more carefully on the related bibliography. A new idea may come from thinking again, reading an article or listening to a lecture, even on different topics. It is a bit like a difficult crossword puzzle, where the discovery of a single letter can unlock the situation.

Is there a nice anecdote from your career you like to share with our readers?

J.-É.: I was a member of the national committee for scientific research, and another member of this council was an engineer at Bull, a French computer company. Taking the opportunity of seeing many CVs, he occasionally hired a researcher for Bull. One day, jokingly, I asked him when he would hire me, but to my surprise, I got a concrete job offer as an answer! And so from 1991 to 1993, I joined the Bull research center, where I first met Jean Goubault-Larrecq, a very rewarding experience. In addition, the management training offered by Bull proved to be very useful when, barely back in the academic world, I was appointed director of LIAFA in 2003.

Do you have any advice for young researchers? In what should they invest time, what should they avoid?

J.-É.: It certainly helps to have a solid background in mathematics. However, it is very difficult to predict which branches of mathematics will be useful to you. It is also important to attend regular research seminars or working groups. You should also read the reference articles in depth. The aim is to achieve a level of understanding that makes you as intimate with the article as the author. A final advice is to keep a copy of P.R. Halmos' article "How to write mathematics" on your bedside table. And always remember that any "theorem" less than twenty-four hours old is wrong...

What are the most important features you look for when searching for graduate students?

J.-É.: Apart from the obvious academic criteria, original ideas and a strong motivation for research are excellent indicators.

Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

J.-É.: I am really curious to know how far proof assistants and theorem provers can go in the future. It would be nice if they could prevent people to use hand-waving arguments.

What kind of opportunities should EATCS offer to researchers, and especially to young researchers?

J.-É.: There has been so far only four EATCS young researchers schools. I hope that the health situation will allow many more to be organised. In addition, after creating the Monographs in Theoretical Computer Science almost forty years ago, EATCS could consider creating an audiovisual platform dedicated to theoretical computer science.

What can be the role of EATCS in solving the challenges of our society?

J.-É.: Given the circumstances, working for peace seems to be the most important thing.

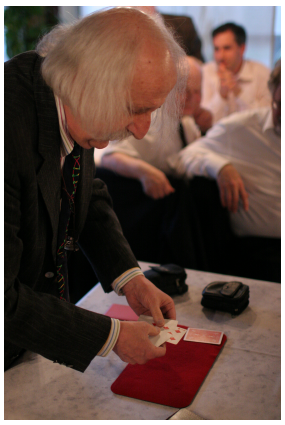
Please complete the following sentences?

- *My favorite movie is ...* Children of Paradise (Les enfants du Paradis) directed by Marcel Carné and written by Jacques Prévert, see `https://en.wikipedia.org/wiki/Children_of_Paradise`. The title is somewhat misleading, since the 'paradis' is the colloquial name for the gallery in a theatre. The heroine of the film is called Garance...
- *Being a researcher...* is a permanent pleasure.
- *My first research discovery...* When I was a child, I tried for years to solve the quadratic equation, but I finally got there. I observed that the equation $x^2+ax+b = 0$ could be written as $x(x+a) = -b$ and I noticed the similarity with a geometry problem: given the area of a rectangle and the difference between its two sides, determine its length and width. This latter question can be easily solved using the identity $(x + y)^2 = (x - y)^2 + 4xy$.
- *Theoretical computer science in 100 years from now...* What a question! A hundred years ago, TCS was limited to algorithms and a bit of computational number theory. Since then, most research topics have been inspired by technological advances, and this creation of totally new fields will certainly continue in the future. To guess the future of the current open questions in TCS, one can try to rely on the only available estimate: Hilbert problems. Of these twenty-three problems, two are considered too vague and one is more of a physics problem, eight are considered solved, eight partially solved and four are still open, including the Riemann hypothesis. This is a rather encouraging result, given the difficulty of the problems. So let us trust the theoretical computer scientists of the future to solve some of the big open questions!
- *EATCS in 50 years from now...* Above all, I hope that EATCS will continue to promote open science. It includes open access to publications and free dissemination of the results, methods and products of scientific research.
- Enjoy research ... *is key to being a happy academic.*

Know the Person behind the Papers

Today: Grzegorz Rozenberg

Bio: *Grzegorz Rozenberg is Professor Emeritus of the Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands and Adjoint Professor of the department of Computer Science, University of Colorado at Boulder, USA. He was EATCS president for 9 years, the editor of the EATCS Bulletin for 23 years, the founding editor-in-chief of the journal Theoretical Computer Science C: Theory of Natural Computing, and a founding editor-in-chief of the International Journal on Natural Computing. Rozenberg is known for his contributions to formal languages and automata theory, concurrency theory, and natural computing. He is sometimes referred to as a guru of natural computing, as he coined the name and defined the scope of this area. He graduated from the Technical University of Warsaw, Poland and obtained his Ph.D. in mathematics from the Institute of Mathematics of the Polish Academy of Science, Warsaw, Poland. Among the prizes he received are: 6 honorary doctorates, the EATCS 2003 award, the first award of the Developments in Formal Languages conference, and in 2017 he was knighted in the Order of the Netherlands Lion.*



We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Grzegorz: There are two pictures I would like to share.

(1) I like giving talks on my research. One can accommodate many more subtleties in a lecture than in a paper. Also, it is easier to introduce someone to a research topic through a lecture than through a "dry" technical paper.

It is a popular "wisdom" that the art of giving lectures is very much like the art of conducting an orchestra: a lecturer must be able to synchronize the minds in the audience on the topic of the lecture. The first picture is made by a member of the audience, when I was giving a lecture on reaction systems, currently my favorite research topic. I was really flabbergasted to see this picture, where it looks like I am, indeed, conducting the "audience orchestra".

(2) I am a performing magician specialized in card magic (no apparatus, just my two hands and a pack of playing cards with no distance from the spectators). Although science is rational and magic emotional, there are many similarities between the two. For example, magic teaches you not to accept things on their face value. This principle also plays a crucial role in science. A goal of the highest level magic is to approximate, as closely as possible, something impossible which is also a key inspiration of the highest level science (for example, disproving a long standing conjecture). My magician's business card says "Be Astonished by the Impossible".

I enjoy to perform at science conferences as this gives me a chance to discuss the nature of magic with scientists (in fact, I performed at several ICALPs). The second picture shows me performing at a conference.

Can you please tell us something about you that probably most of the readers of your papers don't know?

Grzegorz: I wrote above that the double helix, formed by the helices of science and magic, underlies a big part of my creative life. This is known through my papers, books, and talks as well as through my magic performances at scientific events. However there is a third helix.

I have studied the paintings and drawings by Hieronymus Bosch for the past 50 years, and am now an acknowledged specialist on Bosch. By chance, I bought a book with reproductions of his paintings and immediately fell in love with his art. The attractiveness of his art for me is the best described by the statement "it is amazing that a single mind could imagine so many things" made by a Spanish monk and historian, José de Sigüenza, in the 16th century. Interestingly, at the top of one of his well-known drawings there is an inscription in Latin which says "For poor is the mind that always uses the ideas of others and invents none of its own". Quite possibly this was the motto of his workshop. Obviously, it should be a leading motto for researchers.

When reflecting on my long life, I feel very fortunate that the structure of my creative life was determined by the triple helix of science, magic, and Bosch.

Through it I got embedded in three wonderful, but very different communities (scientists, magicians, and art historians).

Is there a paper which influenced you particularly, and which you recommend other community members to read?

Grzegorz: My research directions were influenced by a number of papers and books from various areas of science (mathematics, computer science, biology, linguistics, chemistry and electronics). A theoretical computer science paper that had a big influence at the beginning of my research career is "Finite Automata and their Decision Problems" by Michael Rabin and Dana Scott (see [2]). This paper had an enormous influence on the development of automata theory. It is beautifully written and very inspiring (no wonder, as the authors are real giants of theoretical computer science). In fact, I remember that even during the reading of the paper I started to develop my own ideas on multitape automata.

Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

Grzegorz: Before I recommend a paper, I would like to mention a concern I have had for many years. I never liked the name "computer science", which somehow suggests that this is a discipline centered around (the use and construction of) specific devices, viz., computers. This view was quite prevalent for a long time during my career. Unfortunately (from my perspective), this point of view is sill quite common. For me computer science is *the* science of information processing and in this way it is a fundamental science for many scientific disciplines. Therefore the European term "informatics" is a much better name.

Natural Computing is a good example of this broad understanding of computer science. It is the research area concerned with human-designed computing inspired by nature as well as with computing taking place in nature (i.e., it investigates, in terms of information processing, phenomena taking place in nature). Although a majority of research in natural computing is centered in computer science, it is genuinely interdisciplinary and it forms a solid bridge between computer science and natural sciences.

Research in (both strands of) natural computing constitutes a big part of my research activities. Therefore I would like to recommend the paper "The Many Facets of Natural Computing" by L. Kari and myself (see [1]). Even though the paper was published in 2008 it is still a relevant "nontechnical" introduction to natural computing directed at the general audience of computer scientists. As a follow up, I would like to recommend the "Handbook of Natural Computing", G. Rozenberg, T. Bäck, and J. Kok, editors (see [3]). Just by browsing through it (e.g., through the preface and the table of contents) one can get a sense of the

excitement and relevance of this area as well as the understanding of its enormous importance for the development of computer science.

When (or where) is your most productive working time (or place)?

Grzegorz: I am not well organized in this respect, i.e., I do not have specific times of the day reserved for research. Sometimes it is a whole day when I work on research problems and sometimes it is just plugged into "free slots" during a busy day. My favourite writing places are my home and cafes. I always carry a writing pad with me. In the office I work with my collaborators, students, . . . – for this I need a blackboard (or, nowadays, a whiteboard).

What do you do when you get stuck with a research problem? How do you deal with failures?

Grzegorz: I am always working on a number of research problems in an interleaving fashion, meaning that when I work on a specific problem, my notes on the other problems are set aside. So when I stop working on the given problem (e.g., because I feel that I do not make enough progress on it), the notes on it are set aside and I pick up one of the problems in the waiting line – sometimes it is a problem I set aside a short time ago and sometimes it is a problem from years ago.

I do not accept the negative term "failure" used in the question. It is the process of working on interesting (a subjective term) problems, that makes the life of an active researcher so exciting. Moreover, if you do not succeed in "solving" a problem that you chose to work on, you often learn a lot, produce new interesting notions and/or results, and as a result of this experience start a new research line.

Is there a nice anecdote from your career you like to share with our readers?

Grzegorz: Here is an anecdote I find hilarious, which is well-known among my friends, scientists and magicians. It shows how the perception and appreciation of various professions may be age dependent. One day, when my son, Daniel, was a teenager, I got back home from the office and entered our house. Daniel was then in the hall with his friend, Ferdie. I said "hello", and went to the kitchen. Since the door from the hall was open, I could hear Ferdie asking Daniel about my profession. Daniel answered "he is a university professor" and a moment later he added: "but he is not stupid, he is a very good magician".

Do you have any advice for young researchers? In what should they invest time, what should they avoid?

Grzegorz: Invest your time and energy in following your scientific curiosity and passion *but* avoid to be "chained" to one specific line of research. Try to get some insight into various research areas (e.g., by reading tutorials or following

schools). This may lead to working on different sorts of research issues, which is also good for your intellectual development.

What are the most important features you look for when searching for graduate students?

Grzegorz: Clearly, there must be a "proof" (the master thesis, study grades, presentations) that they are intellectually qualified. But then, I always look for signs of curiosity, passion, and motivation which will allow them to be successful in getting good results and happy with working on them.

Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

Grzegorz: Computer science is evolving towards *the* science of information processing and is accepted as such by many scientific disciplines. As an example of this acceptance I can quote the famous biologist Sydney Brenner, Nobel prize winner, who stated that "biology is essentially (very low energy) physics with computation". For many disciplines computer science provides not only instruments but also a way of thinking. Therefore, the science part of computer science will become a foundational science for many areas of science.

I am convinced that one of the Grand Challenges of computer science is to understand the world around us in terms of information processing. Working on this challenge is a huge opportunity as it also enriches the scope of research problems for theoretical computer science. For example, theoretical computer scientists are already making essential contributions to the fundamental understanding of self-assembly, a central phenomenon of nanoscience.

What kind of opportunities should EATCS offer to researchers, and especially to young researchers?

Grzegorz: In the spirit of my reflections above, I would like to suggest that EATCS gets involved in organizing "broad perspective schools" which would cover, in a tutorial fashion, currently interesting developments in various areas of computer science. By attending such schools, young researchers will get an opportunity to broaden their vision of theoretical computer science and (hopefully) get actively involved in new research directions.

What can be the role of EATCS in solving the challenges of our society?

Grzegorz: In the context of theoretical computer science, our research advances contribute to the foundations of understanding nature, science, and new technologies, all of which will impact lives of people around the world.

The role of EATCS should be extending the scope of theoretical computer science and promoting interdisciplinary research. Extending the scope of research

widens our opportunities to make a progress in a multitude of issues important for our society. The scope of these opportunities is certainly broadened by promoting interdisciplinary research.

Please complete the following sentences?

- *My favorite movie is* ... "Blow-up" by Michelangelo Antonioni.
- *Being a researcher* ... is a wonderful way of living.
- *My first research discovery* ... was in category theory.
- Clever juggling of research, teaching, and administration and a feeling of success/satisfaction in at least one of those areas ... *is key to being a happy academic.*

References

[1] Lila Kari, Grzegorz Rozenberg. The many facets of natural computing. Commun. ACM 51 (2008) 72–83. doi: 10.1145/1400181.1400200

[2] Michael O. Rabin, Dana S. Scott. Finite Automata and Their Decision Problems. IBM J. Res. Dev. 3 (1959) 114–125.

[3] Grzegorz Rozenberg, Thomas Bäck, Joost N. Kok. Handbook of Natural Computing. (4 volumes) Springer 2012. doi: 10.1007/978-3-540-92910-9

KNOW THE PERSON BEHIND THE PAPERS

Today: Robert Cori

Bio: *In 1967, Robert Cori was a doctoral student of Marco Schützenberger joining a pioneering research team in Paris where, among others, J. Berstel, M. Nivat, J.-F. Perrot created the French School of theoretical computer science. His first subject in research was the use of context-free languages as a tool for getting enumerative results for families of graphs. After defending his thesis he obtained a professorship in the University of Bordeaux. There computer science was just a tiny bud in the math department. The bud grew after the arrival of B. Courcelle, X. Viennot and A. Arnold in the next years. In 1982 he was in charge of creating a French collaborative program named Greco de Programmation, this helped to attract people and funds in Bordeaux and build the LaBRI. In 1992, he obtained a part-time teaching position at the Ecole Polytechnique in Paris, which he kept for 15 years. He rubbed shoulders with many computer science researchers there, including J.J. Levy, Claire Mathieu, P. Rosenstiehl and supervised a few doctoral smart students. He is Emeritus Professor at the University of Bordeaux since 2009.*

We ask all interviewees to share a photo with us. Can you please tell us a little bit more about the photo you shared?

Robert: I share two photos. The first above was taken in June 2009 during a three-day meeting organized for my 65th birthday. It also corresponded to the end of my teaching years. The place is in front of the LaBRI building at the University of Bordeaux. On this photo you may recognize researchers who were my doctoral students, some colleagues from LaBRI, others in Paris or Italy, and some Master's students from Polytechnique who came to Bordeaux for the occasion.

The second picture is taken in 1988 in Paris 7 University, I am sitting between Marco Schützenberger and Dominique Perrin during a meeting of a thesis jury.

Can you please tell us something about you that probably most of the readers of your papers don't know?

Robert: Readers of my articles probably don't know that I view research primarily as a kind of game that has allowed me to play with many co-authors and many very brilliant PhD students. However, although teaching is considered by many scholars to be the second most important duty after research, I have devoted time to it as my first priority. It is not easy at certain times to obtain psychological well-being by only doing research, on the contrary I have always achieved it by giving well-prepared courses in front of brilliant students.

Is there a paper which influenced you particularly, and which you recommend other community members to read?

Robert: I recommend many articles by Deepak Dhar which are motivated by describing physical phenomena in mathematical terms which can be very close

to theoretical computer models like finite automata or cellular automata. These articles can give rise to many developments thinking the matter inside as models in theoretical computer science. One of them is *Theoretical studies of self-organized criticality*, which inspired my research, I suspect that many others published by him more recently may give rise to pertinent questions about automata and graphs which may open up new problems in these subjects.

Is there a paper of your own you like to recommend the readers to study? What is the story behind this paper?

Robert: I recommend the paper we wrote with Dominique Rossin entitled *"On the Sandpile group of Dual graphs"*, it appeared in 2000 in the European Journal of Combinatorics. A first version of this paper was a preprint written in 1998 which was available at the Computer Science Lab in Polytechnique. We thought with Dominique that we were the first to give an algebraic presentation of the Sandpile model introduced by Dhar. In fact this was not correct since the model was also considered by N. Biggs with the Chip Firing Game introduced by Björner, Lovasz and Schor. So that the publication of our paper was only possible since it contained a special result on dual planar graphs. However our paper was read by Don Knuth who was inspired to find an algorithm generating all spanning trees of a graph. He presented this algorithm and made a few presentations of ours around the world giving a good advertising for the paper. He also delivered a talk on our version of sandpiles and spanning trees in Stanford for his eleventh annual Christmas Tree Lecture in December 2004.

When (or where) is your most productive working time (or place)?

Robert: When I work alone the best is very early at home in my office. But I also very often try to chat with others at any time in front of a blackboard with chalk in the past or now whiteboards with markers.

What do you do when you get stuck with a research problem? How do you deal with failures?

Robert: I do not consider as a failure to spend time on a research problem, if I did not succeed to solve an open problem I consider that I learned a lot while reading the papers of other researchers working on it.

Is there a nice anecdote from your career you like to share with our readers?

Robert: When I started my thesis with Marco Schützenberger as a referent, I occupied a shared office at the Institut Blaise Pascal in Paris. The building was left by an abandoned factory and was located in the far north of Paris. There was no space available in the *Quartier Latin* for the huge computer dedicated to the Department of Applied Mathematics of the Faculty. The Henri Poincaré

Institute where Algebraic Geometry and Differential Analysis flourished could not accommodate offices for Theoretical Computer Science.

I shared an office with a researcher working in Artificial Intelligence, his daily behavior puzzled me. He walked into the office every day with a huge deck of punched cards. It contained the chess player's algorithm program he had designed. He had taken it to the computer center before coming to the office with also a quantity of paper listing the outputs of the execution of the program on the computer. It was the program's response to the motion it gave as input to the program. Then, his concern was to analyze this answer and to proceed to the improvement of the program.

This led him to remove some cards from the deck to go punch many others, add them to the game and return it to the computer center. He often had the opportunity to repeat this process once in the afternoon. Because I was reluctant to chat with a researcher ten years my senior, and he was very busy, we didn't talk much. This behavior did not make me consider changing the subject of my research. The discussions with Marco on the theory of languages were a far superior attractor.

Do you have any advice for young researchers? In what should they invest time, what should they avoid?

Robert: I don't have very original advice on this subject. I think that one must above all avoid remaining alone on a difficult question. So a young researcher should ask for another direction of research if the current one seems to him intractable.

What are the most important features you look for when searching for graduate students?

Robert: Since I am emeritus I am not allowed to advise PhD students. In the past I often hired those students who attended my master courses or were interested in a talk I gave in a conference or in a seminar.

Do you see a main challenge or opportunity for theoretical computer scientists for the near future?

Robert: It suffices to consult recent scientific journals devoted to this field to see the number of interesting open problems. Applications of theoretical computer science to other sciences have great developments.

What kind of opportunities should EATCS offer to researchers, and especially to young researchers?

Robert: The surveys and tutorials in EATCS journal are often excellent texts helping to introduce new subjects to PhD students. Conferences should not be

limited to hundreds of people each one speaking less than 20 minutes of his last result. ICALP seems to go in the good direction by proposing many invited talks.

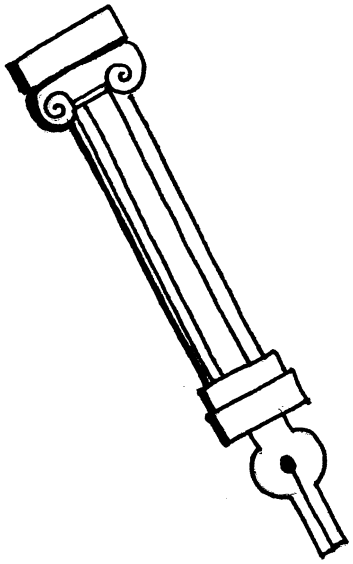
What can be the role of EATCS in solving the challenges of our society?

Robert: This is a big question that deserves long reflection and discussion with those responsible for education. EATCS may just promote minimal knowledge of what an algorithm really is and what a machine can actually learn.

Please complete the following sentences?

- *My favorite movie is...* Amadeus by Milos Forman, a brilliant demonstration that happy people are pleased by the happiness of others and the miserable are poisoned by envy.
- *Being a researcher...* is a wonderful gift given to you.
- *My first research discovery..* allowed me to cross for the first time the Atlantic in order to present it in a conference and to meet William Tutte.
- *Theoretical computer science in 100 years from now ...* will concern a few people working in a hot planet.
- *EATCS in 50 years from now will...* still help researchers to understand what is real progress and what is just a gimmick .
- Considering research and teaching as essential activities and do not spending too much time on administrative tasks *...is the key to being a happy academic.*

EATCS
Columns



The Education Column

BY

Juraj Hromkovič and Dennis Komm

ETH Zürich, Switzerland and PH Graubünden, Chur, Switzerland
juraj.hromkovic@inf.ethz.ch and dennis.komm@phgr.ch

BEATCS no 137

A Minimal Instruction Set for Education

Tobias Kohn
University of Utrecht
t.kohn@uu.nl

Abstract

Computer science education is often caught between the desire to teach algorithmics and the necessity to build a solid basis of programming to properly implement algorithms. We thus raise the question of how much programming we need before we can start tackling interesting computational problems and demonstrate how important computational concepts emerge almost naturally with a minimal set of programming constructs needed.

1 Introduction

The quest of a minimal instruction set is an old and solved problem. How many instructions does a computing system need in order to be Turing complete? In a nutshell, combine increment/decrement with a while-loop and you are good to go (with numerous alternatives existing, of course, some rather surprising [1, 3]).

Let us consider the same question from a different perspective, though. Picture a group of students with virtually no prior experience in programming. Your objective is to enable them to solve 'interesting' computational problems. How interesting? Think, e.g., 'graph algorithms' or 'Project Euler' [4] kind of problems. Your target audience starts from scratch and you want to minimise the time needed for teaching programming and get to solve problems as quickly as possible.

In this setting the set of minimal instructions is no longer dictated by pure logic and a desire for Turing completeness. Rather, psychological factors quickly dominate the discussion. For instance, as trained professionals we would hardly seek to include multiplication into a minimal computing instruction set, whereas we can safely include it into our education setting because students are already highly familiar with it and the necessary amount of teaching is practically null.

On the flip side, we find that the modulo or remainder operation is surprisingly expensive. Even though division and computing the remainder might even be one and the same underlying program running on an actual machine, the modulo operation is significantly harder to understand, apply and hence teach.

So, can we discuss core concepts of computer science with merely a handful of well chosen, 'easily teachable' instructions and programming constructs?

Much of what follows is highly reminiscent of functional programming, although it is not. Our contemplation is based on imperative programming with Python. We embrace its lists as mutable data structures and use iteration instead of recursion. The similarity to functional programming might arise due to two factors: a limited and cautious use of variables and avoidance of list indices.

2 From Code to Data and Back Again

We usually start our introductory programming courses with turtle graphics. The virtual turtle robot is a simple and concrete device that leaves a trace so as to draw figures. Both the set of its capabilities as well as its purpose are thus immediately obvious and in stark contrast to full-blown computing devices with their unbounded complexities.

The set of initial commands consists essentially of `forward(s)` and `back(s)` for movement, and `left(a)` and `right(a)` for turning (on the spot). When talking about finding a minimal set of instructions half of these four instructions are evidently superfluous—even more so when considering that `left(a)` can equally be written as `right(-a)`. In fact, a single unified command that moves the turtle one step forward and turns it by a given angle might suffice.

This example neatly demonstrates how 'engineering minimalism' differs from 'educational minimalism'. The commands `left(a)` and `right(a)` express the same concept and, moreover, directly reflect concepts from the students' everyday experience. Reducing this to a single `turn()` command would rather increase the difficulty as the student would also have to consider which (arbitrary) direction is meant by a 'positive' angle.

The idea of a single unified command comes up later in the course when we introduce for-loops. While for-loops lend themselves to a wide variety of applications, one that stands out is the representation of figures (e.g., draw by the turtle) through data. The introduction of the turtle naturally leads to programs that build intricate figures out of long sequences of instructions. By means of a for-loop and a list, we can now abstract from those long sequences of instructions and retain the data part of them only. The only problem is that the sequences of instructions alternate between movement and rotation. As a first step we therefore have to keep one of these fixed, i.e., stipulate that all rotations are replaced by a left-turn by 45° with zero-movement interspersed where necessary, or, alternatively, keep all forward instructions to ten steps, say (Program 1).

Luckily, Python allows us to create lists of tuples very easily so that we can include an argument for both the forward motion and the angle of rotation. Nonethe-

Program 1 Two ways of drawing a square using a simple for-loop.

```
for a in [90, 90, 90, 90]:
    forward(10)
    left(a)
```

```
for s in [10, 0, 10, ..., 0]:
    forward(s)
    left(45)
```

less, from an abstract point of view, we end up with a 'single instruction' (or a single parametrised action if you prefer), something like `move_and_turn(s, a)`. Alternatively, you might also choose to use coordinates along with the instruction `setpos(x, y)` as the basis for all drawings.

Hence, the step from an instruction- to a data-based format brings us back to the necessity of having a single instruction and therefore considering an extremely minimal set of instructions that suffice to draw as many figures as possible. This example also illustrates the power that comes from single-instruction machines. The extremely regular interface allows for the introduction of a new level of abstraction.

As a perhaps final step along this ladder we might wish to differentiate further between possible actions. So, let us move away from the idea of a single instruction that churns through a list of data and enrich the data with (textual) hints as to which action to perform.

For the sake of simplicity, let us stick with the commands we have already introduced. A minimalistic program might then look as shown in Program 2. If you look closely you will discover that we have come full circle here. The 'data' in the list we are processing is in fact program code in what looks very much like a dialect of Lisp or Logo [5].

Program 2 A simple interpreter for a Logo-like language.

```
for (cmd, arg) in [('fd', 50), ('rt', 90), ('fd', 20)]:
    if cmd == 'lt':
        left(arg)
    if cmd == 'rt':
        right(arg)
    if cmd == 'fd':
        forward(arg)
```

So what did we gain if we end up still writing our drawings as program code, only now with the additional overhead of the for-loop that executes the instructions? We have stumbled on one of the most fundamental ideas in computer science: the 'equivalence' of data and code. Replace the textual instructions above with numeric constants and you are but a small step away from Gödel numbers.

To put it differently, at this stage our students have written their first interpreter. Their programs no longer draw specific pictures, but are in principle capable of

running any other program that draws a picture. Although this is far away from Turing completeness and lacks most attributes of an actual computer, we nonetheless have just introduced the concept of a *universal machine*.

3 Iterating Over Lists

Our journey from sequences of instructions to data processing above was made possible by the use of lists and for-loops.[1] It is quite natural, of course, that abstracting from code to data meant that a sequence of instructions turns into a sequence of data. Moreover, Python's lists are very versatile data structures that can take the role of arrays, lists, stacks, sets or even maps. Let us therefore spend some time taking a closer look at Python's lists.

True to the overall topic of this article we argue that a set of three list operations, namely *iteration through a list*, *appending an element to a list* and *testing for inclusion* (i.e., whether a list contains a specific element) is sufficient for tackling a range of interesting problems. Most importantly, all three of these basic operations are conceptually simple enough to be integrated into teaching relatively early on.

The choice to use for-loops and lists comes at a price, though. The resulting loops are clearly bounded and we forgo Turing completeness (i.e., we only have primitive recursion and cannot implement partial recursive functions). On the other hand, considering that educational problem instances are typically quite small and by choosing sufficiently large numbers, we still achieve a working approximation to Turing completeness.

Lists in Python. Lists in Python are implemented as arrays that can grow in size. Accessing elements (both retrieval and modification) are thus in $O(1)$ and appending elements is in amortised $O(1)$ [6]. Using for-loops together with 'append' allows us to easily implement 'map' and 'filter' as shown in Program 3. Naturally, writing 'reduce' in Python is not much more difficult and follows the same pattern as 'map' as shown in Program 4.

There is direct syntactic support for checking whether an element occurs in a list (which is performed through a linear $O(n)$ search). Although this could also be done with a 'reduce' pattern, we include it in our 'instruction set' as it allows us to conveniently use lists like sets with minimal teaching effort.

Based on the basic elements of 'filter', 'map' and 'reduce' we can implement functions like 'pop' and 'cons' (Program 5) known from functional programming.

[1] NB: Python's for-loop corresponds to 'for each' loops in other languages and does not have the generality of for-loops in C, say.

Program 3 Implementations of 'filter' and 'map' in Python.

```
def filter(p, lst):
    mod_lst = []
    for i in lst:
        if p(i):
            mod_lst.append(i)
    return mod_lst
```

```
def map(f, lst):
    mod_lst = []
    for i in lst:
        mod_lst.append( f(i) )
    return mod_lst
```

Program 4 A generic implementation of 'reduce' or 'fold' and a more concrete instance of 'sum'.

```
def reduce(f, lst, init = 0):
    acc = init
    for i in lst:
        acc = f(acc, i)
    return acc
```

```
def sum(lst):
    acc = 0
    for i in lst:
        acc += i
    return acc
```

Variables and conditionals. Apart from the three list processing primitives introduced above we also make use of variables and conditional execution, thereby adding two further concepts. Furthermore, the code examples also use function definition and the return-statement, but those are not necessarily needed when using and implementing the presented concepts with students.

The most problematic aspect of these constructs is variable assignment, in particular updating a variable's value. To mitigate this issue we restrict variable modification to a small set of operations such as `acc += x` or `acc *= 2` etc.

Instead of 'append' we could also have gone for list concatenation along the lines of `list = list + [item]` or its short form `list += [item]`. At first glance this seems not only to be more powerful, but also easier as you do not need a special function, but rely on the concept of 'adding' instead. However, there are some subtle rules to consider for variable assignments in Python in terms of scoping. Python infers the scope of variables from context, unless explicitly de-

Program 5 The 'pop' function to split a list into its head and tail is a combination of 'filter' and 'reduce' from above whereas 'cons' follows a 'reduce' pattern.

```
def pop(lst):
    head = None
    tail = []
    for item in lst:
        if head == None:
            head = item
        else:
            tail.append(item)
    return head, tail
```

```
def cons(head, tail):
    result = [ head ]
    for i in tail:
        result.append(i)
    return result
```

clared as global, which adds a large and complex topic to the teaching curriculum. In short, using 'append' avoids some common pitfalls, helps us reduce the use of variable (re)assignments and is therefore in line with our aim to strive for a minimum set of instructions from an educational perspective.

Implementing Dijkstra's algorithm. As a proof of concept we present an implementation of Dijkstra's algorithm for finding the shortest path in an (undirected) graph in Program 6. The graph itself is defined close to its mathematical formulation with two lists representing vertices and edges. However, the actual vertex coordinates are used in the edge set, too, so as to avoid the use of indices.

Program 6 An implementation of Dijkstra's algorithm in Python.

```
vertices = [ (0, 0), (10, -2), (1, 15), (10, 10), ... ]
edges = [ ( 7, (0, 0), (10, -2)), (14, (0, 0), (1, 15)), ... ]
visited = [(0, 0)]
visited_dists = [((0, 0), 0)]

def dijkstra():
    for k in range(1, 50):
        for (anchor, cur_dist) in visited_dists:
            for (w, p1, p2) in edges:
                if cur_dist + w == k:
                    if anchor == p1 and p2 not in visited:
                        (x, y) = p2
                        visited.append((x, y))
                        visited_dists.append(((x, y), k))
                    if anchor == p2 and p1 not in visited:
                        (x, y) = p1
                        visited.append((x, y))
                        visited_dists.append(((x, y), k))
```

The algorithm presented here works without queues, minima or relaxation. This is achieved by iterating over possible path lengths (k in Program 6), i.e., first establishing all paths with length 1, then with length 2, etc. Once a vertex has been added to the 'visited' set, no other path can reach it with a shorter path length.

On the flip side, we achieve this 'simplification' and reduction to our minimal set of instructions by sacrificing performance. During each iteration for a specific path length, we iterate over all edges to find suitable candidates. For larger graphs, this is clearly not practical. However, the graphs used in education tend to be small enough, particularly when considering that we aim to write such programs as early in the curriculum as possible.

Although we have also implemented Prim's algorithm for finding a minimum spanning tree with a similar approach, we omit it here for the sake of brevity.

However, note that these algorithms frequently form a basis for solving 'interesting' graph problems (see, e.g., Skiena and Revilla [7]).

4 Accessing Items by Index

The attentive reader will have noticed that we have not made use of Python's facilities to directly access any element in the list by its index, although we mentioned $O(1)$ support. Moreover, the syntax for indexed access includes slices with custom strides. Retrieving every other element from a list `L` is thus as simple as `L[::2]` and getting a list with its elements in reversed order is `L[::-1]`.

Experienced programmers find thus a very convenient and simple syntax in Python's slices. Novice programmers, on the other hand, tend to struggle with the concept of accessing an element by its index. Consider: the syntax for building lists and stating its member elements almost coincides with the syntax for accessing an element by its index:

```
primes = [2, 3, 5, 7, 11, 13, 17, 19]
print( primes[3], primes[4] )
```

Given the syntactic similarity we should not be surprised that some students explain the above code as: 'First prints the element 3 because it is actually in the list and then throws an error or something because 4 is not in the list' (cf. Boulay [2]).

At this point we have not even touched upon the famous issue of starting the indexing with zero and the 'off-by-one' problems this quickly leads to. How quickly can you tell which elements are to be found in the slice `primes[2:5]`? It is the three numbers $5, 7, 11$—the number 13 at index 5 is excluded as Python's 'intervals' always include the first but exclude the last element.

Dealing with a lack of indices. We recently asked students as part of a programming challenge to determine whether a given short sequence occurs in a longer list of numbers. For instance, the sequence $2, 4$ does indeed occur in the following list:

```
[1, 4, 2, 5, 2, 2, 4, 7, 2, 3, 4]
```

With the tools provided and, in particular, the absence of indices and slices it may seem very hard to solve this challenge for a general case. Perhaps the canonical approach is to build a finite state machine for the short sequence—an approach, however, that does not scale well with the length of the sequence to find.

Program 7 demonstrates an alternative solution with nothing more than the three basic list operations introduced earlier. The bulk of the program consists in listing every possible sequence of the correct length occurring in the given list. The actual test whether the sequence then occurs happens in the very last line.

The approach in Program 7 can equally be used to solve a challenge posed as part of the Swiss Olympiad in Informatics competition 2021/22 [8, 'Peaks'].

Program 7 Determining whether a list contains a given sequence.

```
seq = [2, 4]
numbers = [1, 4, 2, 5, 2, 2, 4, 7, 2, 3, 4]
part_lists = []
for num in numbers:
    part_lists.append([])
    for lst in part_lists:
        if len(lst) < len(seq):
            lst.append(num)
print(seq in part_lists)
```

Given a sequence of numbers count the number of local maxima, i.e., how often a number N_i is larger than its two immediate neighbours: $N_i > N_{i-1}$ and $N_i > N_{i+1}$. The follow-up challenge asks the student to find a subsequence of length K with the highest amount of local maxima in it. This can, again, easily be solved using the approach introduced above.

As before, however, we also have to concede that there are far superior approaches in terms of performance. That is, our solution is probably not fit to compete in such contests, but again demonstrates that the tools provided suffice to solve 'interesting' problems.

5 Turing's Return

After having implemented searches in graphs and sequences, let us tackle another classic problem: *sorting*. At this point, it should be fairly obvious what minimum sort would look like. So, let us focus on *merge sort* instead.

When implementing *merge sort* we have to deal with two challenges: merging two lists and the recursive structure of the problem decomposition. Merging two lists is difficult because we have to advance the iteration in both list separately and non-uniformly. However, Program 8 shows how we may use the `pop()` function from above to address this and perform explicit iteration.

While merging two lists without list indices is challenging from a technical point of view, we may find ourselves more interested in the main function where we have to deal with the recursive 'tree' structure of the problem. Our approach uses a 'todo' buffer containing all the already sorted lists. The algorithm then picks two lists from this buffer, merges them and replaces the original two lists by the merged one until only one unified list remains. With only an 'append' operation for lists, it may not be immediately obvious how to do this.

Luckily, Python's for-loop iterator and the 'append' function are compatible with each other. We may append elements to the list we are currently iterating over and the for-loop will correctly pick up these additional elements. This gives us a neat *queue* data structure that acts as our buffer (Program 9).

Program 8 Merging two lists for *merge sort* requires 'explicit iteration'.

```
def merge(lstA, lstB):
    result = []
    (a, tailA) = pop(lstA)
    (b, tailB) = pop(lstB)
    for _ in range(len(lstA) + len(lstB)):
        if b == None or (a != None and a <= b):
            result.append(a)
            (a, tailA) = pop(tailA)
        else:
            result.append(b)
            (b, tailB) = pop(tailB)
    return result
```

Program 9 The core function of *merge sort* where we use the list `todo` as a queue. By both iterating over `todo` and simultaneously appending elements to it, we effectively end up with a while-loop.

```
def merge_sort(lst):
    todo = []
    for k in lst:
        todo.append( [k] )
    first = None
    for item in todo:
        if first == None:
            first = item
        else:
            todo.append( merge(first, item) )
            first = None
    return first
```

Because we may extend the list we are iterating over inside the loop, we have a tool to control whether the loop shall terminate at any one point. More importantly, though, we can keep the loop going for an indefinite amount of time! Let us reiterate this point: although Python's for-loops can only iterate over lists, they are still as powerful as while-loops (when combined with if-conditions, of course). In other words, the small set of instructions we chose is Turing complete after all.

6 Conclusion

What a ride! By drawing figures with a turtle we accidentally wrote an interpreter and stumbled on a data-code-equivalency. Moreover, by restricting ourselves to iteration and simple list operations, we implemented our own while-loop and dis-

covered that our minimal set of instructions is Turing complete after all.

All this was achieved with a minimal set of instructions. It is thus perhaps tempting to interpret these ideas so as to de-emphasise the teaching of programming. However, using the small set of programming constructs the way we have in this article requires training in how to use and apply them as well as a thorough and solid understanding of their effects. In reality, this is very difficult to achieve, but puts the focus of learning on the right topics and concepts.

In fact, there is a tendency in programming courses to introduce a wealth of programming constructs early on with endless listings of available functions, methods and data types. Yet, if we are interested in core concepts of computer science, in algorithmics and computational thinking, we might actually be better off with a minimalistic set of instructions and data types. We would even argue that a minimalistic instruction set automatically places much more emphasis on algorithm design and problem solving. On any account, we have demonstrated that core concepts of computer science emerge almost naturally even with such a minimal set of instructions. Programming is not only a prerequisite for implementing algorithms but also a fertile ground for discovering and discussing algorithms.

References

[1] M. Böhme and B. Manthey. The computational power of compiling C++. *Bulletin of the European Association for Theoretical Computer Science*, 81:264–270, 2003.

[2] B. Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2(1):57–73, 1986.

[3] S. Dolan. mov is Turing-complete (2013).

[4] C. Hughes et al. Project Euler. `https://projecteuler.net`.

[5] S. Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc., USA, 1980.

[6] Python Wiki: Time Complexity. `https://wiki.python.org/moin/TimeComplexity`.

[7] S. S. Skiena and M. A. Revilla. *Programming Challenges*. Springer, 2003.

[8] Swiss Olympiad in Informatics. `https://soi.ch`.

The Computational Complexity Column

by

Michal Koucký

Computer Science Institute, Charles University
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
koucky@iuuk.mff.cuni.cz
https://iuuk.mff.cuni.cz/~koucky/

BEATCS no 137

Theory and Applications of Probabilistic Kolmogorov Complexity

Zhenjian Lu* Igor C. Oliveira†

Abstract

Diverse applications of Kolmogorov complexity to learning [CIKK16], circuit complexity [OPS19], cryptography [LP20], average-case complexity [Hir21], and proof search [Kra22] have been discovered in recent years. Since the running time of algorithms is a key resource in these fields, it is crucial in the corresponding arguments to consider *time-bounded* variants of Kolmogorov complexity. While fruitful interactions between time-bounded Kolmogorov complexity and different areas of theoretical computer science have been known for quite a while (e.g., [Sip83, Ko91, ABK+06, AF09], to name a few), the aforementioned results have led to a renewed interest in this topic.

The theory of Kolmogorov complexity is well understood, but many useful results and properties of Kolmogorov complexity are not known to hold in time-bounded settings. Unfortunately, this creates technical difficulties or leads to conditional results when applying methods from time-bounded Kolmogorov complexity to algorithms and complexity theory. Perhaps even more importantly, in many cases it is desirable or even necessary to consider *randomised* algorithms. Since random strings have high complexity, the classical theory of time-bounded Kolmogorov complexity might be inappropriate or simply cannot be applied in such contexts.

To mitigate these issues and develop a more robust theory of time-bounded Kolmogorov complexity that survives in the important setting of randomised computations, some recent papers [Oli19, LO21, LOS21, GKLO22, LOZ22] have explored *probabilistic* notions of time-bounded Kolmogorov complexity, such as rKt complexity [Oli19], rK^t complexity [LOS21], and pK^t complexity [GKLO22]. These measures consider different ways of encoding an object via a *probabilistic representation*. In this survey, we provide an introduction to probabilistic time-bounded Kolmogorov complexity and its applications, highlighting many open problems and research directions.

*University of Warwick, UK. E-mail: zhen.j.lu@warwick.ac.uk
†University of Warwick, UK. E-mail: igor.oliveira@warwick.ac.uk

1 Introduction

Consider an arbitrary binary string $x \in \{0,1\}^*$, e.g.,

$$x = 10101010101010101010111100101100000101011\,. \tag{1}$$

The Kolmogorov complexity of x, $\mathsf{K}(x)$, is the length $|M|$ of the shortest program M that prints x when computing over the empty input string.[1] Intuitively, $\mathsf{K}(x)$ can be seen as a measure of the "randomness" of x, in the sense that simple strings exhibiting an apparent pattern have bounded Kolmogorov complexity (e.g., the leftmost 20 bits of the string x from Equation (1)), while a typical random n-bit string has $\mathsf{K}(x)$ close to n, i.e., it cannot be compressed. The investigation of Kolmogorov complexity has uncovered surprising connections to distant areas of mathematics and computer science, ranging from computability, logic, and algorithm design to number theory, combinatorics, statistics and a number of other fields. We refer to [SUV17, LV19] for a comprehensive treatment of Kolmogorov complexity and its applications.

Despite the appealing nature and wide applicability of Kolmogorov complexity, its results and techniques tend to be inappropriate in settings where the *running time of algorithms is of concern*, e.g., in complexity theory, computational learning theory, and cryptography. This is because $\mathsf{K}(x)$ does not take into account the time that the machine M takes to output x. To address this issue, several authors have contributed to the development of *time-bounded* Kolmogorov complexity. In order to proceed with our discussion, we describe two prominent time-bounded Kolmogorov complexity notions. (A formal treatment appears in Section 2.)

In an influential paper, Levin [Lev84] introduced $\mathsf{Kt}(x)$, a variant of Kolmogorov complexity that simultaneously takes into account the running time t and description length $|M|$ of all programs M that output x. More precisely, given a string $x \in \{0,1\}^*$, we let

$$\mathsf{Kt}(x) = \min_{M,\, t \geq 1} \{|M| + \lceil \log t \rceil \mid M \text{ outputs } x \text{ in } t \text{ steps}\}\,. \tag{2}$$

To provide intuition and give a concrete example of the usefulness of this time-bounded variant of Kolmogorov complexity to algorithms and complexity theory, we consider the following computational problem at the intersection of mathematics and computer science:

Explicit Construction of Primes: Given an integer $n \geq 2$, deterministically compute an n-bit prime number.[2]

[1] We formally define (time-bounded) Kolmogorov complexity in Section 2.

[2] For instance, the string x in Equation (1) is a 40-bit prime (733008047147 in decimal representation).

The fastest known algorithm that solves this problem runs in time $\widetilde{O}(2^{n/2})$ [LO87], and it is a longstanding open problem to improve this bound (see [TCH12]). Let $A(n)$ denote this procedure, and consider the sequence $\{p_n\}_{n\geq 2}$ of primes output by $A(n)$. Since we can encode the fixed algorithm A using $O(1)$ bits and any fixed number n using $O(\log n)$ bits, it follows that some program M of description length $O(\log n)$ runs in time $t = \widetilde{O}(2^{n/2})$ and prints p_n. Consequently, there is an n-bit prime p_n such that $\mathsf{Kt}(p_n) \leq O(1) + O(\log n) + \log t \leq n/2 + O(\log n)$. More generally, a faster algorithm yields improved bounds on the Kt complexity of some sequence of prime numbers. Conversely, it is possible to prove that if there is a sequence $\{q_n\}_{n\geq 2}$ of n-bit primes such that $\mathsf{Kt}(q_n) = \lambda_n$, then the problem of explicit constructing primes can be solved in time $\widetilde{O}(2^{\lambda_n})$.[3] This shows that one can completely capture the problem of explicitly constructing primes via time-bounded Kolmogorov complexity!

Note that in Kt complexity the time bound is not fixed and depends on the best possible description of x. In some contexts, it is desirable to restrict attention to programs M that run under a specified time bound $t(n)$, e.g., in time $\leq n^3$. This is captured by K^t complexity (see, e.g., [Sip83]), where $t\colon \mathbb{N} \to \mathbb{N}$ is a fixed function:

$$\mathsf{K}^t(x) = \min_M \{|M| \mid M \text{ outputs } x \text{ in } t(|x|) \text{ steps}\}. \tag{3}$$

As a recent application of time-bounded Kolmogorov complexity, Liu and Pass [LP20] connected one-way functions (OWF), a primitive that is essential to cryptography, to the computational difficulty of estimating the K^t complexity of an input string x, when t is a fixed polynomial. A bit more precisely, they showed that OWFs exist if and only if it is computationally hard on average to estimate $\mathsf{K}^t(x)$ for a random input string x (see their paper for the exact statement). This provides another striking example of the power and reach of time-bounded Kolmogorov complexity.

While connections between time-bounded Kolmogorov complexity and different areas of theoretical computer science have been known for a long time (see, e.g., [Sip83, Ko91, ABK+06, AF09]), recent applications of it to cryptography [LP20, RS21, LP21], learning [CIKK16, HN21], average-case complexity [Hir21], circuit complexity [OPS19], and proof search [Kra22] have led to much interest in this topic and to a number of related developments. We refer the reader to these papers and to [All92, All01, For04, Lee06, All17, LV19, All21] for more information on different time-bounded Kolmogorov complexity measures and their applications.

[3] As discovered by Levin, this is achieved by an algorithm that attempts to compute an n-bit prime by carefully simulating all programs of small description length for an appropriate number of steps until an n-bit prime is found.

Probabilistic (Time-Bounded) Kolmogorov Complexity. The need to use time-bounded Kolmogorov complexity in certain applications can create issues that are not present in the case of (time-unbounded) Kolmogorov complexity. More precisely, several central results from Kolmogorov complexity are not known to hold in a time-bounded setting. Some of them do survive under a plausible assumption (e.g., a *source coding theorem* holds for K^t under a strong derandomisation assumption [AF09]), but this leads to *conditional* results only. In other cases, the validity of a result in the setting of time-bounded Kolmogorov complexity is closely tied to a longstanding open problem in complexity theory (e.g., the computational difficulty of estimating $\mathsf{K}^t(x)$ and the aforementioned connection to OWFs [LP20]). We refer to [Lee06] for an extensive discussion on the similarities and differences between Kolmogorov complexity and its time-bounded counterparts.

Going beyond the technical difficulties of employing time-bounded Kolmogorov complexity, which some papers such as [Hir21] managed to overcome with the right assumptions in place, there is perhaps a more relevant issue in the application of notions such as Kt and K^t to algorithms and complexity: these classical measures refer to *deterministic* algorithms and programs. However, in many cases it is desirable or even necessary to consider *randomised* algorithms. Since the random strings that are part of the input of a randomised algorithm have high complexity, the classical theory of time-bounded Kolmogorov complexity might be inappropriate or simply cannot be applied in such contexts.

To mitigate these issues and develop a more robust theory of time-bounded Kolmogorov complexity that can be deployed in the important setting of randomised computations, some recent papers have explored *probabilistic* notions of time-bounded Kolmogorov complexity [Oli19, LO21, LOS21, GKLO22, LOZ22]. For this to make sense, we must conciliate the high complexity of a random string, which can be accessed by a randomised algorithm, with the goal of obtaining a succinct representation of $x \in \{0,1\}^*$. Note that simply storing a good choice of the random string r for a small program M that prints x when given r does not lead to a succinct representation of x.

The key concept employed in the aforementioned papers is that of a *probabilistic representation* of the string x. In other words, this is the code of a randomised program M such that, for most choices of its internal random string r, M prints x from r. Observe that the representation itself is a deterministic object: the code of M. However, to recover x from M, we must run the randomised algorithm M, meaning that we obtain x with high probability but there might be a small chance that M outputs a different string.[4] If $|M|$ is small, we obtain a succinct probabilistic representation of x. It is possible to introduce different variants of probabilistic time-bounded Kolmogorov complexity, and we properly define them in Section 3.

[4]This is similar to the notion of a pseudodeterministic algorithm from [GG11].

The investigation of probabilistic Kolmogorov complexity and of probabilistic representations is motivated from several angles:

(*i*) If we are running a randomised algorithm over an input string x, then storing a probabilistic representation of x instead of x can be done without loss of generality. There is already a small probability that the randomised algorithm outputs an incorrect answer, so it makes sense to tolerate a small probability of computing over a wrong input as well (i.e., when x is not correctly recovered from its probabilistic representation).

(*ii*) We will see later in the survey that probabilistic Kolmogorov complexity allows us in some cases to obtain *unconditional* versions of results that previously were only known to hold under strong complexity-theoretic assumptions.

(*iii*) As alluded to above, there are situations where the deterministic time-bounded measures simply cannot be applied due to the presence of randomised computations involving random strings of high complexity.

(*iv*) Finally, advances in probabilistic Kolmogorov complexity can be translated into results and insights for the classical notions of Kt complexity and K^t complexity, under certain derandomisation hypotheses.

Before describing our results and explaining the points mentioned above in more detail, we present a list of five fundamental questions to guide our investigation and exposition of probabilistic Kolmogorov complexity.

Q1. Usefulness: *Are there shorter probabilistic representations for natural objects, such as prime numbers? Can such representations detect structure in data that is inaccessible for* Kt *and* K^t*?*

Q2. Probabilistic Compression: *If succinct probabilistic representations exist, how can we efficiently compute one such representation? This is particularly relevant for data compression.*

Q3. Applications: *Are there interesting applications of probabilistic time-bounded Kolmogorov complexity to algorithms and complexity theory?*

Q4. Computational Hardness: *If provably secure cryptography exists, it must be impossible to efficiently detect certain patterns in data. Is it computationally hard to decide if a string admits a succinct probabilistic representation?*

Q5. Finding an Incompressible String: *Can we explicitly produce a string that does not admit a short probabilistic representation? What are such strings useful for?*

In the remaining parts of this article, we explain the recent progress on Questions Q1-Q5 achieved by references [Oli19, LO21, LOS21, GKLO22, LOZ22]. Along the way, we highlight some concrete open problems and present directions for further research. Due to space constraints, we often provide only a sketch of the underlying arguments, referring to the original references for more details.

Organisation and Overview. For convenience of the reader, we provide below a brief overview of each remaining section of this survey and how it relates to Questions Q1-Q5 described above.

– Section 2 fixes notation and formalises the deterministic time-bounded Kolmogorov complexity notions Kt and K^t.

– Section 3 formalises the intuitive concept of probabilistic representations discussed above. We introduce the probabilistic measures rKt, rK^t, and pK^t and describe some simple applications.

– Section 4 addresses Question Q1 (Usefulness) and explains a result from [LOS21] showing that infinitely many primes admit efficient probabilistic representations of sub-polynomial complexity. This is a significant improvement over the aforementioned $\approx n/2$ bound for Kt complexity.

– Section 5 covers the relation between sampling algorithms for a distribution over strings and the existence of probabilistic representations for individual strings [LO21, LOZ22]. Such results are called source coding theorems and have applications to Question Q2 (Probabilistic Compression).

– Section 6 approaches Question Q3 (Applications) and discusses applications of rKt, rK^t, and pK^t to average-case complexity and learning [GKLO22, LOZ22]. We employ these notions to simplify previous proofs, obtain new results that crucially rely on probabilistic Kolmogorov complexity, and establish unconditional analogues of theorems that were only known under derandomisation hypotheses.

– Section 7 is connected to Question Q1 (Usefulness) and focuses on the relation between time-bounded deterministic and probabilistic measures. We observe that these notions essentially coincide under strong enough derandomisation assumptions [Oli19, GKLO22]. Assuming them, insights from probabilistic Kolmogorov complexity readily translate into information about Kt and K^t.

– Section 8 sheds light on Question Q4 (Computational Hardness) by unconditionally establishing that certain computational problems about estimating the probabilistic time-bounded Kolmogorov complexity of an input string cannot be solved in probabilistic polynomial time [Oli19, LOS21].

– Section 9 shows that Question Q5 (Finding an Incompressible String) is closely related to the existence of hierarchy theorems for probabilistic time [LO21, LOS21], a fundamental question in computational complexity theory.

– Section 10 provides some concluding remarks and prospects for the potential impact of (probabilistic) time-bounded Kolmogorov complexity in algorithms and complexity.

Acknowledgements. We thank Michal Koucký for the invitation to write this survey. We are grateful to Eric Allender, Bruno P. Cavalar, Lijie Chen, Valentine Kabanets, Michal Koucký, Ninad Rajgopal, and Marius Zimand for sharing comments and suggestions on a preliminary version of the text. This work received support from the Royal Society University Research Fellowship URF\R1\191059 and from the EPSRC New Horizons Grant EP/V048201/1.

2 Preliminaries

For a positive integer m, we let $[m] \stackrel{\text{def}}{=} \{1, 2, \ldots, m\}$. Given a non-negative real number α, we let $\lceil \alpha \rceil \in \mathbb{N}$ denote the smallest integer a such that $\alpha \leq a$. For a string $w \in \{0,1\}^*$, we use $|w| \in \mathbb{N}$ to denote its length. We let ϵ represent the empty string.

Let U be a Turing machine. For a function $t\colon \mathbb{N} \to \mathbb{N}$ and a string $x \in \{0,1\}^*$, we let

$$\mathsf{K}^t_U(x) \stackrel{\text{def}}{=} \min_{p \in \{0,1\}^*} \left\{ |p| \mid U(p, \epsilon) \text{ outputs } x \text{ in at most } t(|x|) \text{ steps} \right\}$$

be the *t-time-bounded Kolmogorov complexity* of x. The machine U is said to be *time-optimal* if for every machine M there exists a constant c_M such that for all $x \in \{0,1\}^n$ and $t\colon \mathbb{N} \to \mathbb{N}$ satisfying $t(n) \geq n$,

$$\mathsf{K}^{c_M \cdot t \log t}_U(x) \leq \mathsf{K}^t_M(x) + c_M,$$

where for simplicity we write $t = t(n)$. It is well known that there exist time-optimal machines (see, e.g., [LV19, Chapter 7]). We fix such a machine, and drop the index U when referring to time-bounded Kolmogorov complexity measures.

Given strings $x, y \in \{0,1\}^*$, we can also consider the *conditional t-time-bounded Kolmogorov complexity of x given y*, defined as

$$\mathsf{K}^t(x \mid y) \stackrel{\text{def}}{=} \min_{p \in \{0,1\}^*} \Big\{ |p| \mid U(p,y) \text{ outputs } x \text{ in at most } t(|x|) \text{ steps} \Big\}.$$

In the definitions above, the function $t\colon \mathbb{N} \to \mathbb{N}$ is fixed in advance. In many situations, it is also useful to consider a notion of time-bounded Kolmogorov complexity where the time bound of the machine is not fixed but instead affects the resulting complexity measure. One of the most prominent such measures is Levin's Kt complexity, defined as

$$\mathsf{Kt}(x) \stackrel{\text{def}}{=} \min_{p \in \{0,1\}^*,\, t \in \mathbb{N}} \Big\{ |p| + \lceil \log t \rceil \mid U(p,\epsilon) \text{ outputs } x \text{ in at most } t \text{ steps} \Big\}.$$

This definition can be extended to conditional Kt complexity $\mathsf{Kt}(x \mid y)$ in the natural way.

From now on, we will not distinguish between a Turing machine M and its encoding p_M according to U. While the running time t of M on an input y and the running time of the universal machine U on (p_M, y) might differ by a multiplicative factor of $O(\log t)$, this will be inessential in all results and applications discussed in this survey.[5]

We use $\mathsf{K}(x)$ to refer to the (time-unbounded) Kolmogorov complexity of the string x.

3 Probabilistic Notions of Kolmogorov Complexity: rKt, rK^t, and pK^t

In Section 2, we introduced two *deterministic* notions of time-bounded Kolmogorov complexity: K^t and Kt. In order to extend these definitions to the setting of *randomised* computations, we consider an algorithm with a short description that outputs a fixed string $x \in \{0,1\}^n$ with high probability. Intuitively, the code of this algorithm serves as a *probabilistic representation* of x.

A bit more formally, we consider a randomised Turing machine (RTM) M such that

$$\Pr_M[M(\epsilon) \text{ outputs } x] \geq 2/3.$$

Since we are interested in time-bounded representations, in our definitions we must decide if we require (1) $M(\epsilon)$ to run in time $\leq t$ over all computation paths;

[5] It is also possible to consider prefix-free notions of Kolmogorov complexity. Since our results hold up to additive $O(\log |x|)$ terms, we will not make an explicit distinction.

or (2) with probability $\geq 2/3$, $M(\epsilon)$ runs in time $\leq t$ and outputs x. It turns out that this distinction is not really crucial for the results discussed in this survey, since they are robust to additive overheads of order $\log n$. In more detail, by specifying and storing a positive integer $i \in [n]$, which can be represented using just $\log n$ bits, we can always enforce the machine M to stop in time 2^i.

Remark 1. In the definitions presented below, we abuse notation and refer to a machine M and its code. Formally, as in the definitions from the preceding section, M should be an arbitrary string (and not be restricted to a string that is a well-formed description of a machine) that is provided as input to the machine U.[6] This is important to guarantee that the Kolmogorov complexity of an arbitrary string of length n is at most $n + O(1)$. Defining Kolmogorov complexity and its time-bounded variants using the *code* of a machine might only allow us to prove an upper bound of $O(n)$, which can create issues in some applications where a tight worst-case bound is needed. To simplify the presentation, we blur this distinction in the remaining parts of this survey.

rK^t Complexity [BLvM05, LOS21].[7] This is the randomised analogue of K^t, where the time function $t\colon \mathbb{N} \to \mathbb{N}$ is fixed in advance. For a string $x \in \{0,1\}^*$, we let

$$\mathsf{rK}^t(x) \stackrel{\text{def}}{=} \min_{\text{RTM } M} \{|M| \mid M(\epsilon) \text{ outputs } x \text{ in } t(|x|) \text{ steps with probability} \geq 2/3\}$$

denote its *randomised t-time-bounded Kolmogorov complexity*. As an example of the use of rK^t, suppose a computationally unbounded party A holds a string x, and that A would like to communicate x to a t-time-bounded party B that has access to random bits. Then A can send $k = \mathsf{rK}^t(x)$ bits to B by communicating the description of a randomised Turing machine M as above. B is able to recover x from M with high probability simply by running $M(\epsilon)$.

pK^t Complexity [GKLO22]. Fix a function $t\colon \mathbb{N} \to \mathbb{N}$, as before. For a string $x \in \{0,1\}^*$, the *probabilistic t-time-bounded Kolmogorov complexity* of x is defined as

$$\mathsf{pK}^t(x) \stackrel{\text{def}}{=} \min\left\{k \in \mathbb{N} \;\middle|\; \mathop{\mathbf{Pr}}_{w \sim \{0,1\}^{t(|x|)}}\left[\exists \text{TM } M \in \{0,1\}^k,\, M(w) \text{ outputs } x \text{ within } t(|x|) \text{ steps}\right] \geq \frac{2}{3}\right\}.$$

Note that M is a deterministic machine in the above definition. In other words, if $k = \mathsf{pK}^t(x)$, then with probability at least $2/3$ over the choice of the random string w, given w the string x admits a t-time-bounded encoding of length k, i.e., $\mathsf{K}^t(x \mid w) \leq k$. In particular, if two parties share a typical *public* random string w,

[6] We assume that U has access to a tape with random bits.

[7] [BLvM05] refers to this notion as CBP^t complexity.

then x can be transmitted with k bits and decompressed in time $t = t(|x|)$. For a reader familiar with standard complexity classes, the condition $\mathsf{K}^t(x) \leq s$ is reminiscent of NP, while $\mathsf{rK}^t(x) \leq s$ and $\mathsf{pK}^t(x) \leq s$ essentially correspond to MA and AM, respectively.

The definition of pK^t complexity is more subtle than the definitions of K^t and rK^t. In particular, small pK^t complexity provides a short efficient description only in the presence of a fixed, "good" random string. Interestingly, pK^t turns out to be surprisingly useful in applications of time-bounded Kolmogorov complexity, as discussed in Sections 5 and 6.

The following inequalities immediately follow from these definitions.

Fact 2. *For every string $x \in \{0,1\}^*$ and function $t\colon \mathbb{N} \to \mathbb{N}$, we have $\mathsf{pK}^t(x) \leq \mathsf{rK}^t(x) \leq \mathsf{K}^t(x)$.*

rKt Complexity [Oli19]. We can also consider the *randomised* Kt *complexity* of a string $x \in \{0,1\}^*$, defined as

$$\mathsf{rKt}(x) \stackrel{\text{def}}{=} \min_{\text{RTM } M,\, t\in\mathbb{N}} \{|M| + \lceil \log t \rceil \mid M(\epsilon) \text{ outputs } x \text{ in } t \text{ steps with probability} \geq 2/3\}\,.$$

All these probabilistic notions of time-bounded Kolmogorov complexity can be generalised to capture the conditional complexity of x given y in the natural way. As a concrete example, suppose a Boolean formula $F(x_1, \ldots, x_n)$ admits a satisfying assignment $\alpha \in \{0,1\}^n$ such that $\mathsf{rKt}(\alpha \mid F) \leq k$. Then we can find in time $\widetilde{O}(2^k \cdot |F|)$ and with probability $\geq 2/3$ a satisfying assignment of F by performing the following randomised computation: for each $i \in [k]$, enumerate all RTM M of description length i, run $M(F)$ for at most 2^{k-i} steps, and output the first string $\beta \in \{0,1\}^n$ generated in one of the simulations such that $F(\beta) = 1$.

An important property of Kolmogorov complexity is that, by a simple counting argument, most strings of length n are *incompressible*, i.e., they do not admit representations of length noticeably shorter than n. Similarly, most strings do not admit succinct probabilistic representations, even in the presence of a fixed advice string y.

Proposition 3 (Incompressibility)**.** *Let $n \geq 1$ and consider an arbitrary time bound $t(n)$. For each string $y \in \{0,1\}^*$, measure $C \in \{\mathsf{rK}^t, \mathsf{pK}^t, \mathsf{rKt}\}$, and integer $k \geq 1$, the following holds.*

$$\Pr_{x\sim\{0,1\}^n}\left[C(x \mid y) < n - k\right] \;=\; O(2^{-k}).$$

Proof Sketch. For $C \in \{\mathsf{rK}^t, \mathsf{rKt}\}$, the result follows from a simple counting argument, using that a valid probabilistic representation represents a single string (i.e.,

the success probability of printing the string is $\geq 2/3$, so it is uniquely specified given the machine).

On the other hand, when $C = \mathsf{pK}^t$, we argue as follows. If a large fraction of n-bit strings x have bounded pK^t complexity, by an averaging argument, there is a fixed choice of the random string $w \in \{0,1\}^{t(n)}$ such that, given w, a large fraction of the n-bit strings admit bounded descriptions for this choice of w as the random string. We can then use a similar counting argument to show that this is contradictory. See [GKLO22] for the details. □

It is also possible to define pKt complexity, in analogy with the aforementioned definitions. However, since we are not aware of an interesting application of pKt, we will not discuss it here.

Other notions of time-bounded Kolmogorov complexity involving randomised computations have been considered in the literature. For instance, [BLvM05] considers CAM^t, a variant that combines randomness and nondeterminism. Due to space constraints, this survey will only cover rKt, rK^t, pK^t and their recent applications.

4 Prime Numbers with Short Descriptions and Pseudodeterministic PRGs

As briefly discussed in Section 1, an important question about prime numbers is whether they admit succinct representations, which is tightly connected to the fundamental problem of generating large primes deterministically. While this remains a notoriously difficult question to answer, we can still ask whether prime numbers admit succinct *probabilistic* representations. Results for this question were recently obtained in [OS17b, LOS21], by considering different notions of (time-bounded) randomised Kolmogorov complexity.

Before describing these results, we first note that it is impossible to compress *every* prime, given the Prime Number Theorem, which asserts that the number of primes whose values are less than or equal to N is roughly $N/\log N$. In particular, by a simple counting argument, this means that we cannot compress every n-bit prime to $o(n)$ bits. Therefore, here we ask whether there is an infinite sequence $\{p_m\}_{m\in\mathbb{N}}$ of increasing primes p_m that admit non-trivial probabilistic representations. The first non-trivial result of this form was established for rKt complexity.

Theorem 4 (rKt Upper Bounds for Primes [OS17b])**.** *For every $\varepsilon > 0$, there is an infinite sequence $\{p_m\}_{m\geq 1}$ of increasing primes p_m such that $\mathsf{rKt}(p_m) \leq |p_m|^{\varepsilon}$, where $|p_m|$ denotes the bit-length of p_m.*

Theorem 4 was proved via the construction of a *pseudodeterministic pseudorandom generator*. Informally, a pseudorandom generator (PRG) is an efficient procedure mapping a short string (called *seed*) to a long string, with the property that its output "looks random" to algorithms with bounded running time.[8] A PRG G is called *pseudodeterministic* if there is a probabilistic algorithm that, given a seed z, computes $G(z)$ with high probability. The following pseudodeterministic PRG was obtained in [OS17b].

Theorem 5 (A Pseudodeterministic Sub-Exponential Time PRG [OS17b])**.** *For every $\varepsilon > 0$ and $c, d \geq 1$, there exists a generator $G = \{G_n\}_{n\geq 1}$ with $G_n \colon \{0,1\}^{n^\varepsilon} \to \{0,1\}^n$ for which the following holds:*

Running Time: *There is a probabilistic algorithm that given n, $x \in \{0,1\}^{n^\varepsilon}$, runs in time $O\left(2^{n^\varepsilon}\right)$ and outputs $G_n(x)$ with probability $\geq 2/3$.*

Pseudorandomness: *For every algorithm A that runs in time at most n^c, there exist infinitely many input lengths n such that*

$$\left| \Pr_{x\sim\{0,1\}^n}[A(x) = 1] - \Pr_{z\sim\{0,1\}^{n^\varepsilon}}[A(G_n(z)) = 1] \right| \leq \frac{1}{n^d}.$$

Assuming Theorem 5, we show how to obtain Theorem 4.

Proof of Theorem 4. Let A be a deterministic polynomial-time algorithm for primality testing (e.g., [AKS02]), which takes as input an n-bit integer x and outputs 1 if and only if x is a prime. Suppose A runs in time n^c for some constant $c > 0$. Note that by the Prime Number Theorem, a uniformly random n-bit integer is a prime number with probability at least $1/O(n)$.

Let $\varepsilon > 0$ be any constant, and consider an infinitely often pseudodeterministic PRG $\{G_n\}_n$ from Theorem 5 with $G_n \colon \{0,1\}^{n^{\varepsilon/2}} \to \{0,1\}^n$ that is secure against (n^c)-time algorithms and has associated error parameter $\gamma = 1/n^2$. By the second item of Theorem 5, for infinitely many values of n, we have

$$\left| \mathbf{Pr}_{x\sim\{0,1\}^n}[A(x) = 1] - \mathbf{Pr}_{z\sim\{0,1\}^{n^{\varepsilon/2}}}[A(G_n(z)) = 1] \right| \leq \frac{1}{n^2},$$

which implies

$$\mathbf{Pr}_{z\sim\{0,1\}^{n^{\varepsilon/2}}}[A(G_n(z)) = 1] \geq \frac{1}{O(n)} - \frac{1}{n^2} \geq \frac{1}{O(n)}.$$

[8] Unconditionally constructing such PRGs is tightly connected to the derandomisation of probabilistic algorithms. While this remains a longstanding open problem, there has been progress in designing *pseudodeterministic* PRGs.

In particular, this means that there exists some $z \in \{0,1\}^{n^{\varepsilon/2}}$ such that $p := G(z)$ is an n-bit prime. By hardcoding n and this seed z, and using that $G(z)$ is a uniform procedure that can be computed probabilistically in time $t(n) = O\left(2^{n^{\varepsilon/2}}\right)$, we get that for infinitely many values of n, there is an n-bit prime p such that

$$\mathsf{rKt}(p) \leq \left(n^{\varepsilon/2} + O(\log n) + O(1)\right) + \log\left(O\left(2^{n^{\varepsilon/2}}\right)\right) \leq n^{\varepsilon},$$

as desired. □

For those primes shown to have small rKt complexity, given the corresponding encoding, one can probabilistically recover the prime in sub-exponential time. We can then further ask whether we can obtain succinct representations that can be decoded more efficiently, say, in polynomial time. Note that this is precisely to show that there are infinitely many primes whose $\mathsf{rK}^{\mathrm{poly}}$ complexity is small. This question was answered in the affirmative by a subsequent work of Lu, Oliveira and Santhanam.

Theorem 6 ($\mathsf{rK}^{\mathrm{poly}}$ Upper Bounds for Primes [LOS21])**.** *For every $\varepsilon > 0$, there is an infinite sequence $\{p_m\}_{m\geq 1}$ of increasing primes p_m such that $\mathsf{rK}^t(p_m) \leq |p_m|^{\varepsilon}$, where $t(n) = n^k$ for some constant $k = k(\varepsilon) \geq 1$, and $|p_m|$ denotes the bit-length of p_m.*

Similar to Theorem 4, Theorem 6 was proved via the construction of a certain pseudodeterministic PRG. Note that the reason why we got sub-exponential decoding time in Theorem 4 is due to the fact that the PRG from Theorem 5 requires sub-exponential time to compute. Then to obtain a polynomial decoding time as in Theorem 6, it suffices to construct a (pseudodeterministic) PRG that can be computed in polynomial time. Such a PRG was obtained in [LOS21] using a more sophisticated approach that builds on [OS17b].

Theorem 7 (A Pseudodeterministic Polynomial-Time PRG with 1 Bit of Advice [LOS21])**.**
For every $\varepsilon > 0$ and $c, d \geq 1$, there exists a generator $G = \{G_n\}_{n\geq 1}$ with $G_n \colon \{0,1\}^{n^{\varepsilon}} \to \{0,1\}^n$ for which the following holds:

Running Time: *There is a probabilistic polynomial-time algorithm that given n, $x \in \{0,1\}^{n^{\varepsilon}}$, and an advice bit $\alpha(n) \in \{0,1\}$ that is independent of x, outputs $G_n(x)$ with probability $\geq 2/3$.*

Pseudorandomness: *For every algorithm A that runs in time at most n^c, there exist infinitely many input lengths n such that*

$$\left| \Pr_{x\sim\{0,1\}^n}[A(x) = 1] - \Pr_{z\sim\{0,1\}^{n^{\varepsilon}}}[A(G_n(z)) = 1] \right| \leq \frac{1}{n^d}.$$

Using Theorem 7, it is easy to show Theorem 6 by mimicking the above proof of Theorem 4, with one caveat that computing the PRG in Theorem 7 requires one bit of advice. However, this extra bit can be hardcoded into the encoding without affecting its length by much.

We remark that the results presented above work in much more generality, and can be used to show that any dense language decidable in polynomial time admits infinitely many positive inputs of sub-polynomial $\mathsf{rKt}^{\mathrm{poly}}$ complexity. The set of primes is just one interesting example of such a language. We refer to [OS17b, LOS21] for additional applications of pseudodeterministic PRGs and for the proofs of Theorems 5 and 7.

We end this section with a couple of open problems. Note that both Theorem 4 and Theorem 6 show only that there are *infinitely many* values of n such that some n-bit prime has rKt or $\mathsf{rK}^{\mathrm{poly}}$ complexity at most n^{ε}.

Problem 8. *Show that for each $\varepsilon > 0$, there exists n_0 such that for every $n \geq n_0$, there is an n-bit prime p_n such that $\mathsf{rKt}(p_n) \leq n^{\varepsilon}$.*

Also, can we improve the sub-polynomial upper bounds to, say, poly-logarithmic?

Problem 9. *Prove that there is a constant $C \geq 1$ and an infinite sequence $\{p_m\}_{m\geq 1}$ of increasing primes p_m such that $\mathsf{rKt}(p_m) = (\log |p_m|)^C$.*

5 Sampling Algorithms, Coding Theorems, and Search-to-Decision Reductions

The coding theorem for Kolmogorov complexity roughly states that if a string x can be sampled with probability δ by some algorithm A, then its Kolmogorov complexity $\mathsf{K}(x)$ is at most $\log(1/\delta) + O_A(1)$. In particular, strings that can be generated with non-trivial probability by a program of small description length admit shorter representations. The coding theorem is a fundamental result in Kolmogorov complexity theory that has found many applications in theoretical computer science (see, e.g., [LV92, Lee06, Aar14, IRS21]). In fact, [Lee06] regards the coding theorem as one of the four pillars of Kolmogorov complexity.[9]

The proof of the coding theorem crucially explores the time-unbounded feature of the Kolmogorov complexity measure, and it is unclear how it can be extended to the time-bounded setting. Ideally, we would like to show that if a string x can be generated with probability δ by some *efficiently samplable* distribution, then its time-bounded Kolmogorov complexity $\mathsf{Kt}(x)$ is about $\log(1/\delta)$. One reason why such a time-bounded coding theorem is hopeful is that it can

[9]The other three are incompressibility, language compression, and symmetry of information.

be proven under certain strong derandomisation assumption [AF09].[10] In particular, under such an assumption, if a polynomial-time samplable distribution outputs a string x with probability at least δ, then $\mathsf{Kt}(x) \leq \log(1/\delta) + O(\log n)$. However, the latter result is only *conditional*, in the sense that it relies on an unproven assumption that seems far beyond the reach of currently known techniques. Moreover, strong assumptions of this form could even be false. While it remains unclear whether we can obtain a coding theorem for Kt, [LO21] considered the problem of establishing an *unconditional* coding theorem in the *randomised time-bounded setting*. Somewhat surprisingly, it can be shown *unconditionally* that if a string x can be sampled efficiently with probability δ, then $\mathsf{rKt}(x) \leq O(\log 1/\delta) + O(\log n)$. In a subsequent work [LOZ22], this result is further improved to $\mathsf{rKt}(x) \leq (2 + o(1)) \cdot \log 1/\delta + O(\log n)$.

Theorem 10 (Coding Theorem for rKt [LOZ22])**.** *Suppose there is an efficient algorithm A for sampling strings such that $A(1^n)$ outputs a string $x \in \{0,1\}^n$ with probability at least δ. Then*

$$\mathsf{rKt}(x) \leq 2\log(1/\delta) + O\Big(\log n + \log^2 \log(1/\delta)\Big),$$

where the constant behind the $O(\cdot)$ depends on A and is independent of the remaining parameters. Moreover, given x, the code of A, and δ, it is possible to compute in time $\mathsf{poly}(n, |A|)$, with probability ≥ 0.99, a probabilistic representation of x that satisfies this rKt-complexity bound. (*The running time of this algorithm does not depend on the time complexity of A.*)

Similar to the results in the previous section that are concerned with the compressibility of prime numbers, the results of [LO21, LOZ22] again show the power of utilizing randomness in Kolmogorov complexity, which enables us to establish results for time-bounded Kolmogorov complexity that seem very difficult to show in the deterministic setting. We refer to these papers for a discussion of the techniques employed to show an unconditional coding theorem for rKt.

We note that (as in previous work of [LO21]) the coding theorem in Theorem 10 has an unexpected *constructive* feature: it gives a polynomial-time probabilistic algorithm that, when given x, the code of the sampler, and δ, outputs a probabilistic representation of x that certifies the claimed rKt complexity bound. (Additionally, the running time of this algorithm does not depend on the running time of the sampler.) Such an *efficient* coding theorem has interesting implications for search-to-decision reductions for rKt. Recall that a search-to-decision reduction is an efficient procedure that allows one to find solutions to a problem from the

[10] The assumption in [AF09] states that there is a language $L \in \mathsf{TIME}\big[2^{O(n)}\big]$ that requires Boolean circuits of size $2^{\Omega(n)}$ for all but finitely many n, even in the presence of oracle gates to a Σ_2^p-complete problem in the circuit.

mere ability to decide when a solution exists. Using results from [LO21, LOZ22], one can show the following search-to-decision reduction for rKt.

Theorem 11 (Instance-Wise Search-to-Decision Reduction for rKt [LO21])**.** *Let O be a function that linearly approximates* rKt *complexity. That is, for every $x \in \{0,1\}^*$,*

$$\Omega(\mathsf{rKt}(x)) \;\leq\; O(x) \;\leq\; O(\mathsf{rKt}(x)).$$

Then there is a randomised polynomial-time algorithm with access to O that, when given an input string x, outputs with probability ≥ 0.99 a valid rKt *representation of x of complexity $O(\mathsf{rKt}(x))$. Furthermore, this algorithm makes a single query q to O, where $q = x$.*

Proof Sketch. We would like to invoke Theorem 10 to efficiently compute an rKt representation of x, but the "moreover" part of this result requires the explicit code of a sampler. The idea is to construct a "universal" sampler that outputs x with the desired probability, then to hit this sampler with an appropriate coding theorem for rKt. For simplicity, suppose we knew the exact value $k = \mathsf{rKt}(x) \in \mathbb{N}$. Consider the following sampler A:

$A(1^n)$: Randomly selects a randomised program M of length k among all strings in $\{0,1\}^k$. Run M for at most 2^k steps, then output the n-bit string that M outputs during this simulation (or the string 0^n if M does not stop or its output is not an n-bit string).

Note that A runs in time $t = \mathsf{poly}(n, 2^k) = \mathsf{poly}(2^k)$ (since $k \geq \log n$ for any n-bit string), and that it outputs x with probability at least $\delta = 2^{-k} \cdot 2/3$, since by the definition of k at least one such program prints x with probability at least $2/3$. By the coding theorem for rKt from [LO21] (which is stated in a slightly more general form than Theorem 10), one obtains that $\mathsf{rKt}(x) = O(\log(1/\delta)) + O(\log t) = O(k)$. Since A is an explicit algorithm, crucially, its "moreover" part implies that we can efficiently output an rKt-representation of x of complexity $O(k)$. This completes the sketch of the proof.

We refer to [LO21, Section 4] for the formal proof of Theorem 11, which is a simple adaptation of the idea described here. □

An interesting feature of the above search-to-decision reduction is that it is *instance-wise* in the sense that to produce a near-optimal rKt representation of x, we only need to make a single query to a decision oracle for rKt on the same x.[11] Note that there are known search-to-decision reductions in the context of time-bounded Kolmogorov complexity with respect to various notions of complexity

[11] This is also called a *search-to-profile* reduction in some references in Kolmogorov complexity [RSZ21].

(e.g., [CIKK16, Hir18, Ila20, ILO20, LP20, Ila21]), but they require an oracle to the decision problem that is correct on all or at least on a large fraction of inputs. As a consequence of this feature, we can easily derive the following result.[12]

Corollary 12 ("Short Lists with Short Programs" [LO21])**.** *Given a string x of length n, it is possible to compute with probability ≥ 0.99 and in polynomial time a collection of at most $\ell = \log(n)$ strings $M_1, \ldots, M_\ell$ such that at least one of these strings is a valid* rKt *representation of x of complexity $O(\mathsf{rKt}(x))$.*

Proof. We run the instance-wise search-to-decision reduction on the input x. While it is not clear how to efficiently estimate $\mathsf{rKt}(x)$, we can still "guess" the rKt complexity of x to be of order 2^i, for each $i \in \{1, 2, \ldots, \log n\}$. We run the procedure on each possible guess, obtaining a list of strings $M_1, \ldots, M_\ell$, where $\ell = \log n$. Since there is at least one value i such that $2^i = \Theta(\mathsf{rKt}(x))$, we have the guarantee that in this case the reduction outputs with probability at least 0.99 a valid rKt representation of x of similar complexity. Therefore, the list contains with probability at least 0.99 a representation of the desired form. □

While the above coding theorem for rKt is a novel development after a long gap in an area with only conditional results, it has an important drawback: the rKt upper bound is at least $2\log(1/\delta))$ and hence is *sub-optimal*. In contrast, the bounds in the time-unbounded setting and in the conditional result of [AF09] mentioned above have the form $\log(1/\delta)$. A natural question then is whether we can show a coding theorem for rKt with an optimal dependence on the probability parameter δ, which is crucial in many applications of the result. It turns out that under a certain hypothesis about the security of cryptographic pseudorandom generators[13], the rKt bound in Theorem 10 is essentially optimal if we consider only coding theorems that are *efficient*, i.e., where an rKt representation can be constructed in polynomial time regardless of the running time of the sampler. In particular, [LOZ22] showed that in this case, there is no efficient coding theorem that can achieve a bound of the form $\mathsf{rKt}(x) \leq (2 - o(1)) \cdot \log(1/\delta) + \mathrm{poly}(\log n)$. On the other hand, the conditional coding theorem for Kt in [AF09] is not efficient. This leads to the following open problem on (unconditionally) showing an *existential* coding theorem for rKt with optimal parameters.

[12] Results of this form were previously known in time-unbounded Kolmogorov complexity (see [BMVZ18]).

[13] The hypothesis states that there is a pseudorandom generator $G\colon \{0,1\}^{\ell(n)} \to \{0,1\}^n$, where $(\log n)^{\omega(1)} \leq \ell(n) \leq n/2$, computable in time $\mathrm{poly}(n)$ that is secure against *uniform algorithms* running in time $2^{(1-\Omega(1))\cdot\ell(n)}$. Note that every candidate PRG of seed length $\ell(n)$ can be broken in time $2^{\ell(n)} \cdot \mathsf{poly}(n)$ by trying all possible seeds. This hypothesis can be viewed as a cryptographic analogue of the well-known strong exponential time hypothesis (SETH) about the complexity of k-CNF SAT [IP01].

Problem 13. *Show that if there is an efficient algorithm A for sampling strings such that $A(1^n)$ outputs a string $x \in \{0,1\}^n$ with probability at least δ, then* $\mathsf{rKt}(x) \leq \log(1/\delta) + \mathrm{poly}(\log n)$.

To this point, we have mentioned the existence of an optimal coding theorem for time-unbounded Kolmogorov complexity and an optimal conditional coding theorem for Kt (in fact, the conditional result holds even for K^t for $t = \mathsf{poly}(n)$). Also, an unconditional coding theorem can be obtained for rKt but its dependency on the probability parameter δ is not $\log(1/\delta)$ (Theorem 10). Note that rKt can be viewed as a "relaxed" notion of Kt and is intermediate between K and Kt. If we consider some further relaxed notion of time-bounded Kolmogorov complexity, can we show a coding theorem that is both unconditional and optimal?

Note that the time-bounded measure pK can be viewed as an intermediate notion between time-unbounded Kolmogorov complexity and time-bounded rK. It turns out that pK^t admits an optimal coding theorem.

Theorem 14 (Coding Theorem for pK^t [LOZ22])**.** *Suppose there is a randomised algorithm A for sampling strings such that $A(1^n)$ runs in time $T(n) \geq n$ and outputs a string $x \in \{0,1\}^n$ with probability at least $\delta > 0$. Then*

$$\mathsf{pK}^t(x) \;=\; \log(1/\delta) + O(\log T(n))\,,$$

where $t(n) = \mathrm{poly}(T(n))$ and the constant behind the $O(\cdot)$ depends on $|A|$ and is independent of the remaining parameters.

The proof of Theorem 14 is similar in spirit to that of the conditional coding theorem for $\mathsf{K}^{\mathrm{poly}}$ in [AF09]. As an application of the latter, [AF09] showed a *conditional* characterisation of the worst-case running times of languages that are in average polynomial time over all samplable distributions. Using Theorem 14, [LOZ22] provided an *unconditional* characterisation, and this will be discussed in Section 6.

Finally, we can connect the time-bounded coding theorems discussed in this section to the compressibility of prime numbers discussed in the previous section, via the following equivalence.

Theorem 15 (Equivalence Between Samplability and Compressibility [LO21])**.** *Let $\delta\colon \mathbb{N} \to [0,1]$ be a time-constructible function. The following statements are equivalent.*

(*i*) **Samplability.** *There is a randomised algorithm A for sampling strings such that, for infinitely many* (*resp. all but finitely many*) *n, $A(1^n)$ runs in time $(1/\delta(n))^{O(1)}$ and outputs an n-bit prime q_n with probability at least $\delta(n)^{O(1)}$.*

(*ii*) **Compressibility.** *For infinitely many* (*resp. all but finitely many*) *n, there is an n-bit prime* p_n *with* $\mathsf{rKt}(p_n) = O(\log(1/\delta(n)))$.

Proof Sketch. The implication from (*i*) to (*ii*) relies on the existing coding theorem for rKt. The other direction employs a universal sampler in the spirit of the proof of Theorem 11 sketched above. See [LO21] for the details. □

Theorem 15 can be seen as an analogue of the relation between deterministically constructing large primes and obtaining Kt upper bounds for primes, which was explained in Section 1. Using this result, the problem of showing that prime numbers have smaller rKt complexity (Problem 9) can be reduced to showing the existence of a faster sampling algorithm for primes. In particular, if we can sample an n-bit prime p_n in time $2^{\mathrm{poly}(\log n)}$ with probability at least $2^{-\mathrm{poly}(\log n)}$, then $\mathsf{rKt}(p_n) \leq \mathrm{poly}(\log n)$.

We remark that an even tighter equivalence between samplability and compressibility can be established using pK^t complexity, thanks to the optimality of Theorem 14.

6 Applications to Average-Case Complexity and Learning Theory

Understanding the relation between the average-case complexity of NP and its worst-case complexity is a central problem in complexity theory. More concretely, if every problem in NP is easy to solve on average, can we solve NP problems in polynomial time in the worst case? While addressing this question remains a longstanding open problem, significant results have been achieved in recent years using techniques from time-bounded Kolmogorov complexity [Hir20a, Hir21, CHV22] (see [Hir22a] for an overview). Related techniques have also led to the design of faster learning algorithms under the assumption that NP is easy on average [HN21]. Interestingly, the problems investigated in these references make no reference to Kolmogorov complexity. Still, the corresponding proofs rely on K^t complexity and its properties in important ways.

In this section, we describe recent applications of pK^t complexity to average-case complexity and learning theory [GKLO22, LOZ22]. While the definition of pK^t is more subtle compared with K^t and rK^t, its use comes with important benefits. As we explain later in this section, depending on the context, pK^t complexity allows us to extend previous results to the important setting of randomised computations, significantly simplify an existing proof, or obtain an unconditional result.

Average-Case Complexity. We first review some standard definitions from average-case complexity theory (see [BT06] for a survey of this area). Recall that $D =$

$\{D_n\}_{n\geq 1}$, where each D_n is a distribution supported over $\{0,1\}^*$, is called an ensemble of distributions. We say that $D \in \mathsf{PSamp}$ (or D is P-*samplable*) if there is a randomised polynomial-time algorithm A such that, for every $n \geq 1$, $A(1^n)$ is distributed according to D_n.

Let D be an ensemble of distributions. We say that a language L is solvable in *polynomial time on average* with respect to D if there is a deterministic algorithm A such that, for every n and for every x in the support of D_n, $A(x;n) = L(x)$, and there is a constant $\varepsilon > 0$ such that $\mathbf{E}_{x\sim D_n}[t_{A,n}(x)^\varepsilon/n] = O(1)$, where $t_{A,n}(x)$ denotes the running time of A on input $(x;n)$. We remark that this is equivalent to the existence of a deterministic algorithm B and of a polynomial p such that the following conditions hold:

- For every $n, \delta > 0$, and string x in the support of D_n, $B(x;n,\delta)$ outputs either $L(x)$ or the failure symbol $\perp$;
- For every $n, \delta > 0$, and every string x in the support of D_n, $B(x;n,\delta)$ runs in time at most $p(n, 1/\delta)$;
- For every n and every $\delta > 0$,

$$\Pr_{x\sim D_n}[B(x;n,\delta) = \perp] \leq \delta.$$

We refer to [BT06] for more information about this definition and its motivation.

A pair (L,D) is a *distributional problem* if $L \subseteq \{0,1\}^*$ and D is an ensemble of distributions. For a complexity class $\mathfrak{C}$ (e.g., $\mathfrak{C} = \mathsf{NP}$), we let $\mathsf{Dist}\mathfrak{C}$ denote the set of distributional problems (L,D) with $L \in \mathfrak{C}$ and $D \in \mathsf{PSamp}$. We say that $(L,D) \in \mathsf{AvgP}$ if L is solvable in polynomial time on average with respect to D.

Note that in the equivalent definition of AvgP the deterministic algorithm is never incorrect on an input x in the support of the distribution. Similarly, it is possible to consider average-case complexity with respect to *randomised errorless heuristic schemes*. Roughly speaking, such randomised algorithms are allowed to sometimes output the wrong answer, provided that on every input x in the support of the distribution, the fraction of random strings for which the algorithm outputs the wrong answer is small compared to the fraction of random strings for which it outputs either the right answer or the fail symbol $\perp$. Analogously to the definition of AvgP, if a distributional problem (L,D) admits a randomised errorless heuristic scheme, we say that $(L,D) \in \mathsf{AvgBPP}$. We refer again to [BT06] for the precise definition of this class and for an extensive discussion of this notion and its extensions.[14]

[14] It is also possible to consider randomised algorithms that can sometimes be incorrect on an input x with high probability over their internal randomness. This leads to the class $\mathsf{HeurBPP}$ of distributional problems. Relaxing some assumptions in this section to the setting of $\mathsf{HeurBPP}$ is an interesting research direction (see, e.g., [HS22]).

6.1 Worst-Case Time Bounds for Average-Case Easy Problems

Suppose that a language L is average-case easy. That is, L is solvable in deterministic polynomial time on average with respect to *all* P-samplable distributions. What can we say about the time needed to solve L in the *worst case*? In a beautiful work, Antunes and Fortnow [AF09] *characterised* the worst-case running time of such a language using the notion of *computational depth* [AFvM01]. Here the computational depth of a string x for a time bound t is defined as the difference $\mathsf{K}^t(x) - \mathsf{K}(x)$. It was shown in [AF09], under a strong derandomisation assumption, that a language L is average-case easy *if and only if* it can be solved in time $2^{O(\mathsf{K}^{\mathrm{poly}}(x)-\mathsf{K}(x)+\log(|x|))}$ for every input $x \in \{0,1\}^*$. The proof of this result crucially relied on the use of an *optimal* coding theorem for K^t. Since such a coding theorem is only known under a strong derandomisation assumption (see Section 5), the aforementioned characterisation is subject to the same unproven assumption.

As also mentioned in Section 5, it was observed in [LOZ22] that an optimal coding theorem can be unconditionally proved for pK^t (Theorem 14). It turns out that such a coding theorem enables us to show an *unconditional* version of Antunes and Fortnow's characterisation, where the worst-case running times for languages that are average-case easy can be characterised using a notion of *probabilistic computational depth*.

A key idea in the proof of this result is a notion of *universal* distribution via pK^t. More specifically, for a computable time bound function t, we define m^t to be the (semi-)distribution whose probability density function is $\mathsf{m}^t(x) \stackrel{\text{def}}{=} 2^{-\mathsf{pK}^t(x)-b\log|x|}$, where $b > 0$ is a large enough constant (that depends only on t).[15]

Theorem 16 (Unconditional "Worst-Case Time Bounds for Average-Case Easy Problems" [LOZ22])**.** *The following conditions are equivalent for any language* $L \subseteq \{0,1\}^*$.[16]

1. *For every* P*-samplable distribution D, L can be solved in polynomial time on average with respect to D.*

2. *For every polynomial p, L can be solved in polynomial time on average with respect to* m^p.

[15] The reason why we define $\mathsf{m}^t(x)$ this way instead of using just $2^{-\mathsf{pK}^t(x)}$ is to make sure that it forms a (semi-)distribution, i.e., that the sum of the probabilities is at most 1. More specifically, for every t, there is some constant $b > 0$ such that $\mathsf{K}(x) \leq \mathsf{pK}^t(x) + b\log|x|$ for every x (see [LOZ22, Lemma 32]), so $\sum_{x\in\{0,1\}^*} 2^{-\mathsf{pK}^t(x)-b\log|x|} \leq \sum_{x\in\{0,1\}^*} 2^{-\mathsf{K}(x)} \leq 1$, where the second inequality follows from Kraft's inequality. (Formally, to apply Kraft's inequality we need to consider prefix-free encodings. This is not an issue here, as a large enough constant b makes this possible.)

[16] In this statement and in its proof, we do not make a distinction between distributions and semi-distributions. (In a semi-distribution, the sum of the probabilities might add up to less than 1.)

3. *For every polynomial p, there exists a constant $c > 0$ such that the running time of some algorithm that computes L is bounded by $2^{O\left(\mathsf{pK}^p(x) - \mathsf{K}(x) + c\log(|x|)\right)}$ for every input $x \in \{0,1\}^*$.*

Proof Sketch. For simplicity, to sketch the proof of this theorem we will also consider the notion of average-case easiness with respect to *single distributions* instead of ensembles of distributions,[17] which does not incur a loss of generality (see [BT06, Section 6]).

We first sketch the equivalence between Item 1 and Item 2. We need to show that the class of distributions $\mathsf{m}^{\mathrm{poly}}$ is "universal" for the class of P-samplable distributions, in the sense that a language L is polynomial-time on average with respect to $\mathsf{m}^{\mathrm{poly}}$ if and only if the same holds with respect to all P-samplable distributions. Recall that if a distribution D *dominates* another distribution D' (i.e., $D(x) \gtrsim D'(x)$ for all x) and L is polynomial-time on average with respect to D, then the same holds with respect to D'. Therefore, to show the "universality" of $\mathsf{m}^{\mathrm{poly}}$, it suffices to establish the following claims.

1. Every P-samplable distribution is dominated by m^p, for some polynomial p.

2. For every polynomial p, m^p is dominated by some P-samplable distribution.

The first item above says that for every P-samplable D, $\mathsf{m}^p(x) \gtrsim D(x)$ for some polynomial p, which, by the definition of m^p, means $\mathsf{pK}^p(x) \lesssim \log(1/D(x))$. Note that this is essentially an optimal coding theorem for $\mathsf{pK}^{\mathrm{poly}}$ and hence follows from Theorem 14. To see the second item, consider any polynomial p. We define a P-samplable distribution roughly as follows. We first pick n with probability $\frac{1}{n\cdot(n+1)}$, and then randomly pick $k \in [2n]$, $w \in \{0,1\}^{p(n)}$, and a program $M \in \{0,1\}^k$. We then run $M(w)$ for at most $p(n)$ steps and output the string that M outputs. It is easy to see that for every $x \in \{0,1\}^n$, the above sampling process outputs x with probability at least $2^{-\mathsf{pK}^p(x)}/n^{O(1)}$ and hence dominates m^p.

It remains to show the equivalence between Item 2 and Item 3. Here we describe the implication from Item 2 to Item 3, which highlights the use of a fundamental result in Kolmogorov complexity called *Language Compression*. The other direction follows from a simple calculation (see [LOZ22]).

Consider the time bound t described by an arbitrary polynomial p. Let A be an algorithm that solves L in polynomial time on average with respect to m^t, and let $t_A(x)$ denote the running time of A on input x. For $n, i, j \in \mathbb{N}$ with $i, j \le n^2$, define

$$S_{i,j,n} \stackrel{\text{def}}{=} \left\{ x \in \{0,1\}^n \mid 2^i \le t_A(x) \le 2^{i+1} \text{ and } \mathsf{pK}^t(x) + b\log|x| = j \right\}.$$

[17] An algorithm A runs in polynomial time on average with respect to a (semi-)distribution D if there exists a constant ε such that, $\sum_{x\in\{0,1\}^*} \frac{t_A(x)^\varepsilon}{|x|} D(x) \le O(1)$, where $t_A(x)$ denotes the running time of A on input x.

Consider a nonempty set $S_{i,j,n}$, and let $r \in \mathbb{N}$ be such that $2^r \leq \left|S_{i,j,n}\right| < 2^{r+1}$. We claim that for every $x \in S_{i,j,n}$, its (time-unbounded) Kolmogorov complexity

$$\mathsf{K}(x) \leq r + O(\log n). \tag{4}$$

To see this, note that given i, j, n, we can first enumerate all the elements in $S_{i,j,n}$, which can be done since t is computable, and then using additional $r + 1$ bits, we can specify x in $S_{i,j,n}$. We remark that the core idea behind the above argument is the language compression theorem for (time-unbounded) Kolmogorov complexity, which states that for every (computable) language L, $\mathsf{K}(x) \leq \log |L \cap \{0,1\}^n| + O(\log n)$ for all $x \in L \cap \{0,1\}^n$.[18]

Now fix any n and $i, j \leq n^2$. Let r be such that $2^r \leq \left|S_{i,j,n}\right| < 2^{r+1}$. Then by assumption and by the definition of $S_{i,j,n}$, we have for some constants $\varepsilon, d > 0$,

$$d \geq \sum_{x \in S_{i,j,n}} \frac{t_A(x)^\varepsilon}{|x|} \cdot \mathsf{m}^t(x) \geq 2^r \cdot \frac{2^{\varepsilon \cdot i}}{n} \cdot 2^{-j} = 2^{\varepsilon \cdot i + r - j - \log n},$$

which yields $\varepsilon \cdot i + r - j - \log n \leq \log d$. By Equation (4) and using $j = \mathsf{pK}^t(x) + b \log n$, this implies that for every $x \in S_{i,j,n}$,

$$\varepsilon \cdot i \leq \mathsf{pK}^t(x) - \mathsf{K}(x) + O(\log n).$$

Therefore, we have that for every $x \in S_{i,j,n}$,

$$t_A(x) \leq 2^{i+1} \leq 2^{\varepsilon^{-1} \cdot \left(\mathsf{pK}^t(x) - \mathsf{K}(x) + O(\log n)\right)} = 2^{O\left(\mathsf{pK}^t(x) - \mathsf{K}(x) + c \log(|x|)\right)},$$

where $c > 0$ is a large enough constant independent of $n = |x|$. Since it is not hard to see that every $x \in \{0,1\}^n$ is in some set $S_{i,j,n}$, the result follows. □

6.2 Probabilistic Average-Case Easiness Implies Worst-Case Upper Bounds

The section covers recent developments from [GKLO22], which build on the breakthrough results of [Hir21] and on the subsequent papers [CHV22, GK22, Hir22b]. In short, the results from [Hir21] hold in the setting of *deterministic* computations, while [GKLO22] provides a framework that allows new relations between average-case complexity and worst-case complexity to be established in the more robust setting of *randomised* computations.

[18] In fact, it is possible to slightly modify the above argument, by appropriately defining a language with slices in correspondence to the sets $S_{i,j,n}$, so that language compression can be applied directly.

Next, we provide a high-level exposition of some results from [GKLO22] and their proofs. In particular, we explain the role of (conditional) versions of "language compression" and "symmetry of information" for pK^t, and how pK^t turns out to be a complexity measure that is particularly well-suited for these applications (see Remark 21 on "Why pK^t complexity?").

Our goal is to show a *worst-case* complexity upper bound for an arbitrary language $L \in \mathsf{NP}$ under an *average-case* easiness assumption, such as $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ or the weaker $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$. Note that Theorem 16 naturally suggests an approach: if L is easy on average (Item 1), then we can compute L on every input $x \in \{0,1\}^*$ (Item 3) in time

$$2^{O(\mathsf{pK}^p(x)-\mathsf{K}(x))} \cdot \mathsf{poly}(|x|),$$

where $p(\cdot)$ is a fixed but arbitrary polynomial. Therefore, if we could show that the quantity $\mathsf{pK}^p(x) - \mathsf{K}(x)$ is bounded for *every* x, we would be done. (Note that this is indeed the case for a uniformly *random* x, since $\mathsf{pK}^t(x)$ and $\mathsf{K}(x)$ are close to $n = |x|$ with high probability.)

This is not possible, but we can still hope to adapt the proof of Theorem 16 to obtain a more useful bound, under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$. A closer inspection of the argument reveals that the value $\mathsf{K}(x)$ in the bound $\mathsf{pK}^p(x) - \mathsf{K}(x)$ comes from the use of *language compression* for (time-unbounded) Kolmogorov complexity, which is applied to the sets $S_{i,j,n}$. If we had a language compression theorem for a time-bounded measure γ (e.g., $\gamma = \mathsf{K}^t$), we would be able to derive a worst-case running time exponent of the form $\mathsf{pK}^p(x) - \gamma(x)$. This makes progress towards our goal, since $\gamma(x) \geq \mathsf{K}(x)$. This initial idea turns out to be feasible, for $\gamma = \mathsf{pK}^q$ (think of $q(\cdot)$ as a polynomial larger than $p(\cdot)$).

Theorem 17 (Language Compression for pK^t under $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$; Informal[19])**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then for every language* $S \in \mathsf{AM}$*, there is a polynomial* q *such that for every* $x \in S \cap \{0,1\}^n$*,*

$$\mathsf{pK}^q(x) \leq \log |S \cap \{0,1\}^n| + \log q(n).$$

In order to implement the aforementioned plan, we need to make sure that the sets $S_{i,j,n}$ provide a language S that is *easy to compute*, since this is an assumption in Theorem 17. One can sidestep this issue by settling for a weaker result which assumes that the running time t_A of the average-case algorithm on a given input can be efficiently estimated without running the algorithm. This notion leads to a class of distributional problems called $\mathsf{Avg}_{\mathsf{BPP}}\mathsf{BPP}$ in [GKLO22], and to the stronger

[19] For technical reasons, the actual formulation of this result considers an ensemble of promise problems with padded inputs of the form $(x, 1^m)$, where $|x| = \ell(m)$. For simplicity, we omit this here. See [GKLO22] for the precise statement.

initial assumption that $\mathsf{DistNP} \subseteq \mathsf{Avg}_{\mathsf{BPP}}\mathsf{BPP}$. Another crucial idea, which we will not cover in more detail here, is to prove that $\mathsf{pK}^t(y)$ can be efficiently estimated for every string y under the assumption that NP is easy on average. We can then apply (an extension of) Theorem 17 to appropriately modified sets $S'_{i,j,n}$, which yields a worst-case running time of the form

$$2^{O(\mathsf{pK}^p(x)-\mathsf{pK}^q(x))} \cdot \mathsf{poly}(|x|).$$

One could hope for the quantity $\mathsf{pK}^p(x)-\mathsf{pK}^q(x)$, called the (p,q)-*probabilistic computational depth* of x, to be bounded for every string x. While this is not clear for the polynomials $p(n)$ and $q(n)$, a simple but neat argument involving a telescoping sum [Hir21, GKLO22] shows that, for any string x of length n, for some time bound $t(n) \leq 2^{O(n/\log n)}$ we have $\mathsf{pK}^t(x)-\mathsf{pK}^{\mathrm{poly}(t)}(x) = O(n/\log n)$. Intuitively, if we could adapt the previous strategy so that it yields more general worst-case upper bounds involving $(t, \mathrm{poly}(t))$-*probabilistic computational depth*, then a non-trivial exponent of $O(n/\log n)$ would be achieved by applying the argument to each choice of $t \leq 2^{O(n/\log n)}$.

A careful implementation of this plan leads to the following stronger consequence, where the worst-case upper bound holds for any language $L \in \mathsf{AM}$.

Theorem 18 ([GKLO22])**.** *If* $\mathsf{DistNP} \subseteq \mathsf{Avg}_{\mathsf{BPP}}\mathsf{BPP}$*, then* $\mathsf{AM} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$.

Can we obtain a similar worst-case upper bound under the weaker and more natural assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$? (In other words, without assuming that the running time of the average-case algorithm can be efficiently estimated?) This is currently open. However, it is possible to prove the following implications, which can be seen as a strengthening of some results from [Hir21] to the randomised setting. Recall that UP denotes the set of languages in NP whose positive instances admit unique witnesses.

Theorem 19 (Probabilistic Worst-Case to Average-Case Reductions [GKLO22])**.** *The following results hold.*

1. *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then* $\mathsf{UP} \subseteq \mathsf{RTIME}\left[2^{O(n/\log n)}\right]$.
2. *If* $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$*, then* $\mathsf{AM} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$.
3. *If* $\mathsf{DistPH} \subseteq \mathsf{AvgBPP}$*, then* $\mathsf{PH} \subseteq \mathsf{BPTIME}\left[2^{O(n/\log n)}\right]$.

The proof of Theorem 19 relies on *Symmetry of Information*, another pillar of Kolmogorov complexity (see [Lee06]). To describe a pair (x, y) of strings, one can combine the most succinct representation of x with the most succinct representation of y when x is given as advice. In Kolmogorov complexity, this

is captured by the inequality $\mathsf{K}(x,y) \leq \mathsf{K}(x) + \mathsf{K}(y \mid x) + O(\log(|x| + |y|))$. The symmetry of information principle is a theorem in Kolmogorov complexity stating that this is essentially the most economical way of describing the pair (x,y). In other words: $\mathsf{K}(x,y) \geq \mathsf{K}(x) + \mathsf{K}(y \mid x) - O(\log(|x| + |y|))$. One can then easily derive that $\mathsf{K}(x) - \mathsf{K}(x|y) = \mathsf{K}(y) - \mathsf{K}(y \mid x)$, up to a term of order $O(\log(|x| + |y|))$. Roughly speaking, the information that x contains about y is about the same the information that y contains about x.

The proof of symmetry of information for K requires an exhaustive search, which is not available in the time-bounded setting. Nevertheless, different forms of the principle can still be established in this more delicate setting under average-case easiness assumptions [GK22, Hir22b, GKLO22].

Theorem 20 (Symmetry of Information for pK^t under $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$ [GKLO22][20])**.** *If* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$*, then there exist polynomials p and p_0 such that for all sufficiently large $x, y \in \{0,1\}^*$ and every $t \geq p_0(|x|, |y|)$,*

$$\mathsf{pK}^t(x,y) \geq \mathsf{pK}^{p(t)}(x) + \mathsf{pK}^{p(t)}(y \mid x) - \log p(t).$$

Assuming Theorem 20, we provide a high-level exposition of the proof of a variant of Item 2 from Theorem 19: If $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$ then $\mathsf{NP} \subseteq \mathsf{RTIME}[2^{O(n/\log n)}]$. (A detailed informal presentation of Item 2 of Theorem 19 can be found in [GKLO22, Section 1.3].) Assume that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, and let $L \in \mathsf{NP}$. Fix some NP-verifier V for this language. For a string $x \in L$ of length n, let y_x be the lexicographic first string such that $V(x, y_x) = 1$.

1. On the one hand, it follows from Theorem 20 that there is a universal constant $a \geq 1$ such that, for every large enough t, $\mathsf{pK}^{t^a}(y_x \mid x) \leq \mathsf{pK}^t(x, y_x) - \mathsf{pK}^{t^a}(x) + O(\log t)$.

2. On the other hand, under the assumption that $\mathsf{Dist}\Sigma_2^{\mathsf{P}} \subseteq \mathsf{AvgBPP}$, it is possible to prove that, for some universal constant $\varepsilon > 0$ and for every large enough t, $\mathsf{pK}^t(x, y_x) \leq \mathsf{pK}^{t^\varepsilon}(x) + O(\log t)$. This is non-trivial: while it is possible to recover y_x from x with a powerful enough oracle, we must obtain a description of the pair (x, y_x) from (a fixed but arbitrary) x without the aid of such an oracle, using only an average-case easiness assumption.

3. Putting together the previous inequalities from Steps 1 and 2, we get that for every large enough t, $\mathsf{pK}^{t^a}(y_x|x) \leq \mathsf{pK}^{t^\varepsilon}(x) - \mathsf{pK}^{t^a}(x) + O(\log t)$. Consequently, we can upper bound $\mathsf{pK}^{t^a}(y_x|x)$ by the (t^ε, t^a)-probabilistic computational depth of x plus $O(\log t)$, for any $t \geq \mathrm{poly}(n)$, where $n = |x|$.

[20] A more general version of this result is used by [GKLO22] to establish Theorem 19 and its extensions.

4. As in the proof sketch of Theorem 18, one can show that for every x there is some $t(n) = 2^{O(n/\log n)}$ such that $\mathsf{pK}^{t^e}(x) - \mathsf{pK}^{t^a}(x) = O(n/\log n)$. Consequently, using that $\mathsf{pK}^{t_1}(\cdot) \leq \mathsf{pK}^{t_2}(\cdot)$ if $t_1 \geq t_2$, there is a constant $C \geq 1$ such that, for every string x of length n, we have $\mathsf{pK}^{\gamma}(y_x \mid x) \leq C \cdot n/\log n$, where $\gamma(n) = 2^{C \cdot n/\log n}$.

5. Finally, given a positive instance x of L and the upper bound on $\mathsf{pK}^{\gamma}(y_x \mid x)$ from Step 4, we can recover y_x with probability $\geq 2/3$ in time $2^{O(n/\log n)}$. Indeed, this follows from the definition of conditional pK^t complexity: by sampling a random string w of length $2^{C \cdot n/\log n}$ and simulating all machines M of length $\leq C \cdot n/\log n$ on input (x, w) for at most $2^{C \cdot n/\log n}$ steps, we generate y_x with probability at least $2/3$ over the choice of w. Since we can test each string produced in this way using the polynomial-time verifier $V(x, \cdot)$, it follows that $L \in \mathsf{RTIME}[2^{O(n/\log n)}]$.

Remark 21 (**Why pK^t complexity?**). Both language compression (Theorem 17) and symmetry of information (Theorem 20) are established using techniques from computational pseudorandomness related to the design and analysis of pseudorandom generators (PRGs). This approach has proven extremely useful in time-bounded Kolmogorov complexity (see, e.g., [ABK+06]). In a bit more detail, in the proof of both results we are interested in establishing bounds on the Kolmogorov complexity of a string x. A way of doing this is by considering the string x as a source of "hardness" (e.g., view x as a hard truth-table) in the construction of a generator G^x. The typical analysis of a PRG provides a reconstruction routine, i.e., an algorithm implementing the proof that if we can break G^x using a distinguisher D, then x cannot be hard. In other words, we obtain bounds on the conditional time-bounded Kolmogorov complexity of x given D. Crucially, under assumptions such as $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, it is often possible to break the corresponding PRG G^x. This provides a powerful way of analysing the time-bounded Kolmogorov complexity of strings in the context of Theorems 18 and 19. More recently, the papers [Hir20b, Hir21] have highlighted the importance of a particular "direct product" generator $\mathsf{G}^x = \mathsf{DP}^x$, which has near-optimal "advice" complexity in its reconstruction procedure and provides tighter bounds on the complexity of x. In the *randomised* reconstruction procedure of DP^x, the advice depends on the particular choice of the random string employed by the procedure, which shows that for a noticeable fraction of random strings w, x has a small description if we are given the random string w. Now observe that this corresponds precisely to pK^t complexity! In previous work [Hir21], this issue is not present because the stronger assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ provides near-optimal derandomisation [BFP05] that allows one to directly get K^t bounds. However, the same PRG is not known to be available under the weaker assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$.

As explained in [GKLO22], while previous works have employed various techniques to *remove* randomness from their arguments in order to analyze K^t

complexity, the idea of *incorporating* randomness in the framework (via pK^t) comes with other benefits beyond the extension of results to the setting of randomised computations. For instance, [CHV22] established *fine-grained* connections between worst-case and average-case complexity. Among other results, they showed that if $\mathsf{NTIME}[n]$ can be deterministically solved in quasi-linear time on average, then $\mathsf{UP} \subseteq \mathsf{DTIME}[2^{O(\sqrt{n \log n})}]$. While the argument from [CHV22] requires the construction of an extremely fast PRG via a delicate analysis, the same result can be proved using pK^t complexity with a simpler proof [GKLO22].

As a potentially accessible direction, we pose the following problem related to Theorem 18 and Item 1 of Theorem 19.

Problem 22. *Show that if* $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$ *then* $\mathsf{NP} \subseteq \mathsf{BPTIME}[2^{O(n/\log n)}]$.

6.3 Learning Algorithms from Probabilistic Average-Case Easiness

This section describes an application of probabilistic Kolmogorov complexity to computational learning theory. More precisely, we show that if $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, then polynomial-size Boolean circuits can be (agnostically) PAC learned under any samplable distribution in polynomial time. While it is not hard to learn general Boolean circuits under a *worst-case* easiness assumption (e.g, $\mathsf{NP} \subseteq \mathsf{BPP}$) using Occam's razor (see, e.g., [KV94]), here we obtain an interesting consequence for learning under a *weaker* average-case easiness assumption.

The proof adapts a similar learning result from [HN21], established under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ (i.e., average-case easiness for *deterministic* algorithms). This exhibits a natural example of a result that can be lifted to the randomised setting with little effort via pK^t complexity.

Let C be a class of Boolean functions. In the PAC learning model, a learner has access to examples $(x, f(x))$ labelled according to an unknown function $f \in C$. The examples x are drawn according to an unknown probability distribution D_n supported over $\{0,1\}^n$. The goal of the learning algorithm is to produce, with high probability over its internal randomness and draw of labelled examples, a hypothesis h such that $\Pr_{x \sim D_n}[h(x) \neq f(x)] \leq \varepsilon$.

We say that the distribution $D_n \in \mathsf{Samp}[T(n)]/a(n)$ if it can be sampled by an algorithm that runs in time $T(n)$ and has advice complexity $a(n)$. (A sampler described by a uniform machine of code length a counts as advice of length a.) We consider the learnability of the class $C = \mathsf{SIZE}[s]$ of Boolean circuits of size at most $s(n)$, with respect to an unknown distribution D_n from $\mathsf{Samp}[T(n)]/a(n)$.

As in [HN21], the result described below also holds in the more challenging setting of *agnostic learning*, where the function f only needs to be *close* to some function in C. (See [GKLO22] for a concise presentation of this learning model.)

Theorem 23 (Agnostic Learning from Probabilistic Average-Case Easiness of NP [GKLO22])**.**
If DistNP $\subseteq$ AvgBPP*, then for any time constructible functions* $s, T, a\colon \mathbb{N} \to \mathbb{N}$*, and* $\varepsilon \in [0,1]$*,* SIZE$[s(n)]$ *is agnostic learnable on* Samp$[T(n)]/a(n)$ *in time* $\mathrm{poly}\big(n, \varepsilon^{-1}, s(n), T(n), a(n)\big)$.

For the proof of this result, the main idea is to design a *random-right-hand-side-refuter* (RRHS-refuter; see [Vad17, KL18]). In short, this is an algorithm that distinguishes the distribution $\big(x^{(1)}, \ldots, x^{(m)}, f(x^{(1)}), \ldots f(x^{(m)})\big)$ from the distribution $\big(x^{(1)}, \ldots, x^{(m)}, b^{(1)}, \ldots b^{(m)}\big)$, where each $x^{(i)}$ is picked from a fixed but unknown distribution D_n, $f \in C$ is a fixed but unknown function and each $b^{(i)}$ is a uniformly random bit. It is known that such an algorithm can be converted into an agnostic learner for C under the distribution D_n.

In [HN21] an efficient RRHS-refuter is constructed using an algorithm that estimates the K^t complexity of a given string, which can be shown to exist under the assumption that DistNP $\subseteq$ AvgP [Hir21]. In more detail, [HN21] proved that if a string is sampled from the first distribution, where D_n is efficiently samplable and f is computable by a polynomial size circuit, then it is likely to have bounded K^t complexity (for carefully chosen parameters m and t). On the other hand, using symmetry of information and optimal coding for K^t, which hold under an average-case easiness assumption [Hir21], it can be shown that a random string from the second distribution is likely to have large K^t complexity.

In contrast, under the weaker assumption that DistNP $\subseteq$ AvgBPP, we design an efficient algorithm that estimates the pK^t complexity of a given string, which is a more delicate measure than K^t. Combining this algorithm with the symmetry of information for pK^t (Theorem 20), which holds under the same probabilistic average-case easiness assumption, and the optimal coding result for pK^t (Theorem 14), we are able to construct in a similar way an efficient randomised RRHS-refuter. As before, this is sufficient to obtain the desired learning conclusion.

It would be interesting to understand if under the same average-case easiness assumption one can non-trivially learn general Boolean circuits with respect to an arbitrary distribution, i.e., in the standard sense of the PAC learning model.

Problem 24. *Suppose that* DistNP $\subseteq$ AvgBPP*. Is it possible to PAC learn Boolean circuits of size* $O(n)$ *(say, with error* $\varepsilon = 1/10$*) in time* $2^n/n^{\omega(1)}$*?*

We note that this would be possible (via Occam's Razor) if the same average-case easiness assumption led to stronger worst-case upper bounds for languages in NP, such as the conclusion that NP $\subseteq$ BPTIME$[2^{n^{0.499}}]$.

7 Probabilistic Versus Deterministic Time-Bounded Kolmogorov Complexity

We have seen that some questions that remain open for classical notions of time-bounded Kolmogorov complexity (such as Kt) can be unconditionally answered in the case of rKt, rK^t, and pK^t. For instance, we presented better bounds for primes with respect to rKt (Theorem 4) and rK^t (Theorem 6) in Section 4, and stated an optimal coding theorem for pK^t (Theorem 14) in Section 5. Moreover, we exhibit several applications of probabilistic time-bounded Kolmogorov complexity to algorithms and complexity in Section 6. It is perhaps a good point to discuss in more detail the relation between deterministic and probabilistic notions of Kolmogorov complexity.

It turns out that, *under strong enough derandomisation hypotheses*, for every string x, its deterministic and probabilistic time-bounded Kolmogorov complexities essentially coincide. For instance, for Kt and rKt we have the following relation.[21]

Theorem 25 ([Oli19])**.** *The following results hold.*

- *If* $\mathsf{promise\text{-}BPE} \subseteq \mathsf{promise\text{-}E}$*, then* $\mathsf{Kt}(x) \leq O(\mathsf{rKt}(x))$ *for every string* x*.*
- *If* $\mathsf{Kt}(x) \leq O(\mathsf{rKt}(x))$ *for every string* x*, then* $\mathsf{BPE} \subseteq \mathsf{E}/O(n)$*.*

In particular, rKt *and* Kt *are linearly related measures if* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}\left[2^{\Omega(n)}\right]$.

Note that the connection between derandomisation of probabilistic complexity classes and time-bounded Kolmogorov complexity holds in both directions: in a sense, collapsing Kt and rKt for every string x (up to a constant multiplicative factor) is essentially equivalent to the derandomisation of (promise) BPE, as stated in Theorem 25.

Similarly, under strong enough assumptions, we can show that $\mathsf{pK}^{\mathrm{poly}}(x)$, $\mathsf{rK}^{\mathrm{poly}}(x)$, and $\mathsf{K}^{\mathrm{poly}}(x)$ coincide up to an additive term of order $O(\log n)$.

Theorem 26 ([GKLO22])**.** *The following results hold.*

- *If* $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}\left[2^{\Omega(n)}\right]$*, then there is a polynomial* p *such that* $\mathsf{K}^{p(t)}(x) \leq \mathsf{rK}^t(x) + \log p(t)$*, for every* n*-bit string* x *and time bound* $t(n) \geq n$*.*

[21] Recall that $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$ refers to the set of languages that can be decided in deterministic time $2^{O(n)}$, while $\mathsf{BPE} = \mathsf{BTIME}[2^{O(n)}]$ is the set of languages that can be decided in probabilistic time $2^{O(n)}$. The promise version of E is defined in the natural way. Recall that for $\mathsf{promise\text{-}BPE}$ we do not enforce the acceptance probability of the randomised machine to be bounded away from $1/2$ on inputs that do not satisfy the promise.

- *If* $\mathsf{E} \not\subseteq \mathsf{i.o.NSIZE}\left[2^{\Omega(n)}\right]$, *then there is a polynomial* p *such that* $\mathsf{K}^{p(t)}(x) \leq \mathsf{pK}^t(x) + \log p(t)$, *for every* n*-bit string* x *and time bound* $t(n) \geq n$.

- *If* $\mathsf{BPE} \not\subseteq \mathsf{i.o.NSIZE}\left[2^{\Omega(n)}\right]$, *then there is a polynomial* p *such that* $\mathsf{rK}^{p(t)}(x) \leq \mathsf{pK}^t(x) + \log p(t)$, *for every* n*-bit string* x *and time bound* $t(n) \geq n$.

Proof. We describe the proof of the first item. The other two relations can be established by an appropriate modification of the argument, and we refer to [GKLO22] for the details.

Let $x \in \{0,1\}^n$, and let $t(n) \geq n$. First, the assumption $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}\left[2^{\Omega(n)}\right]$ implies that there is a PRG $G\colon \{0,1\}^{O(\log s)} \to \{0,1\}^s$ that $(1/s)$-fools size-s Boolean circuits and has running time $\mathrm{poly}(s)$ [IW97]. Suppose $\mathsf{rK}^t(x) \leq k$. Let $M \in \{0,1\}^k$ be a probabilistic machine of running time at most t that outputs x with probability at least $2/3$.[22] Consider the following function C on inputs of length t:

$$C(w) = 1 \iff M(w) = x.$$

Clearly, C can be implemented as a $\mathrm{poly}(t)$-size circuit. By definition, the acceptance probability of C is at least $2/3$. Consequently, there is a seed $z \in \{0,1\}^{O(\log t)}$ such that $C(G(z)) = 1$, which in turn implies that $M(G(z)) = x$. This means that, given the description of M and z, we can *deterministically* compute x in time $\mathrm{poly}(t)$. In particular, $\mathsf{K}^{p(t)} \leq k + \log p(t)$, for some large enough polynomial $p(\cdot)$. This polynomial is selected as a function of the overhead in running time and description length caused by the PRG. For this reason, it does not depend on x and t. This completes the proof. □

As a consequence of these (conditional) equivalences, new insights about probabilistic time-bounded Kolmogorov complexity can also shed light on the classical deterministic notions.[23] In particular, if one believes in the corresponding derandomisation assumptions, establishing certain results for rKt, rK^t, and pK^t can be seen as a necessary step before we are able to obtain similar statements for Kt and K^t. One such example is the task of showing better upper bounds on the time-bounded Kolmogorov complexity of prime numbers (Section 4).

Of course, one of the main advantages of probabilistic time-bounded Kolmogorov complexity is that certain results are known unconditionally. In particular, in applications there is often no need to rely on unproven conjectures from complexity theory.

[22] If M runs for more than t steps on some computation path, we simply truncate its computation.

[23] As a concrete example, after proving Theorem 11 in [LO21], we noticed that a similar result also holds for Kt, *unconditionally*. See [LO21] for more information on this.

8 Unconditional Hardness of Estimating Time-Bounded Kolmogorov Complexity

In this section, we turn our attention to *meta-computational* problems, which are problems that are themselves about computations and their complexity. An example of such a problem is MCSP (Minimum Circuit Size Problem), where we are given the truth table of a Boolean function $f\colon \{0,1\}^m \to \{0,1\}$ (represented as a Boolean string x of length $n = 2^m$) and a size bound s, and must decide if f can be computed by a Boolean circuit containing at most s gates. Similarly, we can consider the problem of computing the K^t complexity of an input string $x \in \{0,1\}^n$, where $t(n)$ is some fixed polynomial, such as $t(n) = n^3$. In both cases, it is not hard to see that we obtain a problem in NP. Due to their meta-computational nature, intriguing properties (e.g., [OPS19]), and connections to other areas such as learning theory (e.g., [CIKK16]) and cryptography (e.g., [LP20]), it is possible that the investigation of the complexity of meta-computational problems can offer a fruitful path towards a proof that $\mathsf{P} \neq \mathsf{NP}$.

Given the challenge of establishing strong unconditional lower bounds for problems in NP, it is also interesting to consider the complexity of computing other notions of time-bounded Kolmogorov complexity, such as Kt and rKt. For instance, given a string $x \in \{0,1\}^n$, can we efficiently estimate $\mathsf{Kt}(x)$? Note that this can be done in exponential time using a brute-force search, which places the decision version of this problem in $\mathsf{E} = \mathsf{DTIME}[2^{O(n)}]$. Intuitively, it seems that computing Kt and rKt should be computationally hard for the following reasons:

(*i*) It looks like we must perform an exhaustive search over machines of non-trivial description length.

(*ii*) Thanks to the definitions of Kt and rKt, even the mere act of checking whether a specific machine M prints the string x could require an exponential time simulation.

Note that (*ii*) is not present in problems such as MCSP. (We will revisit this intuition later in the section.)

The next result shows that MrKtP, the Minimum rKt Problem, is computationally hard for randomised algorithms. Indeed, even a gap version of the problem remains difficult. Note that the result provides an *unconditional* complexity lower bound for a natural problem.[24]

[24] As observed by [Oli19], the problem stated next can be solved in randomised exponential time.

Theorem 27 (Complexity Lower Bound for Estimating rKt [Oli19])**.** *For any* $0 < \varepsilon < 1$*, consider the promise problem* $\Pi^{\varepsilon}_{\mathsf{rKt}} = (\mathcal{YES}_n, \mathcal{NO}_n)_{n\in\mathbb{N}}$*, where*

$$\begin{aligned} \mathcal{YES}_n &= \{x \in \{0,1\}^n \mid \mathsf{rKt} \leq n^{\varepsilon}\}, \\ \mathcal{NO}_n &= \{x \in \{0,1\}^n \mid \mathsf{rKt}(x) \geq n-1\}. \end{aligned}$$

Then $\Pi^{\varepsilon}_{\mathsf{rKt}} \notin \mathsf{promise\text{-}BPTIME}[n^{\mathrm{polylog}(n)}]$.

Proof Sketch. The proof can be described in different ways. Here we provide a high-level exposition of the argument using insights from computational learning theory. For simplicity, we consider the weaker lower bound $\Pi^{\varepsilon}_{\mathsf{rKt}} \notin \mathsf{promise\text{-}BPP}$.

Assume towards a contradiction that $\Pi^{\varepsilon}_{\mathsf{rKt}} \in \mathsf{promise\text{-}BPP}$. We proceed as follows.

1. Under this assumption, it is possible to show that there is a (promise) *natural property* (in the sense of [RR97]) against functions computed by circuits of size $2^{\delta n}$, for some $\delta > 0$. In other words, we can efficiently distinguish truth-tables of bounded complexity from random truth-tables.

2. By the main result of [CIKK16], this implies that Boolean circuits of size s can be PAC learned under the uniform distribution with membership queries in time $\mathsf{poly}(s)$.

3. Exploring the connection between learning and circuit lower bounds from [OS17a], the existence of such learning algorithms implies that $\mathsf{BPE} \not\subseteq \mathsf{SIZE}(\mathrm{poly})$, where $\mathsf{SIZE}(\mathrm{poly})$ denotes the set of languages computed by Boolean circuits of polynomial size.

4. Finally, we argue that if $\Pi^{\varepsilon}_{\mathsf{rKt}}$ is in $\mathsf{promise\text{-}BPP}$ then $\mathsf{BPE} \subseteq \mathsf{SIZE}(\mathrm{poly})$. Roughly speaking, this step explores techniques from pseudorandomess [ABK+06] to show that every $L \in \mathsf{BPE}$ non-uniformly reduces to $\Pi^{\varepsilon}_{\mathsf{rKt}}$. Since by assumption this problem can be solved by efficient probabilistic algorithms, and such algorithms can be non-uniformly simulated by polynomial-size circuits, the inclusion follows.

Given that Items 3 and 4 are in contradiction, we obtain the desired complexity lower bound. (A proof that employs a different perspective is provided in [Oli19].) □

Curiously, establishing an analogous lower bound for MKtP remains a notorious open problem (see, e.g., [ABK+06]). Here MKtP refers to the problem of deciding, given a string x and a positive integer s, whether $\mathsf{Kt}(x) \leq s$. While it is believed that $\mathsf{MKtP} \notin \mathsf{P}$, we currently only know how to resolve the randomised

version of the problem (Theorem 27).[25] This provides another setting where probabilistic time-bounded Kolmogorov complexity offers an advantage over its deterministic counterpart. Note that Theorem 27 implies that $\mathsf{MKtP} \notin \mathsf{BPP}$ under a derandomisation assumption (Theorem 25).

Before presenting a different lower bound, we revise our initial intuition about the hardness of computing rKt and Kt. In light of Corollary 12, a result established after [Oli19], we now understand that the hardness of the gap version of MrKtP can be blamed on Item (*ii*) only. Interestingly, an unexpected algorithmic result sheds light on the hardness of estimating rKt complexity. At the same time, this tells us that different techniques will be needed to understand the computational hardness of problems such as MCSP or computing K^t, where the hardness must come from the analogue of Item (*i*).

Next, we discuss a complexity lower bound for estimating the $\mathsf{rK}^{\mathrm{poly}}$ complexity of an input string.

Theorem 28 (Complexity Lower Bound for Estimating $\mathsf{rK}^{\mathrm{poly}}$ [LOS21])**.** *For any $0 < \varepsilon < 1$ and $d \geq 1$ there exists a constant $k \geq 1$ for which the following holds. Consider the promise problem $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} = (\mathcal{YES}_n, \mathcal{NO}_n)_{n\in\mathbb{N}}$, where*

$$\begin{aligned} \mathcal{YES}_n &= \{x \in \{0,1\}^n \mid \mathsf{rK}^t(x) \leq n^{\varepsilon}\}, \\ \mathcal{NO}_n &= \{x \in \{0,1\}^n \mid \mathsf{rK}^t(x) \geq n-1\}, \end{aligned}$$

and $t(n) = n^k$. Then $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} \notin \mathsf{promise\text{-}BPTIME}[n^d]$.

Proof. We establish the weaker result that $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} \notin \mathsf{promise\text{-}DTIME}[n^d]$. The lower bound against probabilistic time can be established in a similar way, using that the pseudodeterministic PRG from Theorem 7 also fools probabilistic algorithms (see [LOS21] for the details).

Fix $0 < \varepsilon < 1$ and $d \geq 1$. Let $\varepsilon' = \varepsilon/2$, $d' = 1$, and $c' = d$. Instantiate the pseudodeterministic PRG from Theorem 7 with the parameters ε', c', and d', and assume that $G_n \colon \{0,1\}^{n^{\varepsilon'}} \to \{0,1\}^n$ can be computed probabilistically in time $n^{k'}$, for some constant k' (when provided with the correct advice bit $\alpha'(n)$). We let $k = 2k'$.

Now suppose, towards a contradiction, that $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} \in \mathsf{promise\text{-}DTIME}[n^d]$. Let A be a deterministic algorithm running in time n^d that accepts $\mathcal{YES}_n$ and rejects $\mathcal{NO}_n$, for every large enough n. We argue that the existence of A contradicts the infinitely often guarantee of pseudorandomness provided by the PRG G_n. Indeed, fix a large enough input length n for which G_n succeeds. On the one hand, by our

[25] The proof of Theorem 27 explores randomised computation to perform an indirect diagonalisation, and it is not clear how to implement a similar strategy when only deterministic computations are available.

choice of k and ε', it is easy to see that every string $y \in \{0,1\}^n$ in the image of G_n satisfies $\mathsf{rK}^{n^k}(y) \leq n^{\varepsilon}$. For this reason, $\Pr_{z\sim\{0,1\}^{n^{\varepsilon'}}}[A(G(z)) = 1] = 1$. On the other hand, by a counting argument, a random string $x \sim \{0,1\}^n$ satisfies $\mathsf{rK}^{n^k}(x) \geq n - 1$ with probability $\Omega(1)$ (Proposition 3). This implies that $\Pr_{x\sim\{0,1\}^n}[A(x) = 1] \leq 1 - \Omega(1)$, since A rejects strings in $\mathcal{NO}_n$. Now notice that this violates the pseudorandomness of G_n. In other words, we get that $\Pi^{\varepsilon,k}_{\mathsf{rK}^t} \notin \mathsf{promise\text{-}DTIME}[n^d]$. □

A complexity lower bound for computing K^t against deterministic polynomial-time algorithms and for $t = n^{\omega(1)}$ was established by Hirahara [Hir20b] using different techniques. In both cases, the time bound in the definition of the Kolmogorov complexity measure is larger than the time bound of the algorithm trying to compute or estimate Kolmogorov complexity. Needless to say, it would be extremely interesting to establish a complexity lower bound for computing Kolmogorov complexity with respect to a *fixed* polynomial t in K^t or rK^t that holds against *arbitrary* polynomial-time algorithms (see [LP20]).

A lower bound question that should be more accessible is presented next.

Problem 29 (Exponential Hardness of Estimating rKt)**.** *Show that for any constant* $0 < \varepsilon < 1$ *there is a constant* $\delta > 0$ *such that* $\Pi^{\varepsilon}_{\mathsf{rKt}} \notin \mathsf{promise\text{-}BPTIME}[2^{n^{\delta}}]$.

9 Constructing Strings of Large rKt Complexity and Hierarchy Theorems

The problem of *explicitly* constructing mathematical objects of different types (beyond merely showing their existence) has received much attention in computer science and mathematics. For instance, in Section 1 we described the problem of deterministically producing an n-bit prime. In this section, we are interested in the problem of constructing *incompressible strings*. Some problems of this form are particularly challenging, since given a long incompressible string (e.g., with respect to circuit size or $\mathsf{K}^{\mathrm{poly}}$ complexity), several other constructions problems can be solved (see, e.g., [San12, Kor21]).

In more detail, here we consider the problem of explicitly constructing strings that have large rKt complexity. To provide intuition, let us first consider the much simpler case of Kt complexity. Our goal is to design a deterministic algorithm that, given 1^n, outputs an n-bit string x such that $\mathsf{Kt}(x) \geq n/10$. Does this problem admit a polynomial-time algorithm? It is easy to see that this problem cannot be solved in time $2^{o(n)}$. Indeed, it follows from the very definition of Kt complexity that any deterministic algorithm $A(1^n)$ running in time $2^{o(n)}$ can only print an n-bit string of Kt complexity $o(n)$. However, it is not hard to see that this explicit

construction problem can be solved in time $2^{O(n)}$ via an exhaustive search (for instance, by enumerating all strings produced in time $\leq 2^{n/10}$ by machines of description length $\leq n/10$).

Similarly, we ask if there is an algorithm that runs in time $2^{O(n)}$ and produces an n-bit string x such that $\mathsf{rKt}(x) \geq n/10$. The natural brute-force approach to solve this problems involves the simulation of *randomised* algorithms. For this reason, we relax our goal as follows: Is there a *randomised* algorithm $A(1^n)$ that runs in time $2^{O(n)}$ and outputs with probability at least $2/3$ a *fixed* n-bit string w_n such that $\mathsf{rKt}(w_n) \geq n/10$? In other words, we would like to have a *pseudodeterministic* construction of strings of large rKt complexity, in the sense of [GG11].

A careful inspection of the natural brute-force approach that works for Kt reveals that it simply does not work in the case of rKt: roughly speaking, the simulation of different randomised machines comes with uncertainties, and it is not clear if after all the simulations we isolate the same string w_n with high probability.

In [LOS21], we connected the problem of constructing strings of large rKt complexity to the longstanding question of establishing a strong time hierarchy theorem for probabilistic computations. Recall that, while it is known that $\mathsf{BPEXP} \not\subseteq \mathsf{BPP}$, it is consistent with current knowledge that inclusions such as $\mathsf{BPTIME}[2^n] \subseteq \mathsf{BPTIME}[2^{n^{0.01}}]$ and $\mathsf{BPTIME}[n^{50}] \subseteq \mathsf{BPTIME}[n^2]$ might hold.[26]

Theorem 30 (Explicit Construction Problem for rKt and Probabilistic Time Hierarchies)**.** *The following statements are equivalent:*

(1) Pseudodeterministic construction of strings of large rKt complexity: *There is a constant $\varepsilon > 0$ and a randomised algorithm A that, given m, runs in time $2^{O(m)}$ and outputs with probability at least $2/3$ a fixed m-bit string w_m such that $\mathsf{rKt}(w_m) \geq \varepsilon m$.*

(2) Strong time hierarchy theorem for probabilistic computation: *There are constants $k \geq 1$ and $\lambda > 0$ for which the following holds. For any constructive function $n \leq t(n) \leq 2^{\lambda \cdot 2^n}$, there is a language $L \in \mathsf{BPTIME}[(t(n)^k]$ such that $L \notin \mathsf{i.o.BPTIME}[t(n)]/\log t(n)$.*

The proof of Theorem 30 is elementary, and proceeds by associating with a language L a sequence of truth-tables, one for each input length n (each truth-table can be seen as a string of length $m = 2^n$). For a sketch of the argument and a detailed proof, see [LOS21].

Note that the connection between the explicit (pseudodeterministic) construction problem for rKt and hierarchy theorems goes in both ways. More generally, [LOS21] explored the fruitful relation between pseudodeterministic PRGs (see

[26] Some separations have been established if we allow advice bits in the upper bound and lower bound. For instance, $\mathsf{BPTIME}[n^{50}]/1 \not\subseteq \mathsf{BPTIME}[n^2]/1$ (see, e.g., [Bar02, FS04]).

Section 4), the explicit construction problem for rKt complexity, and hierarchy theorems for probabilistic time to make advances in all these areas. On the other hand, [LO21] connected rKt complexity and its coding theorem (Theorem 10) to the study of time hierarchy theorems for sampling distributions (cf., [Wat14]).

10 Concluding Remarks

We presented key results in probabilistic Kolmogorov complexity and applications to several areas, including explicit constructions, complexity lower bounds, sampling algorithms, average-case complexity, and learning theory. The probabilistic measures rK^t, pK^t, and rKt are particularly useful in settings that involve randomised algorithms. While it is quite possible for these complexity measures to be essentially equivalent to their deterministic counterparts (Section 7), they allow us to obtain unconditional results that do not rely on derandomisation assumptions. In some cases, probabilistic Kolmogorov complexity can significantly simplify existing arguments or is the only known approach to certain results.

The results presented in the preceding sections naturally suggest several problems and directions. For example, we believe that it should be possible to make progress on the following fronts:

- Designing improved pseudodeterministic PRGs and obtaining better upper bounds on the rKt complexity of prime numbers.

- Establishing new unconditional lower bounds on the complexity of meta-computational problems such as MKtP and MrKtP.

For a more precise formulation of these problems, we refer to the concrete questions stated in the corresponding sections of the article (Section 4 and Section 8). Additional questions of interest are presented in other parts of the survey.

Given the number of recent advances and applications of time-bounded Kolmogorov complexity to algorithms and complexity theory (see Section 1), it is hard to predict which directions will be more fruitful. Nevertheless, we are particularly optimistic about the role that probabilistic Kolmogorov complexity can take in the investigation of the relations between average-case complexity and worst-case complexity, cryptography, and learning algorithms. In particular, analogously to results of [Hir21], under the assumption that $\mathsf{DistNP} \subseteq \mathsf{AvgBPP}$, all main pillars of Kolmogorov complexity are known to hold for pK^t complexity: incompressibility (Proposition 3), coding theorem (Theorem 14), language compression (Theorem 17), and symmetry of information (Theorem 20). Taking into account the wide applicability of these results and the ubiquitous role of randomised algorithms in theoretical computer science, we expect to see further

developments in average-case complexity powered by tools and perspectives from probabilistic Kolmogorov complexity.

References

[Aar14] Scott Aaronson. The equivalence of sampling and searching. *Theory Comput. Syst.*, 55(2):281–298, 2014.

[ABK+06] Eric Allender, Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger. Power from random strings. *SIAM J. Comput.*, 35(6):1467–1493, 2006.

[AF09] Luis Antunes and Lance Fortnow. Worst-case running times for average-case algorithms. In *Conference on Computational Complexity* (CCC), pages 298–303, 2009.

[AFvM01] Luis Antunes, Lance Fortnow, and Dieter van Melkebeek. Computational depth. In *Conference on Computational Complexity* (CCC), pages 266–273, 2001.

[AKS02] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math.*, 2:781–793, 2002.

[All92] Eric Allender. Applications of time-bounded Kolmogorov complexity in complexity theory. In *Kolmogorov complexity and computational complexity*, pages 4–22. Springer, 1992.

[All01] Eric Allender. When worlds collide: Derandomization, lower bounds, and Kolmogorov complexity. In *International Conference on Foundations of Software Technology and Theoretical Computer Science* (FSTTCS), pages 1–15. Springer, 2001.

[All17] Eric Allender. The complexity of complexity. In *Computability and Complexity*, pages 79–94. Springer, 2017.

[All21] Eric Allender. Vaughan Jones, Kolmogorov complexity, and the new complexity landscape around circuit minimization. *New Zealand Journal of Mathematics*, 52:585–604, 2021.

[Bar02] Boaz Barak. A probabilistic-time hierarchy theorem for "slightly non-uniform" algorithms. In *International Workshop on Randomization and Approximation Techniques* (RANDOM), pages 194–208, 2002.

[BFP05] Harry Buhrman, Lance Fortnow, and Aduri Pavan. Some results on derandomization. *Theory Comput. Syst.*, 38(2):211–227, 2005.

[BLvM05] Harry Buhrman, Troy Lee, and Dieter van Melkebeek. Language compression and pseudorandom generators. *Comput. Complex.*, 14(3):228–255, 2005.

[BMVZ18] Bruno Bauwens, Anton Makhlin, Nikolai K. Vereshchagin, and Marius Zimand. Short lists with short programs in short time. *Comput. Complex.*, 27(1):31–61, 2018.

[BT06] Andrej Bogdanov and Luca Trevisan. Average-case complexity. *Found. Trends Theor. Comput. Sci.*, 2(1), 2006.

[CHV22] Lijie Chen, Shuichi Hirahara, and Neekon Vafa. Average-case hardness of NP and PH from worst-case fine-grained assumptions. In *Innovations in Theoretical Computer Science* (ITCS), 2022.

[CIKK16] Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning algorithms from natural proofs. In *Conference on Computational Complexity* (CCC), pages 10:1–10:24, 2016.

[For04] Lance Fortnow. Kolmogorov complexity and computational complexity. *Complexity of Computations and Proofs. Quaderni di Matematica*, 13, 2004.

[FS04] Lance Fortnow and Rahul Santhanam. Hierarchy theorems for probabilistic polynomial time. In *Symposium on Foundations of Computer Science* (FOCS, pages 316–324, 2004.

[GG11] Eran Gat and Shafi Goldwasser. Probabilistic search algorithms with unique answers and their cryptographic applications. *Electronic Colloquium on Computational Complexity* (ECCC), 18:136, 2011.

[GK22] Halley Goldberg and Valentine Kabanets. A simpler proof of the worst-case to average-case reduction for polynomial hierarchy via symmetry of information. *Electron. Colloquium Comput. Complex.*, 7:1–14, 2022.

[GKLO22] Halley Goldberg, Valentine Kabanets, Zhenjian Lu, and Igor C. Oliveira. Probabilistic Kolmogorov complexity with applications to average-case complexity. In *Computational Complexity Conference* (CCC), 2022.

[Hir18] Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *Symposium on Foundations of Computer Science* (FOCS), pages 247–258, 2018.

[Hir20a] Shuichi Hirahara. Characterizing average-case complexity of PH by worst-case meta-complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 50–60, 2020.

[Hir20b] Shuichi Hirahara. Unexpected hardness results for Kolmogorov complexity under uniform reductions. In *Symposium on Theory of Computing* (STOC), pages 1038–1051, 2020.

[Hir21] Shuichi Hirahara. Average-case hardness of NP from exponential worst-case hardness assumptions. In *Symposium on Theory of Computing* (STOC), pages 292–302, 2021.

[Hir22a] Shuichi Hirahara. Meta-computational average-case complexity: A new paradigm toward excluding heuristica. *Bull. EATCS*, 136, 2022.

[Hir22b] Shuichi Hirahara. Symmetry of information in heuristica. Manuscript, 2022.

[HN21] Shuichi Hirahara and Mikito Nanashima. On worst-case learning in relativized heuristica. In *Symposium on Foundations of Computer Science* (FOCS), 2021.

[HS22] Shuichi Hirahara and Rahul Santhanam. Errorless versus error-prone average-case complexity. In *Innovations in Theoretical Computer Science Conference* (ITCS), 2022.

[Ila20] Rahul Ilango. Connecting Perebor conjectures: Towards a search to decision reduction for minimizing formulas. In *Computational Complexity Conference* (CCC), 2020.

[Ila21] Rahul Ilango. The minimum formula size problem is (ETH) hard. In *Symposium on Foundations of Computer Science* (FOCS), pages 427–432, 2021.

[ILO20] Rahul Ilango, Bruno Loff, and Igor C. Oliveira. NP-hardness of circuit minimization for multi-output functions. In *Computational Complexity Conference* (CCC), 2020.

[IP01] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[IRS21] Rahul Ilango, Hanlin Ren, and Rahul Santhanam. Hardness on any samplable distribution suffices: New characterizations of one-way functions by meta-complexity. *Electron. Colloquium Comput. Complex.*, page 82, 2021.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Symposium on Theory of Computing* (STOC), pages 220–229. ACM, 1997.

[KL18] Pravesh K. Kothari and Roi Livni. Improper learning by refuting. In *Innovations in Theoretical Computer Science Conference* (ITCS), pages 55:1–55:10, 2018.

[Ko91] Ker-I Ko. On the complexity of learning minimum time-bounded Turing machines. *SIAM J. Comput.*, 20(5):962–986, 1991.

[Kor21] Oliver Korten. The hardest explicit construction. In *Symposium on Foundations of Computer Science* (FOCS), pages 433–444, 2021.

[Kra22] Jan Krajíček. Information in propositional proofs and algorithmic proof search. *The Journal of Symbolic Logic*, 2022.

[KV94] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[Lee06] Troy Lee. *Kolmogorov complexity and formula lower bounds*. PhD thesis, University of Amsterdam, 2006.

[Lev84] Leonid A. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61(1):15–37, 1984.

[LO87] Jeffrey C. Lagarias and Andrew M. Odlyzko. Computing π(x): An analytic method. *J. Algorithms*, 8(2):173–191, 1987.

[LO21] Zhenjian Lu and Igor C. Oliveira. An efficient coding theorem via probabilistic representations and its applications. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 94:1–94:20, 2021.

[LOS21] Zhenjian Lu, Igor C. Oliveira, and Rahul Santhanam. Pseudodeterministic algorithms and the structure of probabilistic time. In *Symposium on Theory of Computing* (STOC), pages 303–316, 2021.

[LOZ22] Zhenjian Lu, Igor C. Oliveira, and Marius Zimand. Optimal coding theorems in time-bounded Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), 2022.

[LP20] Yanyi Liu and Rafael Pass. On one-way functions and Kolmogorov complexity. In *Symposium on Foundations of Computer Science* (FOCS), pages 1243–1254, 2020.

[LP21] Yanyi Liu and Rafael Pass. On the possibility of basing cryptography on EXP≠BPP. In *Annual International Cryptology Conference* (CRYPTO), pages 11–40, 2021.

[LV92] Ming Li and Paul M. B. Vitányi. Average-case complexity under the universal distribution equals worst-case complexity. *Inf. Process. Lett.*, 42(3):145–149, 1992.

[LV19] Ming Li and Paul M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer, 2019.

[Oli19] Igor C. Oliveira. Randomness and intractability in Kolmogorov complexity. In *International Colloquium on Automata, Languages, and Programming* (ICALP), pages 32:1–32:14, 2019.

[OPS19] Igor C. Oliveira, Ján Pich, and Rahul Santhanam. Hardness magnification near state-of-the-art lower bounds. In *Computational Complexity Conference* (CCC), pages 27:1–27:29, 2019.

[OS17a] Igor C. Oliveira and Rahul Santhanam. Conspiracies between learning algorithms, circuit lower bounds, and pseudorandomness. In *Computational Complexity Conference* (CCC), pages 18:1–18:49, 2017.

[OS17b] Igor C. Oliveira and Rahul Santhanam. Pseudodeterministic constructions in subexponential time. In *Symposium on Theory of Computing* (STOC), pages 665–677, 2017.

[RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.

[RS21] Hanlin Ren and Rahul Santhanam. Hardness of KT characterizes parallel cryptography. In *Computational Complexity Conference* (CCC), pages 35:1–35:58, 2021.

[RSZ21] Andrei E. Romashchenko, Alexander Shen, and Marius Zimand. 27 open problems in Kolmogorov complexity. *SIGACT News*, 52(4):31–54, 2021.

[San12] Rahul Santhanam. The complexity of explicit constructions. *Theory Comput. Syst.*, 51(3):297–312, 2012.

[Sip83] Michael Sipser. A complexity theoretic approach to randomness. In *Symposium on Theory of Computing* (STOC), pages 330–335, 1983.

[SUV17] Alexander Shen, Vladimir Andreyevich Uspensky, and Nikolay Vereshchagin. *Kolmogorov Complexity and Algorithmic Randomness*. American Mathematical Society, 2017.

[TCH12] Terence Tao, Ernest Croot, III, and Harald Helfgott. Deterministic methods to find primes. *Math. Comp.*, 81(278):1233–1246, 2012.

[Vad17] Salil P. Vadhan. On learning vs. refutation. In *Conference on Learning Theory* (COLT), 2017.

[Wat14] Thomas Watson. Time hierarchies for sampling distributions. *SIAM J. Comput.*, 43(5):1709–1727, 2014.

The Theory Blogs Column

by

Luca Trevisan

Bocconi University
Via Sarfatti 25, 20136 Milano, Italy
L.Trevisan@UniBocconi.it
https://lucatrevisan.github.io

In this issue, Professor Scott Aaronson, the David J. Bruton Centennial Professor at UT Austin, Director of the Quantum Information Center, and Fellow of the ACM, will answer our questions about his blog. You won't believe his answer to our fifth question!

Scott writes *shtetl optimized*, one of the most widely read blogs on theoretical computer science and quantum computing. In this issue, Scott tells us what have been the inspirations and anti-inspirations for his blog, what's up with the name of the blog, how computer science theory blogs can influence research, and what are some topics that he feels strongly about. He also revisits some of his older posts for us.

Scott's blog is at https://scottaaronson.blog//

SHTETL OPTIMIZED

A Conversation with Scott Aaronson

Q. Scott, thanks for taking the time to talk about your blog to our readers. When did you start your blog, and what motivated you to start? Also, can you tell us the story, if there is one, behind the name?

I started the blog in Fall 2005, mostly out of boredom. At that point, friends had been urging me to start a blog for several years ("you seem to have a lot of opinions!"), but I'd resisted them, giving reasons like the time commitment, the ephemeral nature of the medium, and the danger of making enemies and offending people. Incidentally, it would later turn out that all of those reasons were 100% valid! But on one particular night in 2005 ... well, I don't remember exactly what was going through my mind, but setting up a blog temporarily *seemed like* a pretty risk-free experiment. Then the experiment quickly snowballed to become an inextricable part of who I was, and the rest is history.

As for the blog's title, with your kind indulgence I'll simply quote what I said when John Horgan asked me the same question in 2016:

> Shtetls were Jewish villages in pre-Holocaust Eastern Europe. They're where all my ancestors came from—some actually from the same place (Vitebsk) as Marc Chagall, who painted the fiddler on the roof. I watched Fiddler many times as a kid, both the movie and the play. And every time, there was a jolt of recognition, like: "So that's the world I was designed to inhabit. All the aspects of my personality that mark me out as weird today, the obsessive reading and the literal-mindedness and even the rocking back and forth—I probably have them because back then they would've made me a better Talmud scholar, or something." So as I saw it, the defining question of my life was whether I'd be able to leverage these traits from a world that no longer existed, for the totally different world into which I was born.
>
> Of course, there are pockets where the shtetl still does exist; there are orthodox Jews. As it happens, I went to an orthodox Hebrew day school, where I was exposed to that. But by the time I was 12, and was reading Bertrand Russell and Richard Dawkins and Carl Sagan and Isaac Asimov and so forth, it was obvious to me that I could never be a believer in any conventional sense, even if I'm happy to

> use Einsteinian pseudo-religious language, as in "why did God make the world quantum rather than classical?" So from then on, the thing I yearned for was a community that would be as welcoming of intellectual obsessives as a yeshiva was—but without any unquestioned dogmas or taboos, where absolutely anything could be revised based on evidence, and which was open to new ideas from anyone of any ethnicity.

Q. Your posts vary very broadly in content and tone, from rather technical expositions to commentary on social or political topics. Did you have some inspiration or model for what you wanted your expository technical posts to be like? What about the non-technical posts?

When I started *Shtetl-Optimized*, my immediate inspirations were the other math, CS, and physics blogs that I regularly read back then, many of which are still active today. I'm talking especially about the blogs of Lance Fortnow, Sean Carroll, Peter Woit, and Dave Bacon. I was also heavily influenced by John Baez's "This Week's Finds in Mathematical Physics," which was a science blog before that concept even existed; and by Chris Fuchs's voluminous, chatty correspondence with other physicists (which he posted on the arXiv) about the foundations of quantum mechanics. Meanwhile, Luboš Motl's blog "The Reference Frame" was a sort of *anti*-inspiration: a cautionary lesson of what my blog could become if I succumbed to the temptation to ridicule and insult those I disagreed with.

Beyond that, there were all the writers who I'd read and admired as a teenager and a young adult, and who I'd broadly categorize as "defenders of the Enlightenment": for example, Carl Sagan, Isaac Asimov, Richard Dawkins, Steven Pinker, Bertrand Russell, Martin Gardner, Steven Weinberg, and Rebecca Goldstein. These writers had (and have) varied politics, from quasi-socialist to quasi-libertarian, and varied expertise and interests, from biology to linguistics to astronomy to philosophy to math. What they shared, you might say, was a radical vision of the unity of knowledge. Like, they'd often be at pains to explain some scientific or philosophical idea as lucidly as possible (and they were all damn good at that). But then a paragraph later they'd be on a soapbox, passionately arguing for what they saw as sanity on some moral or social question. There was no firewall between two; it was all part of the same stew.

Somehow, these writers hadn't got the memo that the whole Enlightenment project, of fixing our broken civilization through science and reason and clear prose, had crashed and burned by WWII or the 60s or whatever; that it was now passé and fit only for knowing sneers. They hadn't heard that there were no longer any profound truths about the nature of the world, that it was all just our tribe's claims to power versus the enemy tribe's, plus some technical know-how of no

broader significance. So these writers persisted in unselfconsciously thrusting scientific and philosophical gems in front of the reader, as if to ask "but how can you persist in unreason when there's *this*?" If Sagan, Dawkins, and the rest didn't always draw a clear line between popularizing science and ideology, that was because popularizing science *was* a central part of their ideology: they burned with an almost-religious conviction (ironically, given their close association with atheism) that the truth could set us free.

I'm acutely aware that I'll never measure up to any of these writers. But their work is the background for almost everything I've tried to do as a blogger.

Q. Your technical posts often get many thoughtful comments and stimulate fairly deep discussions. Have you ever proved a result or formulated a conjecture as a consequence of a discussion in your comment section?

Yes. Probably the best example involves a post from just two years ago, which advertised my upcoming survey article entitled "The Busy Beaver Frontier." (Recall that BB(n), the n^{th} value of the Busy Beaver function, is the maximum number of steps other than infinity that an n-state Turing machine can make on an initially blank tape.) Now, the wonderful thing about a topic like the Busy Beaver function is that so little background is needed to pose new questions or even make original contributions. So, what happened is that a bunch of *Shtetl-Optimized* readers, especially Joshua Zelinsky, Bruce Smith, Nick Drozd, and "Job" (a pseudonym), kept suggesting new variations, such as the "Lazy Beaver" function, defined as LB(n) = the least T for which no n-state Turing machine halts after exactly T steps. And they posed new questions: for example, are n-state Busy Beaver machines essentially unique? are they strongly connected? And they proved new results: for example, that BB($n+1$) $\geq$ BB(n) + 3 for all n. And then I got in on the action and worked with them on some of these things, and every day I had to revise my survey article to account for what was happening in my comment section! I'm still hoping to write a followup paper with Smith about the theory of the Lazy Beaver function.

Q. How else can blogs influence TCS research?

Yeah, more often my blog's influence on my research has been less direct. As one example, debates about quantum supremacy experiments in my comment section played an important role in motivating the work that I subsequently did with Sam Gunn, then an undergrad at UT Austin, on the classical hardness of spoofing Google's "Linear Cross-Entropy Benchmark" (LXEB).

But I've found that maybe the *best* role blogs can play in TCS research, is as a workshop for ideas and open problems that aren't yet ready for formal publication. I've repeatedly been motivated by questions that Lance Fortnow posed on his blog: for example, whether $\mathsf{NP}^{\mathsf{BQP}} \subseteq \mathsf{BQP}^{\mathsf{NP}}$, a question that William

Kretschmer, DeVon Ingram, and I recently answered relative to an oracle. My 2007 post "The Aaronson $25.00 Prize" offered, well, a $25 prize for an interactive protocol that would prove the results of any quantum computation to a classical skeptic. There's since been *spectacular* progress on that problem—I've awarded not one but three $25.00 prizes!—although it's hard to determine what causal role, if any, my prize played. Likewise, my 2006 post "The Quantum PCP Manifesto" was the first place the possibility of a quantum analogue of the PCP Theorem was discussed in writing, and my 2008 post "Arithmetic natural proofs theory is sought" did the same for an arithmetic version of the Razborov-Rudich barrier. Since then, both subjects have become orders-of-magnitude deeper than whatever trivial first ideas I had, but the latest papers on them still need to cite my blog posts, because the blog posts were first! I'd like to imagine that the blog posts did play a role in catalyzing these subjects, although I don't know for sure.

Q. Maybe I am mistaken, but I feel that people in academia, and especially in science and engineering, are increasingly reluctant to speak out about controversial subjects (I know I am). In your blog you don't shy away from weighing in on social and political issues that you feel strongly about. I wonder what are you thoughts about the role of public intellectuals that theoretical computer scientists could fill, and what are public debates where you would like to see more voices coming from our community.

This question feels ironic for me, because I *don't* blog about controversial subjects in the freewheeling way that I used to! There are two reasons for this: one about the whole Internet, the other about me.

In 2005, the open Internet still felt like a playground, where sure, you could get into vicious verbal fights, but then you'd make up the next day, and in any case very few "real-life" friends or colleagues read what was said, and even fewer remembered or cared. So as a blogger, you might as well experiment, take risks, push the boundaries of what you were allowed to say: after all, wasn't that how you became worth reading?

But as I'm far from the first to point out, something changed dramatically around 2013. Since then, the open Internet has felt more like a bullet-riddled battlefield, with the trained snipers of Twitter and Reddit and so forth waiting to take out anybody who foolishly blunders into their sights with an unapproved opinion. And crucially, even if you tell yourself that you're ready to face the snipers—that you're armed with your courage and experience, and your loyal friends and family, and (of course) the protections of academic tenure—there's still the question of *time*! If, like me, you feel a neurotic obligation to respond point-by-point when people criticize or attack you, then when hundreds or thousands people do exactly that, working through all of it can take weeks or months, as it repeatedly has for me.

The other thing that's changed is my situation in life. When I started blogging, I was an obscure postdoc at the University of Waterloo. Not many people cared what I said. Now, though, I can't say anything of any importance, without *creating the news* that Professor Aaronson, David J. Bruton Centennial Professor at UT Austin and Director of the Quantum Information Center and Fellow of the ACM, said X. And my critics then ask: but won't Aaronson's vulnerable students now be intimidated from saying not(X)? How does what Aaronson said reflect on his department and university, or his fellow Enlightenment liberals, or the whole fields of TCS or quantum information? Blogging is a lot less *fun* when you've got all of that, so to speak, peering over your shoulder whenever you type.

All the same, I *do* still weigh in on controversial issues—often because the issues really matter to me, and I'd feel complicit if I didn't use my blog to try to make a difference! In recent years, some of my pet issues have been:

1. the defense of rigorous science and math education, including the increasingly-maligned tracking, acceleration, magnet programs, and standardized tests;

2. the defense of the American scientific enterprise, including support for visas for international students, and opposition to punitive taxes on PhD students;

3. compassion rather than contempt for the social difficulties faced by those with autism-spectrum or Aspberger's-syndrome traits (things that aren't ... *entirely* unknown in math and TCS),

4. advocacy of a "WWII-style response" to the COVID-19 pandemic (which could've included, for example, mass vaccination starting as early as Spring 2020);

5. the necessity of nuclear power in confronting the climate emergency; and

6. the defense of liberal democracy against an authoritarianism that's now resurgent around the world.

You might notice that, of these six, only the first three are plausibly informed by any special experience or knowledge that I have! The others are simply things that I deeply care about as a human being.

In any case, there are countless societal issues where lots of theoretical computer scientists *do* have special expertise, even if *I* don't. A few obvious examples here are encryption policy, digital privacy, and fairness and bias in machine learning. Beyond that, there are all sorts of philosophical questions that the wider intellectual world cares about—e.g., at what point (if any) should machines be

considered conscious? what does Gödel's Theorem mean for the nature of mathematical knowledge? is the universe a computer simulation? how does quantum mechanics bear on free will?—where, with all due humility, I think the perspectives of theoretical computer scientists like ourselves can be pretty clarifying. So when the world hands us these opportunities for broader relevance on a platter, I really hope my colleagues won't shy away from them!

Q. I would like to ask you to pick one or a couple of your favorite posts, and tell us about it/them.

There were dozens of posts that amused me or others at the time. Looking back, though, I'm proudest of the posts that made an actual difference to something. One example would be my recent post "Win a Scott Aaronson Speculation Grant!," which helped me distribute $200,000 (which the philanthropist Jaan Tallinn had placed in my charge) to some wonderful math and science enrichment programs. Many of those programs wouldn't have otherwise been able to continue, and I wouldn't have learned about them if not for the blog. Then there was "First They Came for the Iranians," about the devastating effect that Trump's travel ban was having on my Iranian PhD student Saeed Mehraban, which became the most-shared post in my blog's history and got media coverage. And of course all the posts I wrote pouring cold water on claims of quantum computers getting dramatic speedups for NP-complete problems and the like, which felt therapeutic to write.

Another, very different example would be my 2014 post "Why I Am Not An Integrated Information Theorist (or, The Unconscious Expander)," where I laid out what I saw as a fatal problem with Giulio Tononi's popular Integrated Information Theory (IIT) of consciousness: namely, that the theory predicts immense amounts of consciousness in (e.g.) simple graphs made of XOR gates, which don't even do anything *intelligent*. My post surely wasn't the first place this issue had been mentioned, but it somehow provoked famous scientists and philosophers like David Chalmers, Max Tegmark, and Tononi himself to respond right on my blog, until finally this critique of IIT became impossible to ignore, and I even kept getting invited to workshops and so forth to repeat what I'd said in the blog post! Notably, in writing this post, I drew heavily on facts that I knew from TCS—such as the notion of an expander graph, basic properties of the Reed-Solomon code, and Valiant's construction of linear-size superconcentrators—which, while simple, were generally *not* known to the people who I was trying to reach.

Q. Anything else you would like to tell our readers?

Theoretical computer science, and CS more generally, is a lot younger than math or physics or economics or biology or philosophy. Even while CS more and more drives the world's economy, we still have to contend with critics who

wonder whether it's a real science at all, or whether it's just glorified software development (with, perhaps, TCS as merely an obscure branch of discrete math, promoted by its association with the computer revolution to undeserved importance). Occasionally I'm tempted to agree with them ... but then I reread Alan Turing. Well, also Boole and Babbage and Lovelace and von Neumann and Emil Post and Vannevar Bush, but Turing most of all.

Reading Turing gives me the same goosebumps that I get when I read Newton or Darwin or Einstein: there he is, deducing from abstract principles something absolutely fundamental about the shape of the real world, and in deducing it, *transforming* the real world. Like it or not, we in TCS are the heirs of Turing, keepers of his legacy. So whenever we're tempted to say that our job is just to publish more STOC/FOCS papers, shave off log factors, etc., while leaving the wider societal questions to others, we should remember that Turing faced a similar choice in 1939. He could've just kept publishing papers on computability theory. Instead, in the mere 15 years that remained to him, Turing chose to address himself first to the defeat of Nazi Germany's naval encryption, and then to the nature of consciousness and intelligence, the building and programming of the first electronic computers, and (when he needed some variety) developmental biology, statistics, and quantum mechanics. And I think the world is better for all of that, and we in TCS today can honor Turing by taking a similarly broad view of what our subject is.

THE DISTRIBUTED COMPUTING COLUMN

Seth Gilbert
National University of Singapore
seth.gilbert@comp.nus.edu.sg

In this issue of the distributed computing column, we return to the topic of blockchains. The column provides an overview of the Red Belly Blockchain, an innovative new blockchain developed in Australia (and named after the locally prevalent red-bellied black snake). Red Belly Blockchain has served as a testbed for several cutting edge approaches to designing a blockchain, including ideas from distributed computing, game theory, and formal verification. Some examples described in this paper include:

- Red Belly blockchain is (provably) *accountable*, providing a cryptographically secure "proof-of-fraud" when users act maliciously and disrupt the system.
- The blockchain contains several scalability innovations. For example, it takes an interesting approach to sharding (and "superblocks"), using a leaderless ("democratic") approach to dynamically create new shards as needed.
- The consensus component of the Red Belly blockchain has been formally verified with model checking tools, providing stronger guarantees of correctness.

The article also discusses new optimizations in handling smart contracts, challenges in governence, and trade-offs between mutability, accountability, and privacy. The article provides a window into how recent academic research translates into a large-scale, real-world blockchain design. Overall, it provides interesting insights in the challenges of integrating new ideas into blockchain design. Enjoy this new distributed computing column!

BEATCS no 137

Redbelly Blockchain: a Combination of Recent Advances*

Redbelly Network

Abstract

Redbelly Blockchain builds upon recent scientific advances in the context of distributed computing, game theory and formal verification to apply blockchains to the real world. In this paper, we present how Redbelly Blockchain combines these results to remedy vulnerabilities that affect modern blockchains. In particular, Redbelly Blockchain offers accountability by generating a *Proof-of-Fraud*, an undeniable proof of misbehavior, automatically. The architecture of Redbelly Blockchain is decoupled into a consensus component and a language virtual machine component, called SEVM, to achieve resilience optimality against failures and attacks. For greater security, its consensus component does not assume pure synchrony, is formally verified with model checking and solves the consensus problem deterministically. To run a large ecosystem of dApps efficiently, the SEVM supports Solidity bytecode without the unnecessary redundant validations of traditional designs. Redbelly Blockchain is dynamic as it features a built-in reconfiguration smart contract triggered by a representative governance. Finally, it is designed for mobile devices to interact securely without downloading the blockchain.

1 Distributed Ledger Architecture

The blockchain service is provided by blockchain *nodes*, or machines, offering consensus execution and storage service. Redbelly Blockchain is *open* in that any node has the possibility of providing the service, however, Redbelly Blockchain also aims at not wasting node resources by offering too many redundant services. Hence, instead of incentivising all nodes to execute the same tasks, Redbelly Blockchain allows different nodes to offer different services at different times. To eventually provide the service, nodes must first authenticate themselves, just like nodes must first offer a proof-of-work in classic blockchains [27] to produce valid blocks. One can thus compare the absence of proof-of-fraud in Redbelly Blockchain to the creation of proof-of-work, but without the carbon footprint associated with resolving cryptopuzzles.

More precisely, the architecture of Redbelly Blockchain is decoupled in different components to achieve resilience optimality. It is known that consensus cannot be solved in the general model, when nodes communicate over the Internet and in the presence of arbitrary (byzantine) failures, without $n > 3f$ nodes [26], where f is the number of byzantine nodes. Interestingly, however, this threshold does not apply to provide secure data retrieval: to achieve data integrity and tamper-proofness one simply needs $n > 2f$: despite f byzantine nodes, one is guaranteed to identify the correct copy as the only copy received from $f + 1$ distinct nodes. Hence, decoupling storage/SEVM nodes from consensus nodes allows to better tune the resource provisioning necessary to achieve security.

Figure 1 presents the architecture of Redbelly Blockchain. The SEVM component is web3 compliant, executes *decentralised applications (dApps)* and reliably stores the blockchain. The web3 compliance comes from its `web3.js` server that accepts valid requests to transfer assets, upload or

*Contact author: Vincent Gramoli `vincent.gramoli@redbelly.network`

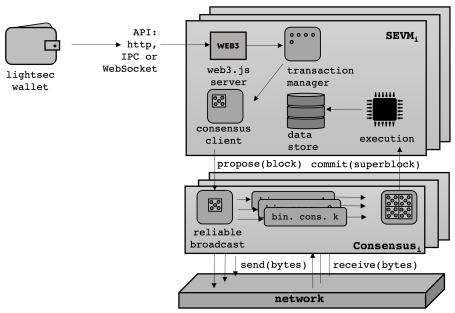


Figure 1: Redbelly Blockchain architecture and the life cycle of a transaction

execute a smart contract. Redbelly Blockchain supports smart contracts written in Solidity bytecode to facilitate the integration of the most used dApps. The transaction manager batches these transactions into blocks that are proposed to the consensus service. The consensus groups as many proposed blocks as possible into a committed *superblock* before the SEVM validates its requests, executes them and reliably stores the results.

The consensus component (§2) features the Democratic Byzantine Fault Tolerant (DBFT) consensus algorithm [11] that solves consensus deterministically without the need to assume pure communication *synchrony* or that there exists a known upper bound on the message delays [14]. (Instead, we assume that this bound is unknown, a property referred to as partial synchrony.) Note that this is appealing to cope with network attacks that could otherwise lead to double spending [28, 16]. To avoid network bottlenecks, DBFT starts with a leaderless all-to-all reliable broadcast where each consensus node shares its proposal with the rest of the system (§2.2). Then DBFT spawns as many binary consensus instances as proposals to decide whether each of these proposals gets included in the decided superblock. As the binary consensus is probably the most elaborate and sensitive part of Redbelly Blockchain it has been formally verified with model checking (§2.1). The decided superblock is then sent back to the SEVM for execution and storage.

Redbelly Blockchain supports a large ecosystem of dApps by offering the *Scalable EVM (SEVM)* (§5). Like for the Ethereum Virtual Machine (EVM), the SEVM allows Redbelly Blockchain to run like a state machine replication whose commands are expressed in a Turing complete programming language. In contrast with Ethereum [36] though, Redbelly Blockchain leverages this capability by utilizing smart contract output as an input to itself. Since the smart contract execution is deterministic and agreed upon by all nodes, it allows Redbelly Blockchain to upgrade itself without hard-forking (§6.1), to change its governance and mitigate the risks of bribery (§6) and for a subset of nodes to spawn their own shard or close it on-demand at runtime (§7.2).

Last but not least, Redbelly Blockchain interfaces the real-world (§8). This is why it comes with a lightweight secure wallet, called *lightsec* (§8.1) and interacts with oracles (§8.2). In particular, the lightsec wallet differs from classic wallets in that it is both (i) lightweight as it can run in handheld devices without having to download a blockchain history, and (ii) secure as it can retrieve correct information, hence its name. The transaction fees are maintained low thanks to the high capacity throughput of Redbelly Blockchain. The oracle offers the necessary identification mechanism to let real users govern Redbelly Blockchain and inform the blockchain about the success of traditional payments.

1.1 Blockchain Abstract Representation

The distributed architecture described above implements a blockchain, a simple *linked list* abstraction chaining blocks, one block to another, each containing a sequence of cryptographically signed transactions. We call it a linked list (and not a directed acyclic graph) as it does not have forks: consensus upon a block at a given index is reached before this block gets appended. We say that a transaction is *final* as soon as it is included in a block of Redbelly Blockchain.

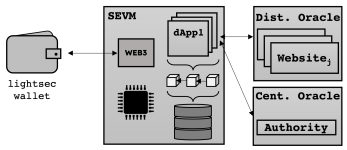


Figure 2: Redbelly Blockchain interactions with oracles

Figure 2 depicts the blockchain abstraction as maintained in the reliable storage of a single SEVM node. Thanks to the consensus protocol depicted in the distributed architecture (Figure 1), among two chains of blocks maintained locally by two honest SEVM nodes, either one is guaranteed to be the prefix of the other or the two are guaranteed to be identical. This property will be listed as a requirement of the blockchain problem (Def. 3). As all SEVM nodes end up having the same blocks anyway, we simply focus here on a single SEVM node, its interaction with oracles and the wallet for simplicity. The dApps are stored in the blockchain (either because these are built-in dApps or because some client uploaded them) and their functions can be invoked remotely by the clients. The dApps can gather real-world information from distributed (Dist.) or centralized (Cent.) oracles as we will discuss in §8.2.

2 Scalable and Verified Consensus

The byzantine consensus problem [26] is for a set of nodes to agree on a unique value despite arbitrary behaving nodes, called *byzantine*. This problem must be solved to design a secure blockchain that guarantees the uniqueness of the block to be appended at its next available index [1].

Redbelly Blockchain features the Democratic Byzantine Fault Tolerance (DBFT) consensus algorithm [11]. By contrast with most consensus algorithms, it is democratic in that it does not elect a leader node that tries to impose its value to the rest of the nodes, which would otherwise limit scalability as described in §2.2. In addition, this consensus algorithm is probably the first blockchain consensus algorithm to be formally verified with model checking [4] as we explain in §2.1, which reduces drastically the risks of human errors, that are otherwise common [33]. Finally, it does not assume that the bound on the delay of messages is known (it only assumes partial synchrony but not synchrony [14]) and it is resilient optimal as it tolerates $f < n/3$ arbitrary (or byzantine) failures.

The consensus problem is generally defined as the problem for *honest* (non-byzantine) nodes of ensuring the conjunction of three properties despite the presence of byzantine nodes.

Definition 1 (Byzantine Consensus)**.** *Assuming that each honest (non-byzantine) node proposes a value, the* Byzantine Consensus *(BC) problem is for each of them to decide on a value in such a way that all the following properties are satisfied:*

- *BC-Termination: every honest node eventually decides a value;*
- *BC-Agreement: no two honest nodes decide on different values;*

- *BC-Validity: the value decided is one of the proposed values.*

To understand the importance of solving consensus to design a blockchain, consider that the blockchain is in its initial state consisting of a genesis block, at index 0. Let an attacker, say Mallory, issue two conflicting transactions, tx_1 where Mallory transfers all her coins to Alice and tx_2 where Mallory transfers all her coins to Bob. Let us consider what can happen if the BC-Agreement property is violated in that distinct nodes of the blockchain believe they can append distinct blocks, one containing tx_1 and the other containing tx_2, at index 1 of the blockchain. This can lead to a *double spending*, where the same coins are spent twice. By contrast, when consensus is reached then all nodes agree on a unique block to be appended and it is simple for each individual node to go through this block and executes the transactions it contains one-by-one unless a transaction conflicts (e.g., lack sufficient funds to execute).

Algorithm 1 Binary consensus algorithm at replica p_i

```
1:  bin-propose(val):                                   ▷ binary consensus at p_i with val ∈ {0, 1}
2:    loop:                                             ▷ loop that starts with round r = 1
3:      (bv-broadcast(EST, r, val) → cvals)             ▷ reliable bcast that is ommitted in some optimization
4:      start-timer(r)                                  ▷ timeout increases with rounds
5:      if i = r mod n then                             ▷ coordinator rebroadcasts
6:        wait until (cvals = {w})                      ▷ cvals stores delivered values
7:        broadcast(COORD, r, w) → c                    ▷ coordinator broadcasts
8:      wait until (cvals ≠ ∅ ∧ timer expired)          ▷ wait enough time
9:      if c ∈ cvals then e ← {c} else e ← cvals        ▷ prioritize coord value
10:     broadcast(AUX, r, e) → bvals                    ▷ broadcast these values
11:     wait until ∃s ⊆ bvals where the two following conditions hold:   ▷ wait to deliver more values until
12:       • s contains contents received from at least n − t distinct nodes   ▷ sufficiently many copies are delivered
13:       • ∀v ∈ s, v ∈ cvals                           ▷ and every value in s is in cvals
14:     if s = {v} then                                 ▷ if there is only one value in s
15:       val ← v                                       ▷ adopt this singleton value
16:       if v = (r mod 2) and not decided yet then decide(v)   ▷ decide only once
17:     else val ← (r mod 2)                            ▷ otherwise, adopt the current parity bit
18:     if decided in round r − 2 then exit()           ▷ help others in two last rounds
19:     r ← r + 1                                       ▷ increment the round number
```

2.1 Formal Verification

As the problem of consensus is particularly difficult to solve, and human errors are frequent [33], it is important to check mathematically the properties of a security system, a process we call formal verification. We formally verified the binary consensus at the heart of DBFT (Alg. 1) using a model checker [4]. This binary consensus algorithm is represented as the bin.cons.1, ..., bin.cons.k blocks on Figure 1 because there is one instance of this binary consensus algorithm to decide whether each proposed block should be included in the committed superblock. All honest participants pass their input value *val* to the loop of Alg. 1 where they exchange values and refine their estimate until they decide a value (line 16). Because verification of liveness properties consists of assuming fairness and showing that a property holds in every fair execution, we relax the partial synchrony assumption [14] of DBFT and replace it by a fairness assumption. Relaxing partial synchrony allows to simplify the pseudocode by ignoring timers (lines 4 and 8 of Alg. 1) and the weak coordinator steps (lines 5–7 of Alg. 1). The fairness assumption states that in any infinite sequence of iterations of the protocol loop, there exists one iteration where, at all honest nodes, a binary value broadcast (or bv-broadcast) instances deliver the same bit first. Note that this fairness could be violated if byzantine nodes were controlling the network, but no solutions to the consensus problem would exist in this case anyway [6].

Algorithm	Verification time
bv-broadcast	48.87 s
Naive consensus	>3 days
Consensus	22.54 s

Table 1: Although none of the properties of the naive blockchain consensus could be verified within a day of execution of the model checker, it takes about ~4 s to verify each property on the simplified automaton of the blockchain consensus. Overall it takes less than 70 seconds to verify both the binary value broadcast and the blockchain consensus protocols.

To formally verify the consensus algorithm we holistically verified that the model of the pseudocode verifies the properties of Def. 1, expressed in linear temporal logic (LTL), for any system size n and fault tolerance $f < n/3$ as detailed elsewhere [3]. To this end, we rely on the parameterized model checker, ByMC [25], convert the pseudocode of the consensus algorithm into a threshold automaton, and deployed it on a Message Passing Interface (MPI) cluster of 4 AMD Opteron 6276 16-core CPU with 64 cores at 2300 MHz and 64 GB of memory. As the naive threshold automaton (Naive consensus) is too large to be formally verified within 3 days of execution on our MPI cluster as indicated in Table 1, we first verified the properties of the binary value broadcast (bv-broadcast) primitive (line 3 of Alg. 1) using ByMC before simplifying the naive threshold automaton using the formally proven properties of the bv-broadcast primitive. Both the safety and liveness properties of the resulting threshold automata (bv-broadcast and Consensus) could be verified in 1 minute and 11 seconds (cf. Table 1).

2.2 Bypassing the Leader Bottleneck

The name Democratic BFT consensus algorithm stems from the fact that this algorithm does not need a leader that tries to impose its block to the rest of the system. Instead, during an execution of DBFT, every node can propose its own block to the consensus and the decided, so called, superblock (cf. §3.2) can include any of the proposed blocks. One advantage is that, in DBFT, there cannot be a misbehaving leader that prevents the convergence towards agreement. In particular, there can be as many different weak coordinators (line 5 of Alg. 1) as there are concurrent binary consensus instances, and even a single weak coordinator cannot prevent for sure the convergence of the binary consensus to which it participates. As opposed to the nodes of traditional blockchains that "compete" to append their own block, the nodes of Redbelly Blockchain "collaborate" to append a combined superblock.

Another advantage of this collaboration is the scalability induced by having every participant exchanging their proposals in parallel over a wide area network. Consider, as an example, that we want to append a new block (or superblock) of b bits with a blockchain system with a traditional leader-based consensus algorithm running on n nodes that have limited bandwidth resources. In particular, each node i is equipped with a download capacity of d_i and an upload capacity of u_i, both expressed in bits per unit of time. Without loss of generality, let the leader be node 1 with download and upload capacities d_1 and u_1, respectively. The time τ it will take the leader to send the block to all nodes (we consider that the leader sends to itself for simplicity as it does not alter our conclusion) is the maximum between these two:

- The time for the leader to upload $n \cdot b$ bits, which is $\frac{n \cdot b}{u_1}$ units of time.
- The time to download b bits for the node that has the lowest download rate among all other nodes, which is $\frac{b}{min(d_i : 1 \leq i \leq n)}$ units of time.

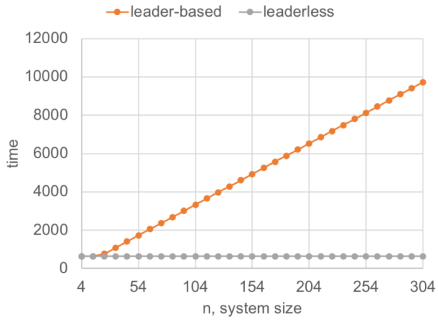


Figure 3: The time it takes for propagating a block in a leader-based consensus algorithm typically increases with the number n of nodes whereas the time to propagate a block in a leaderless consensus algorithm can be independent of n

Figure 3 depicts the time to propagate a block in both our simplistic leaderless and leader-based consensus algorithms as a function of n. The time needed for the leader to propagate the block to all nodes is $\tau = max\left(\frac{n \cdot b}{u_1}, \frac{b}{min(d_i : 1 \leq i \leq n)}\right)$ and we can conclude that this time is $\Omega(n)$ units of time. As n increases, we can see that the time it takes for the leader-based consensus algorithm to propagate the leader block increases with the number of nodes in the system. By contrast, the time it takes if the algorithm is leaderless to propagate a block of b bits to all the nodes is the maximum between the time for the node with the lowest upload rate to upload b/n bits, which is $\frac{b/n}{min(u_i : 1 \leq i \leq n)}$, and the time for the node with the lowest download rate to download b bits, which is $\frac{b}{min(d_i : 1 \leq i \leq n)}$. Provided that the difference in download rates among n nodes is independent of n, the maximum is thus a constant independent of n because each node simply needs to propagate b/n bits. This explains in part why DBFT helps Redbelly Blockchain scale [12].

3 Block Structure

As mentioned previously, the structure of the Redbelly Blockchain is a linked list, hence this can be viewed as a "non-forkable" chain of superblocks.

3.1 A non-forkable chain

As opposed to classic blockchains, Redbelly Blockchain does not fork to mitigate the risks of double spending. The main distinction with classic blockchains is that Redbelly Blockchain solves consensus first, before appending the unique agreed upon superblock as we already explained (§1.1). This guarantees a total order on the blocks that allows anyone to retrieve the current state of accounts by consulting the latest block: we do not talk about "confirmation" as a transaction is either pending or committed. This is in contrast with classic blockchains where an appended block must be sufficiently confirmed (or followed by sufficiently many blocks) for the probability of its transactions aborting to be sufficiently low.

Even during network attacks that can delay the message propagation [15], in common executions, two conflicting transactions cannot be inserted into two branches. We refer to these "common executions", as we will detail in §4, as executions in which an overwhelming number $f \geq n/3$ of

faults do not occur at the same time. In the scenario where conflicting transactions are included in the same block, then the first of these transactions will get executed while the second will not, as the conflict will be detected locally by the SEVM node. This tolerance to network partition can be seen as the result of favoring consistency over availability given that not both consistency and availability can be achieved in this case [21]. The key motivations for favoring consistency is for Redbelly Blockchain to support secure applications: no transactions that appeared committed can later be aborted.

In §4, we will explain how to cope with an overwhelming number of faults. In §7.2, we will explain how we can extend Redbelly Blockchain to spawn additional chains to complement the mainchain, however, none of these chains fork because they inherit the Redbelly Blockchain design.

3.2 Superblocks to scale throughput

Appending superblocks to the blockchain allows us to increase the performance (i.e., throughput) as the number of nodes running the consensus grows. This optimization originally proposed in [22], consists of resolving a different notion of consensus, called the Set Byzantine Consensus [12] as defined below.

Definition 2 (Set Byzantine Consensus)**.** *Assuming that each honest node proposes a set of transactions, the* Set Byzantine Consensus *(SBC) problem is for each of them to decide on a set in such a way that the following properties are satisfied:*

- *SBC-Termination: every honest node eventually decides a set of transactions;*
- *SBC-Agreement: no two honest nodes decide on different sets of transactions;*
- *SBC-Validity: a decided set of transactions is a non-conflicting set of valid transactions taken from the union of the proposed sets;*
- *SBC-Nontriviality: if all nodes are honest and propose a common valid non-conflicting set of transactions, then this set is the decided set.*

Thanks to this new definition, Redbelly Blockchain does not need to decide one block maximum in each iteration of the consensus (as classic blockchains do). Instead, Redbelly Blockchain decides a number of blocks that can grow linearly with the number of nodes in the system. This is key to the scalability of the consensus protocol. There are two important remarks regarding this definition. First, note that it refers to a decided set, although the transactions should be executed in the same order at every node. This can easily be ensured by executing transactions in the order of their hash or nonce. Second, the superblock optimization differs from batching more proposals at the leader, as batching could exacerbate the bottleneck effect of the leader (§2.2).

4 Accountability with PoF

Redbelly Blockchain applies *accountability* [23], the ability to make distributed participants responsible for their actions, to the partially synchronous setting [9]. To this end, it generates undeniable proofs-of-fraud (PoFs) as indicated in §4.2. By requiring participants to deposit assets prior to participating, we can exploit proofs-of-fraud to slash malicious players and compensate the victims, in the unlucky case of an overwhelming majority of participants as we will explain in §4.1.

PoF can be compared to PoW or PoS to cope with Sybil attacks: a block is considered valid in Redbelly Blockchain only if it was produced by a participant that would leave some undeniable PoF in case it tries to attack the system. This is why a valid block is one that is produced by a user who

provided the necessary information required by Redbelly Blockchain. This information is used to identify the user uniquely, which prevents a malicious attacker from impersonating other users to conduct a Sybil attack.

4.1 Strengthening fault tolerance

Accountability allows us to reward only the good behaviors that contribute to the system. Such good behaviors include staking (providing liquidity to the system for a period of time), using storage resources to keep track of the blockchain history, or exploiting network and CPU resources in order to contribute in reaching a consensus. Redbelly Blockchain will require consensus participants to deposit some stake before they start executing the consensus. They will gain a reward based on their contribution to the consensus. Our reward is equally divided among the consensus participants for their contributions but consensus participants will change over time as we will explain in §6.1. If a consensus participant misbehaves, then it will not receive its reward and will be excluded from the set of consensus participants, preventing it from being rewarded in the future.

Accountability is particularly effective in "uncommon" situations, where a coalition of malicious users is of size f sufficiently large to lead honest consensus nodes to a disagreement. As consensus is impossible in the general setting when $n \leq 3f$ where n is the number of consensus nodes [26], we know that this can happen. Fortunately, even when $f \geq n/3$ and as long as $f < 2n/3$ and less than $n/3$ nodes are inactive forever, Redbelly Blockchain can recover. When the disagreement occurs we can upper-bound the number a of branches given the number f of malicious participants, as was shown in [29]. A recent work has even demonstrated that not more than $2n/3$ honest participants is necessary to solve consensus when considering that some participants are rational [30]. For this reason, we can also lower-bound a deposit that consensus nodes need to escrow in order to reimburse any fooled users.

In particular, we require each consensus node to deposit some amount d during the time it participates to the consensus protocol. This deposit amount depends on various parameters. In particular, our recent result [29] expresses the ratio b of the deposit over the value G of funds that is being stolen. Consider a colluding majority of size $n/3 \leq f < 5n/9$, a probability of attack success $\rho = 0.5$, deposits held for $m = 10$ blocks and $G = \$1M$ manipulated funds. As $f < 5n/9$, we have $a \leq 3$ branches, thus $b = 1/500$ is sufficient. As $D = G/500 = 2,000$, for $n = 100$, each node needs to deposit $3bG/n = \$60$ for Redbelly Blockchain to recover.

4.2 Proof-of-Fraud

Our solution to the accountability problem is to build undeniable proofs-of-fraud at runtime. More specifically, our implementation enforces all nodes to sign key messages: a honest node will simply ignore key messages that are not properly signed by their senders. These 'key' messages are those that can influence the decision of other participants during the execution of any binary consensus protocol (Alg. 1) or the reliable broadcast (§1) that precedes these binary consensus executions [29].

In particular, it was shown that the only way for honest nodes to disagree while executing DBFT is for a coalition of malicious nodes to hack one of the broadcast primitives so as to equivocate, by sending different messages to different honest nodes [9]. Provided that these messages are signed, upon reception of these messages, honest nodes simply have to cross-check their received messages to detect a misbehavior. An undeniable proof-of-fraud is thus built by a honest node using the concatenation of two equivocating messages from the same sender. These proofs-of-fraud are rapidly communicated to other honest nodes to stop rewarding malicious nodes.

Note that the original accountable consensus technique, called Polygraph [9], is an extension of our consensus algorithm DBFT [11], which already offered scalable results in geo-distributed experiments [29]. Since then, a more generic transformation has been proposed [10]. It only requires

an additional round of communication involving an additive $O(n^2)$ communication complexity when threshold signatures can be used.

5 Language Virtual Machine

In this section, we present the Scalable EVM (SEVM), an optimized way of validating transactions and executing smart contracts.

5.1 Smart contracts

Redbelly Blockchain is compatible with a large ecosystem of DApps by offering the possibility to execute the same bytecode supported by the Ethereum Virtual Machine (EVM) [36]. The motivation for not developing a new DApp language is simple: the shortage in resources and the high demand of programmers skilled in blockchain raised the cost of programming decentalized finance, which makes it difficult to grow a new ecosystem of DApps.

5.2 Validation reduction

We built upon the EVM to develop what is called the *SEVM*, standing for the Scalable EVM, as detailed in [32]. It inherits the gas mechanism of Ethereum, mitigating denial-of-service attacks, but differs from the EVM to achieve 10,000 TPS on 100 nodes with 33 failures tolerated. With our consensus protocol (§2), the overhead is no longer the consensus but the EVM execution when trying to execute smart contract functions. The key aspect to reduce existing smart-contract blockchain overheads is thus to reduce the unnecessary validation overhead common to existing blockchains (e.g., Ethereum, Libra/Diem) where all validators validate each transaction twice (upon transaction reception and upon block reception), hence validating globally each transaction $2n$ times, where n is the number of SEVM nodes. Redbelly Blockchain simply needs to validate each transaction $n + 1/n$ times, which tends to halving the validation overhead as the system size grows towards infinity, $n \to \infty$. The security is not hampered because all nodes, upon reception of the decided block, validate every transaction, anyway.

Go Ethereum, or `geth` for short, is probably the mostly deployed EVM implementation. In order to check that a request (or transaction) is valid, all of the `geth` *servers* (i.e., miners) must validate each transaction eagerly and lazily, hence we distinguish the two following validations:

- **Eager validation:** This validation occurs upon reception of a new client transaction to check various parmeters of this transaction (gas, balance, signature, size). If the transaction is valid, it is propagated to other servers.

- **Lazy validation:** This validation occurs before transactions are executed in a decided block and simply checks the nonce and the gas. The lazy validation is thus less time consuming in `geth` than the eager validation.

Note that this is an overconservative strategy because each to-be-executed transaction of `geth` is validated twice by each validator/miner. In particular, there is no need for all validators to validate all transactions twice. In Redbelly Blockchain, only a constant number of nodes execute the eager validation, but without re-propagating the transaction to all servers. If the transaction is decided by the consensus algorithm, anyway, all servers will execute the lazy validation before executing the transaction. Note that not forwarding the request still ensures the best effort property of classic blockchains, namely that a transaction needs to be received by a honest node to be eventually

committed. The advantage is that it reduces the number of validations per nodes from $V^n_{eth} = 2$ in Ethereum to $V^n_{rbb} = 1 + \frac{k}{n}$ in Redbelly Blockchain. At large scale, when $n \to \infty$, we thus obtain:

$$\begin{cases} \lim_{n\to\infty} V^n_{rbb} &= \lim_{n\to\infty} \left(1 + \frac{k}{n}\right) = 1, \\ \lim_{n\to\infty} V^n_{eth} &= 2. \end{cases}$$

Hence, the number of validations per node in Redbelly Blockchain tends to become half the number of validations per node in existing blockchains (e.g., Ethereum, Libra/Diem [18]) as the system enlarges.

6 Governance

In this section, we explain the internals of the Redbelly Blockchain governance that decides upon variations in the protocol, the incentives or the governance membership. The governance is handled by the consensus nodes and the SEVM nodes. Note that each of these nodes may reside on different machines and be administered by different entities.

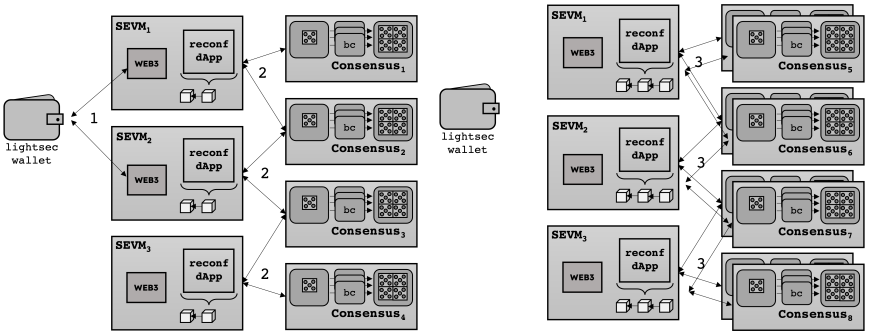
6.1 Reconfiguration to mitigate bribery

The nodes that govern are called *governors*. They must change from time to time to reduce the risks of bribery attacks, under the assumption of a slowly-adaptive adversary (so that it takes more time to bribe more nodes). Just like in other blockchains, nodes are incentivized to become governors by obtaining a reward for offering the blockchain service (e.g., execution, consensus, storage). Before governing, a node must first express its interest in governing and satisfy a series of requirements (personal information, resource allocation). If these requirements are met, the node becomes a *candidate*.

Redbelly Blockchain features a novel smart contract based reconfiguration process to change the governance every k blocks. The difficulty to reconfigure distributed systems is that all nodes must agree on the new configuration, a coordination process that often requires multiple consecutive consensus executions: one to decide to add new nodes to the system, another to discard the old nodes. Thanks to Redbelly Blockchain each of these consensus are reached simply through a *built-in smart contract* function invocation. As each execution of these invocations is deterministic and replicated, we are guaranteed that all honest nodes will be informed of this reconfiguration. This type of smart contracts is called "built-in", because they are already part of Redbelly Blockchain at the time it is started: all users of Redbelly Blockchain can thus observe the power delegated to the governance at the time they start using Redbelly Blockchain.

Figure 4 depicts the smart contract based reconfiguration process where SEVM nodes maintain a copy of the reconfiguration smart contract. For the sake of simplicity, it illustrates how to replace consensus nodes, but this reconfiguration can be used to replace SEVM nodes as well as changing the current version of the software. On the left hand side (Fig.4(a)), the client sends a request that invokes a function of the reconfiguration smart contract. This request is validated and passed onto the current consensus nodes ($consensus_1$, $consensus_2$, $consensus_3$ and $consensus_4$). The consensus nodes agree to encapsulate this request in the next superblock and pass this superblock to the SEVM for storage and execution. Upon execution (Fig.4(b)), the reconfiguration smart contract function replaces the current consensus nodes by new ones ($consensus_5$, $consensus_6$, $consensus_7$ and $consensus_8$). Hence the governors responsible of running the consensus have been replaced.

The software upgrade also relies on a built-in smart contract function, however, it takes, as arguments, the current version number, the new version code (e.g., in the form of a URL and its RSA256 hash representation) and the index *idx* of the chain at which it should start being used.



(a) The client sends a function invocation that gets agreed upon by the consensus nodes

(b) The function is executed on each SEVM node, hence replacing the current consensus nodes by new ones

Figure 4: Smart contract based reconfiguration

When $2n/3 + 1$ of the governors have invoked the function provided that this happens before the chain reaches index *idx*, then the new version is being downloaded, its hash is checked and in case of success, the version is being used when the chain reaches *idx*. (No other version besides the current one can be used until index *idx* is reached.)

6.2 Elections with non-dictatorship

In order to avoid that an attacker acts as a dictator by imposing its chosen governance to the rest of the system, we build upon results from the social choice theory. One way of selecting new governors, is to let existing governors proportionally elect the next set of governors. Black [5] was the first to define proportionality or that elected members represent "all shades of political opinion" of a society.

Dummett [13] introduced *fully proportional representation* to account for ordinal ballots, containing multiple preferences: given a set of n voters aiming at electing a committee of k governors, if there exist $0 < \ell \leq k$ and a group of $\ell \cdot q_H$ voters who all rank the same ℓ candidates at the top of their preference orders, then these ℓ candidates are all elected. However, it builds upon Hare's quota q_H, which is vulnerable to strategic voting, whereby a majority of voters can elect a minority of seats [24]. This problem was solved with the introduction of Droop's quota q_D as the smallest quota such that no more candidates can be elected than there are seats to fill [34].

Woodall [37] replaces Hare's quota with Droop's quota $q = \lfloor \frac{n}{k+1} \rfloor$ and defines the *Droop proportionality criterion* as a variant of the fully proportional representation property: if for some whole numbers j and s satisfying $0 < j \leq s$, more than $j \cdot q_D$ of voters put the same s candidates (not necessarily in the same order) at the top of their preference list, then at least j of those s candidates should be elected.

This desirable "proportionality" property can be achieved using the *Single Transferable Vote (STV)* algorithm with Hare's quota $q_H = \frac{n}{k}$. In STV, candidates are added one by one to the winning committee and removed from the ballots if they obtain a quota q of votes. STV is used to elect the Australian senate and is known to ensure fully proportional representation. Unfortunately, this protocol is synchronous [14] in that its quotas generally rely on the number of votes n received within a maximum voting period. If some of these n voters are byzantine and do not respond, then the protocol could not terminate without synchrony.

As one cannot predict the time it will take to deliver any message without synchrony, one cannot distinguish a slow voter from a byzantine one. Considering n as the number of governors or potential

voters among which up to f can be bribed or byzantine, our protocol can only wait for at most $n-f$ votes to progress without assuming synchrony. Waiting for $n-f$ votes prevents us from guaranteeing that the aforementioned quotas can be reached. We thus define a new quota called the *byzantine quota* $q_B = \lfloor \frac{n-f}{k+1} \rfloor$ such that $f < n/3$ and reduce, to $n-f$, the number of needed votes to start the election. Of course, up to f of these $n-f$ ballots may be cast by byzantine nodes, however, Redbelly Blockchain guarantees that no adversary controlling up to f byzantine nodes can act as a dictator in always imposing its decision.

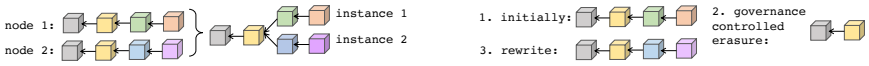
Based on q_B, we propose a smart contract election that expects $n-f$ votes to be cast to run a byzantine fault tolerant version of STV that satisfies proportionality and non-dictatorship without assuming synchrony.

7 Mutability, Auditability, Privacy

In this section, we discuss tradeoffs between auditability and privacy on the one hand, and mutability and immutability on the other hand (§7.1). We also present the two key techniques to offer privacy. First, Redbelly Blockchain offers the possibility to a subset of users to spawn a new shard as a dynamic variant (§7.2) of the Eth2 topology [19]: by invoking a *mainchain* built-in smart contract, some of the users can spawn a new *shardchain*. This allows users to perform transactions that are not directly visible from the mainchain or from other shards and without inducing a negative impact on performance. In addition, to enforce a stronger form of privacy, we will exploit the privatization of fungible and non-fungible tokens (§7.3).

7.1 Mutability with a Built-in Contract

On the one hand, immutability is a particularly appealing property for maintaining a distributed ledger. If transactions can be erased, then the data integrity is at risk. On the other hand, blockchain can be used for storing sensitive data whose immutability is questionable. The General Data Protection Regulation (GDPR) imposed in Europe since 2018 permits personal data to be rectified, withdrawn of permission, and erased. Similarly, the California Consumer Privacy Act (CCPA) and the United States Fair Credit Reporting Act (FCRA) enforces the possibility to remove objectionable data from the blockchain. Even mainstream blockchains are mutable: after the DAO hack that affected Ethereum, the history of transactions was replaced through a hard fork. Unfortunately, this created confusion and led to the use of two blockchains, Ethereum Classic and Ethereum. This is why, Redbelly Blockchain guarantees immutability by preventing any adversary from tampering with data but offers a built-in smart contract rewrite function to the governance.



(a) Rewriting in a classic blockchain creates new instances

(b) Rewriting in Redbelly Blockchain does not fork

Figure 5: To avoid forks while offering mutability, Redbelly Blockchain enforces that any rework of the history is controlled by the governance

Figure 5 compares a rewrite in a classic blockchain (Fig 5(a)) to a rewrite in Redbelly Blockchain (Fig. 5(b)). In classic blockchains, miners may legitimately decide to run the same software version they have been running before, whereas other users may follow the recommendation to upgrade their software. This typically creates multiple instances sharing the same genesis block but followed

by diverging transaction histories. The original instance is usually called "classic" so that the new version retains the original name of the blockchain, which can add to the user's confusion.

By contrast and as depicted in Figure 5(b), in Redbelly Blockchain, all users initially agree to run a reconfigurable blockchain, as we already explained in Section 6.1. This guarantees that all users are ready to upgrade to a new version if the governance majoritarily vote to do so. Typically, the governors cast their vote by passing a rewrite parameter to a built-in contract, a smart contract that was already present when the blockchain was launched, as defined in §6.1. This rewrite parameter is a pair $\langle state, start\text{-}index \rangle$ where the *state* is the hash of the global state of the blockchain, also called "state root" and the *start-index* is the index of the first block of the blockchain suffix to erase. If a quorum of $2f + 1$ governors vote for the same *start-index* within the time it takes to create k blocks (meaning that all their *state* parameters belong to at most k consecutive blocks, then the erasure of the suffix takes place at each honest SEVM node. From then on, all blocks get appended in place of the erased suffix.

7.2 Dynamic Sharding

Our blockchain design allows participants to operate individually in a dedicated shard or *shardchain* without presenting all their transactions to the rest of the system. This benefits privacy and performance by reducing the congestion on the default chain called the *mainchain*. It can also allow users to control the sovereignty of data by allocating the machines hosting these data in a specific jurisdiction. While this sharding may look similar to the beacon chain and the shard chains of Eth2 [19], it differs by being dynamic: one can adjust the number and size of shards at runtime as was recently proposed [31].

Redbelly Blockchain exploits smart contract outputs to configure itself at the lower level. Because our consensus protocol is both deterministic and formally verified (§2.1), it guarantees that all participants agree on a consistent total order on the smart contract upload, invocations and transactions issued to Redbelly Blockchain. As a result, and because smart contract functions are also deterministic, the execution of the sequences of commands at all honest participants results in the same outcome or state.

A built-in smart contract features a `spawnShard()` function that allows a set of participants to create a shard by providing the resources where to run the shard as well as depositing some assets that will be locked on the mainchain to be used in the shard. Similarly to the reconfiguration (§6.1), these participants will invoke the `spawnShard()` function with an index *idx* parameter and a number k of consecutive blocks, so that the shard will be spawned only if more than a single participant has invoked this function between the block at index *idx* and the block at index $idx + k$ of Redbelly Blockchain.

To close the shard, a `closeShard()` function will trigger the closing of a shard s when the blockchain depth reaches d only if a quorum of $\lfloor 2n/3 \rfloor + 1$ participants (or governors) of shard s have invoked this function within a certain range of indices *idx* to $idx + k < d$. To cope with non-termination due to windows of asynchrony, after failure to close a shard, the participants can simply retry with a longer offset k.

7.3 Stronger Forms of Privacy

Our solution will offer privacy of the transaction data (amount, asset, recipient) through the use of Zero-Knowledge Proof or Verifiable Secret Sharing. To this end, a privacy layer on top of Smart Redbelly that will consist of a shield smart contract, like the one used in Nightfall, will make the use of ERC20 (fungible) and ERC721 (non–fungible) tokens private. It allows the user to (i) create token commitments that are anonymous representations of the ECR20/721 tokens through the process of minting, (ii) to retrieve the token associated with a token commitment through the process of burning

and (iii) to transfer confidentially these commitments. This approach requires the user to create a proof off-chain and to store some private information needed to generate the proof locally. The user will then interact with the shield contract by sending their proof, which will then be verified. Upon successful verification, the token commitment will be stored in the commitment Merkle trie until it is spent, in which case it will be stored in a nullifier data structure.

8 Real World Interactions

In this section, we present the interactions between the wallet and the blockchain service and between the blockchain service and the oracle, and we explain how security is strengthened.

Redbelly Blockchain is dedicated to allow users to provide real-world services to one another. The key is to support the smart contracts already supported by Ethereum so as to minimise the efforts of porting existing contracts. We foresee the deployment of smart contracts encapsulating legal clauses. In order to tie these services to the real world, we will have to use oracles. As Redbelly Blockchain is secure, it is important that the oracle interactions do not introduce single point of failure vulnerabilities. To this end, Redbelly Blockchain will communicate with oracles whose implementation is distributed (tolerating isolated failures) or accountable (offering an insurance covering potential failures).

Accountability is instrumental in compensating the error committed by a centralized source. As an example a human error led Chainlink to report a price anomaly on their XAG/USD price feeds requesting gold price (XAU) instead of silver price (XAG) [7]. This was exploited by traders to generate ~US$36,000 in profit at the expense of Synthetix stackers. Distribution is instrumental in reducing such risks. For static information, the SEVM node requests $f + 1$ identical inputs from distributed sources to tolerate f failures. For dynamic information (e.g., stock value) or localised information (e.g., temperatures), the SEVM node extracts the median value among a group of $2f + 1$ distributed sources as we explain in §8.2.

8.1 Lightweight Secure Wallet

The lightsec wallet is both (i) lightweight in that it can run in handheld devices, and (ii) secure in that it can retrieve correct information. This is in contrast with traditional solutions where the device interacting with the blockchain must either trust the node it communicates with, which defeats the purpose of the blockchain; or download the blockchain itself, a task impossible for small devices that do not have sufficient storage space (recall that the Ethereum blockchain already exceeds 600 GB).

The lightsec wallet approach relies on the observation that to retrieve parts of the current state (e.g., the correct balance of a blockchain account), one simply needs to fetch $f + 1$ identical copies of this balance [32]. This is made possible by involving the participation of at most $2f + 1$ SEVM nodes that maintain a local copy of the blockchain and their current state or by requesting a threshold signature corresponding to such a quorum of nodes.

An interesting aspect of the lightsec client, is that it does not need to send its transaction to more than a single blockchain node to achieve the same liveness guarantee as Ethereum because it is anyway well-known that the blockchain cannot ensure that all transactions will be committed as we briefly mentioned in §5.2. This is why the blockchain problem is often defined with the liveness and uniformity properties [20, 8]. But since we require a blockchain to store only valid transactions, we also require to solve the additional validity property of [12] to solve this problem.

Definition 3 (The Blockchain Problem)**.** *The* blockchain problem *is to ensure that a distributed set of nodes maintain a sequence of transaction blocks such that the three following properties hold:*

- Liveness: *if an honest node receives a transaction, then this transaction will eventually be reliably stored in the block sequence of all honest nodes.*

- Uniformity: *the two chains of blocks maintained locally by two honest nodes are either identical or one is a prefix of the other.*

- Validity: *each block appended to the blockchain of each honest node is a set of valid transactions (non-conflicting well-formed transactions that are correctly signed by its issuer).*

Interestingly, the liveness property does not guarantee that a client transaction is included in the blockchain: if a wallet sends its transaction request exclusively to byzantine nodes then byzantine nodes may decide to ignore it. Hence, the lightsec wallet combined with Redbelly Blockchain guarantees this liveness property: it could be the case that a wallet has to send its transactions multiple times before it gets committed (just like in classic blockchains).

8.2 Oracles

An oracle is critical to feed major dApps with trustworthy off-chain information[1]. They can offer identification or payment services by indicating whether a bank payment is successful or whether the identity of a user has been verified.

Typically, a distributed oracle is a software running on some set of computers that gather real-world information from multiple websites in order to relay it to the blockchain (an oracle network [17] is an example of such a distribution). This information is typically used by dApps to trigger an action based on an external event. For example, an authentication system may accept some of the users based on its identity document. This acceptance is a piece of information from the real-world that an oracle must input to the blockchain for the dApp to authenticate the user. Typical examples include an API provider directly inputting this information to a smart contract [2] or offering an authentication proof to the end-user [35]. This input information can have such a large impact on the execution of the blockchain and the transfers of high value assets that it is important that it remains correct.

One cannot trust a single computer inputting this information, as the owner of the computer may easily get bribed by a bidder to steal the reward by pretending that their prediction was correct. This is why an oracle should either be accountable or distributed. One way to gather the information is by consulting online information on some online service. However, if the only service experiences a transient bug, then the blockchain assets are at risk. This is the reason why the distributed oracle (Dist. Oracle) will fetch the information from distinct sources (e.g., different websites) as depicted in Figure 2. To tolerate the failure of f online services announcing some real information, it is sufficient to gather $f + 1$ identical copies of the same information: this will guarantee that this information is correct. The worst case situation thus consists of contacting $2f + 1$ to retrieving the correct information.

Yet, there exist scenarios where the outcome is fluctuating: For example, a stock value on the New York Stock Exchange can change over time and no pair of sources could provide the exact same result due to message delays. In this case, the oracle will extract the median among $2f + 1$ requests to guarantee that no coalition of f byzantine machines can skew the outcome towards one end or the other. Another type of Oracle could be centralised and accountable (cf. Cent. Oracle in Figure 2) and under the control of an authority, like the SEC, that dictates how the regulation evolves over time and whose role is to guarantee that regulation is correctly followed at any time. In this case, we may consider the SEC information to be authoritative and render such a service accountable, removing the need to cross-check multiple sources.

[1] https://www.defipulse.com/.

9 Conclusion

Redbelly Blockchain is an innovative technology to interface the real world through a novel proof-of-fraud design. It ensures accountability of its participants and aims at complying with regulation. It builds upon recent research advances in the context of security [12], distributed computing [9], formal verification [4] and game theory [30]. Two of its key novelties is that it is deterministic, due to its consensus algorithm, and dynamic, due to its built-in smart contracts, that allow the governance to reconfigure it at runtime.

References

[1] Emmanuelle Anceaume, Antonella Del Pozzo, Romaric Ludinard, Maria Potop-Butucaru, and Sara Tucci-Piergiovanni. Blockchain abstract data type. In *The 31st ACM Symposium on Parallelism in Algorithms and Architectures*, page 349–358, 2019.

[2] Burak Benligiray, Saša Milić, and Heikki Vänttinen. API3: Decentralized APIs for Web 3.0. Accessed: 2022-04-28 - https://drive.google.com/file/d/1GzkLKc6DYxImgeDhoKLA4wHGlE0eGGgo/view.

[3] Nathalie Bertrand, Vincent Gramoli, Igor Konnov, Marijana Lazić, Pierre Tholoniat, and Josef Widder. Compositional verification of Byzantine consensus. Technical Report hal-03158911, HAL, June 2021.

[4] Nathalie Bertrand, Vincent Gramoli, Igor Konnov, Marijana Lazić, Pierre Tholoniat, and Josef Widder. Brief announcement: Holistic verification of blockchain consensus. In *Proceedings of the 41st ACM Symposium on Principles of Distributed Computing (PODC)*, 2022.

[5] Duncan Black. *The Theory of Committees and Elections*. Cambridge University Press, 1958.

[6] Zohir Bouzid, Achour Mostfaoui, and Michel Raynal. Minimal synchrony for Byzantine consensus. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, page 461–470, 2015.

[7] Chainlink. Improving and decentralizing chainlink's feature release and network upgrade process, 2020. Accessed: 2022-03-31, https://blog.chain.link/improving-and-decentralizing-chainlinks-feature-release-and-network-upgrade-process/.

[8] Benjamin Y. Chan and Elaine Shi. Streamlet: Textbook streamlined blockchains. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 1–11, 2020.

[9] Pierre Civit, Seth Gilbert, and Vincent Gramoli. Polygraph: Accountable Byzantine agreement. In *Proceedings of the 41st IEEE International Conference on Distributed Computing Systems (ICDCS)*, Jul 2021.

[10] Pierre Civit, Seth Gilbert, Vincent Gramoli, Rachid Guerraoui, and Jovan Komatovic. As easy as abc: Optimal (A)ccountable (B)yzantine (C)onsensus is easy! In *Proceedings of the 36th International Parallel and Distributed Processing Symposium (IPDPS)*, 2022.

[11] Tyler Crain, Vincent Gramoli, Mikel Larrea, and Michel Raynal. DBFT: Efficient leaderless Byzantine consensus and its applications to blockchains. In *Proceedings of the 17th IEEE International Symposium on Network Computing and Applications (NCA)*, 2018.

[12] Tyler Crain, Christopher Natoli, and Vincent Gramoli. Red belly: a secure, fair and scalable open blockchain. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (S&P)*, May 2021.

[13] Michael Dummett. *Voting Procedures*. Oxford University Press, 1984.

[14] Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *J. ACM*, 35(2):288–323, April 1988.

[15] Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. Impact of man-in-the-middle attacks on ethereum. In *2018 IEEE 37th Symposium on Reliable Distributed Systems (SRDS)*, pages 11–20, 2018.

[16] Parinya Ekparinya, Vincent Gramoli, and Guillaume Jourjon. The attack of the clones against proof-of-authority. In *27th Annual Network and Distributed System Security Symposium (NDSS)*, 2020. Presented at the Community Ethereum Development Conference in 2019.

[17] Steve Ellis, Ari Juels, and Sergey Nazarov. Chainlink: A decentralized oracle network, 2017. Accessed: 2022-04-28 - https://research.chain.link/whitepaper-v1.pdf.

[18] Amsden et al. The libra blockchain, 2020. Accessed on 2022-04-27, https://diem-developers-components.netlify.app/papers/the-diem-blockchain/2020-05-26.pdf.

[19] The eth2 upgrades. Accessed: 2022-03-28, https://ethereum.org/en/eth2/.

[20] Juan A. Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *34th Annu. Int. Conf. the Theory and Applications of Crypto. Techniques*, pages 281–310, 2015.

[21] Seth Gilbert and Nancy A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[22] Vincent Gramoli. The red belly blockchain: Bft is back but is it the same? In *Workshop on Blockchain Technology and Theory collocated with DISC*, 2017. Invited Talk.

[23] Andreas Haeberlen, Petr Kouznetsov, and Peter Druschel. PeerReview: Practical Accountability for Distributed Systems. *SOSP'07*, 2007.

[24] Jonathan Lundell & David Hill. To advance the understanding of preferential voting system - notes on the droop quota. *Voting matters*, 2007.

[25] Igor Konnov, Marijana Lazić, Helmut Veith, and Josef Widder. A short counterexample property for safety and liveness verification of fault-tolerant distributed algorithms. In *Symposium on Principles of Programming Languages (POPL)*, pages 719–734. ACM, 2017.

[26] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.

[27] Satoshi Nakamoto. Bitcoin: a peer-to-peer electronic cash system, 2008. Accessed: 2022-05-03 - https://bitcoin.org/bitcoin.pdf.

[28] Christopher Natoli and Vincent Gramoli. The balance attack or why forkable blockchains are ill-suited for consortium. In *47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017.

[29] Alejandro Ranchal-Pedrosa and Vincent Gramoli. Blockchain is dead, long live blockchain! accountable state machine replication for longlasting blockchain. Technical Report abs/2007.10541, arXiv, 2020.

[30] Alejandro Ranchal-Pedrosa and Vincent Gramoli. Trap: The bait of rational players to solve Byzantine consensus. In *Proceedings of the 17th ACM ASIA Conference on Computer and Communications Security (AsiaCCS)*, 2022.

[31] Deepal Tennakoon and Vincent Gramoli. Dynamic blockchain sharding. In *Proceedings of the 5th International Symposium on Foundations and Applications of Blockchain (FAB)*, volume 101. OASIcs, 2022.

[32] Deepal Tennakoon, Yiding Hua, and Vincent Gramoli. CollaChain: A BFT collaborative middleware for decentralized applications. Technical Report 2203.12323, arXiv, 2022.

[33] Pierre Tholoniat and Vincent Gramoli. Formally verifying blockchain Byzantine fault tolerance. In *The 6th Workshop on Formal Reasoning in Distributed Algorithms (FRIDA)*, 2019. Available at https://arxiv.org/pdf/1909.07453.pdf.

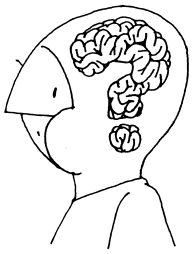
[34] Nicolaus Tideman. The single transferable vote. *Journal of Economic Perspectives*, 9(1):27–38, March 1995.

[35] Verite. Verifying verifiable credentials. Accessed: 2022-04-28 - https://docs.centre.io/verite/patterns/verification-flow.

[36] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger, 2015. Yellow paper.

[37] Douglas Woodall. Properties of preferential election rules. In *Voting Matters*, 1994.

News and Conference Reports



Report on ICALP 2021

48th EATCS International Colloquium on Automata, Languages and Programming

Anca Muscholl[1]

The *48th EATCS International Colloquium on Automata, Languages and Programming (**ICALP 2021**)*, the flagship conference and annual meeting of the *European Association for Theoretical Computer Science (EATCS)*, took place from the 13th to the 16th of July 2021. The event was held as online, hosted by the University of Glasgow.

The main conference was preceded by a series of workshops, which took place on July 11–12, 2021. The following workshops were held as satellite events of ICALP 2021:

- Flavours of Uncertainty in Verification, Planning and Optimization (FUNCTION),
- Verification of Session Types (VEST),
- Programming Research in Mainstream Languages (PRiML),
- Combinatorial Reconfiguration,
- Algorithmic Aspects of Temporal Graphs (AATG),
- Graph Width Parameters: from Structure to Algorithms (GWP),
- Formal Methods Education Online: Tips, Tricks & Tools.

The scientific programme of ICALP 2021 consisted of 3 unifying lectures and 3 invited lectures of the two tracks of ICALP, the presentation of 137 contributed papers (which were selected by the Program Committees out of 462 submissions) and award sessions with the lectures of the EATCS Award 2021 and the Presburger Award 2021 recipients.

[1]LaBRI, Université Bordeaux. Email: anca.muscholl@u-bordeaux.fr.

The online conference. The online conference was managed by the FATA research group of the School of Computing Science, University of Glasgow, led by Simon Gay. The organizers in Glasgow did their very best to make the online conference a lively experience and a scientific feast for all the participants.

ICALP 2021 used the Whova conference platform. Talks were pre-recorded, made available to registered participants two weeks before the conference, and archived on YouTube afterwards.

Each accepted paper had a 5-minute live presentation followed by a 5-minute live Q&A session. The live sessions were held daily (Tuesday – Friday, July 13–16) between 12:00 and 6pm CEST. They included plenary events for the Opening Ceremony, EATCS/ICALP Award sessions, EATCS General Assembly, and the six invited presentations.

The organizers succeeded to offer registration at a significantly reduced price compared to the normal cost. While typical ICALP registration costs are between €400 and €500, this year there were two types of event participation: *author registration* for €175 (€140 for students), and *non-author registration* for €50 (€30 for students). The author registration contributed to the fixed costs of the conference and to the costs of the LIPIcs proceedings.

On behalf of the entire community, we warmly thank the organizers in Glasgow for their fantastic efforts that helped to make ICALP 2021 a great success.

Paper selection and work of the Program Committee. After 15 years running the ICALP conference with three tracks, ICALP returned in 2020 to the original two-track format. The ICALP 2021 program had the following two tracks:

- Track A: Algorithms, Complexity, and Games.
- Track B: Automata, Logic, Semantics, and Theory of Programming.

The PC chairs for the two tracks of ICALP 2021 were Nikhil Bansal (Track A) and James Worrell (Track B). Their efforts were supported by 70 members of the Program Committees (44 in track A and 26 in track B). In response to the call for papers, a total of 462 submissions were received: 361 for Track A and 101 for Track B. Each submission was assigned to at least three Program Committee members, aided by 761 external subreviewers.The Program Committees decided to accept 137 papers for inclusion in the scientific program: 108 papers for Track A and 29 for Track B. This gives the acceptance rate for the entire conference to be 29.6%. The selection was made by the Program Committees based on originality, quality, and relevance to theoretical computer science. The quality of the submitted manuscripts was very high, and unfortunately many strong papers could not

	2021	2020	2019	2018	2017	2016	2015	2014	2013	2012
Total	462	470	490	502	459	515	507	477	423	433
Track A	361	347	316	346	296	319	328	312	249	249
Track B	101	123	103	96	108	121	114	106	114	105
Track C	—	—	71	60	55	75	65	59	60	79

Table 1: Number of submitted papers at ICALP 2012–2021.

be selected. We take this opportunity of thanking all of them, the Program Committees and the subreviewers for doing an exceptional job in these particularly challenging times.

	2021	2020	2019	2018	2017	2016	2015	2014	2013	2012
Total	137	138	146	147	137	146	143	136	124	123
Track A	108	102	94	98	88	89	89	87	71	71
Track B	29	36	31	30	32	36	34	31	33	30
Track C	—	—	21	19	17	21	20	18	20	22

Table 2: Number of accepted papers at ICALP 2012–2021.

	2021	2020	2019	2018	2017	2016	2015	2014	2013	2012
Total	29.6	29.4	29.8	29.3	29.8	28.3	28.2	28.5	29.3	28.4
Track A	29.9	29.4	29.7	28.3	29.7	27.9	27.1	27.9	28.5	28.5
Track B	28.7	29.3	30.1	31.3	29.6	29.8	29.8	29.2	28.9	28.6
Track C	—	—	29.6	31.7	30.9	28.0	30.8	30.5	33.3	27.8

Table 3: Acceptance rates (in %) for ICALP 2012–2021.

Statistical information about the number of papers submitted and accepted for the last several editions of the ICALP conference, as well as acceptance rates, are available in Tables 1–3.

Invited presentations. In addition to the contributed talks, ICALP 2021 featured three invited presentations by Christel Baier (Track B), Keren Censor-Hillel (Track A), and David Woodruff (Track A), and three unifying talks by Andrei Bulatov, Toniann Pitassi and Adi Shamir:

- Christel Baier (Technical University of Dresden): *From Verification to Causality-Based Explications*.
- Keren Censor-Hillel (Technion): *Distributed Subgraph Finding: Progress and Challenges*.
- David Woodruff (Carnegie Mellon University): *A Very Sketchy Talk*.
- Andrei Bulatov (Simon Fraser University): *Symmetries and Complexity*.
- Toniann Pitassi (University of Toronto): *Algebraic Proof Systems*.
- Adi Shamir (Weizmann Institute of Science): *Error Resilient Space Partitioning*.

ICALP Proceedings. As it has been the tradition since 2016, ICALP proceedings were published with *LIPIcs*. *LIPIcs – Leibniz International Proceedings in Informatics* is a series of high-quality conference proceedings across all fields in informatics established in cooperation with Schloss Dagstuhl–Leibniz Center for Informatics. All 137 ICALP 2021 contributed papers presented at the conference, papers accompanying the invited talks of Christel Baier, Andrei Bulatov, Keren Censor-Hillel, and Adi Shamir, and abstracts of the invited presentation of Toniann Pitassi and David Woodruff were published according to the principle of Open Access in LIPIcs Proceedings volume 198, and made available online and free of charge at `https://drops.dagstuhl.de/opus/portals/lipics/index.php?semnr=16196`. The proceedings editors are Nikhil Bansal, Emanuela Merelli and James Worrell.

ICALP and EATCS Awards. The EATCS sponsors awards for both a best paper and a best student paper in each of the two tracks at ICALP, as selected by the Program Committees. During the general assembly, the ICALP Best Paper and Best Student Paper Awards were presented to the authors of the following papers:

Best Paper in Track A: Sayan Bhattacharya and Peter Kiss. *Deterministic Rounding of Dynamic Fractional Matchings*.

Best Paper in Track B: Antoine Amarilli, Louis Jachiet and Charles Paperman. *Dynamic Membership for Regular Languages*.

Best Student Paper in Track A: Or Zamir. *Breaking the* 2^n *barrier for 5-coloring and 6-coloring*.

Best Student Paper in Track B: none.

The program of ICALP 2021 included presentations of several prestigious scientific awards sponsored or co-sponsored by EATCS:

- The **EATCS Award 2021**, the annual EATCS Distinguished Achievements Award given to acknowledge extensive and widely recognized contributions to theoretical computer science over a life long scientific career, was awarded to **Toniann Pitassi** (University of Toronto) 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.[2]

- The **Presburger Award 2020** for Young Scientists was awarded to **Shayan Oveis Gharan** (University of Washington) for his creative, profound, and ambitious contributions to the Traveling Salesman problem.[3]

- The **EATCS Distinguished Dissertation Awards 2021**, to promote and recognize outstanding dissertations in theoretical computer science were awarded to three researchers:

  - **Talya Eden** for her dissertation *Counting, Sampling and Testing Subgraphs in Sublinear-Time*, advised by Dana Ron at Tel-Aviv University,
  - **Marie Fortin** for her dissertation *Expressivity of first-order logic, star-free propositional dynamic logic and communicating automata*, advised by Benedikt Bollig and Paul Gastin at ENS Paris-Saclay,
  - **Vera Traub** for her dissertation *Approximation Algorithms for Traveling Salesman Problems*, advised by Jens Vygen at the University of Bonn.

- The EATCS has recognized six of its members for their outstanding contributions to theoretical computer science by naming them **EATCS Fellows** class of 2021:

  - **Luca Aceto** (Reykjavik University and Gran Sasso Science Institute) for fundamental contributions to concurrency theory, and outstanding

[2]The laudatio for the EATCS Award 2021 is available at `https://eatcs.org/index.php/component/content/article/1-news/2877-the-eatcs-award-2021-laudatio-for-toniann-toni-pitassi-`

[3]The laudatio for the Presburger Award 2021 is available at `https://eatcs.org/index.php/component/content/article/1-news/2886-presburger-award-2021-laudatio-for-shayan-oveis-gharan`

merits for the community of theoretical computer science, in particular as an inspiring president of EATCS.

- **Rajeev Alur** (University of Pennsylvania) for 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) for 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) for 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) for 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) for fundamental contributions to algorithms, including parameterized algorithms and kernelization.

The recipients of the EATCS Award 2021 (Toniann Pitassi), and of the Presburger Award 2021 (Shayan Oveis Gharan) gave online presentations in the Award Sessions at ICALP 2021.

We congratulate all the winners and we hope their achievements will put the highlights of research in theoretical computer science in the spotlight, and will serve as inspirations to young researchers in the years to come.

We hope that this conference report gives you a glimpse of the rich scientific and social programme that made the 48th ICALP an unforgettable conference. Everyone involved in the organization of ICALP 2021 deserves the warmest thanks from the TCS community.

We wish to thank all authors who submitted extended abstracts for consideration, the Program Committees for their effort, and all the referees who assisted the Program Committees in the evaluation process. We are also grateful to the Conference Chair Simon Gay and all the support staff of the Organizing Committee for organizing ICALP 2021. We also acknowledge support by the Scottish Informatics and Computer Science Alliance (SICSA) for sponsoring participation by PhD students from Scotland.

The General Assembly of the EATCS was informed about the progress of the organization of the *49th EATCS International Colloquium on Automata, Lan-*

guages and Programming, ICALP 2022, that will take place in Paris, France, on July 4–8, 2022, with Thomas Colcombet as the general chair for the conference. The ICALP 2022 Program Committee Chairs are David Woodruff (Track A) and Mikolaj Bojanczyk (Track B). The ICALP 2022 call for papers is available at `https://icalp2022.irif.fr/?page_id=17`.

We hope that you will make plans to submit your best work to ICALP 2022 and be able to go to Paris for the conference. We look forward to seeing you there.

Report on BCTCS 2022

The 38th British Colloquium for Theoretical Computer Science

11–13 April 2022, Swansea University

Olga Petrovska and Monika Seisenberger

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 2022 was hosted by Swansea University and held from 11th to 13th April 2022. The event attracted 67 registered participants, and featured an interesting and wide-ranging programme, including special sessions on explainable AI and theoretical aspects of security. A total of 28 contributed talks – predominantly by PhD students – were presented at the meeting alongside the six special session speakers and six keynote speakers.

A special theme for the meeting was **History of Theoretical Computer Science**. The four Keynote Speakers featured during the middle day of the colloquium were all chosen based on their lifetime achievements in developing theoretical computer ccience in the UK, and were invited to present personal histories of their fields of expertise. The meeting also featured a special discussion group on the pedagogy of theoretical computer science, led by Barnaby Martin and Eleni Akrida. Finally, there was a special session following the AGM devoted to celerating two important birthdays: John Tucker, 70, who founded BCTCS in 1985; and Faron Moller, 60, who served as its President for 15 years (2004-2019).

BCTCS 2023 will be hosted by Glasgow University from 3rd–5th March 2023. 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

Cliff Jones (Newcastle University)
Formal Methods in the UK - A (Personal) History

I owe my real start in "formal methods" to two spells in Vienna but that was long enough ago that I can talk about many years of developing and promoting such approaches in the UK. I'd like to describe how difficult things were back in the 70s, how much better they are today, but also to address why it has taken so long and the situation is still far from ideal. I won't have time to get very technical but will provide pointers to more technical material.

Alexander Knapp (Ausburg University)
Specifying Event/Data-based Systems

Event/data-based systems are controlled by events, their local data state may change in reaction to events. Numerous methods and notations for specifying such reactive systems have been designed, though with varying focus on the different development steps and their refinement relations. We first briefly review some of such methods, like temporal/modal logic, TLA, UML state machines, symbolic transition systems, CSP, synchronous languages, and Event-B with their support for parallel composition and refinement. We then present E↓-logic for covering a broad range of abstraction levels of event/data-based systems from abstract requirements to constructive specifications in a uniform foundation. E↓-logic uses diamond and box modalities over structured events adopted from dynamic logic, for recursive process specifications it offers (control) state variables and binders from hybrid logic. The semantic interpretation relies on event/data transition systems; specification refinement is defined by model class inclusion. Constructive operational specifications given by state transition graphs can be characterised by a single E↓-sentence. Also a variety of implementation constructors is available in E↓-logic to support, among others, event refinement and parallel composition. Thus the whole development process can rely on E↓-logic and its semantics as a common basis.

Mike Paterson (University of Warwick)
Algorithms and Complexity in the UK - A (Personal) History

One person's story of how they found their way into this area during the 60's and 70's.

Rick Thomas (University of Leicester / University of St Andrews)
Some Interactions Between Automata Theory and Algebraic Structures in the UK - A (Personal) History

We will survey some of the impact that automata and formal language theory has

had on the theory of algebraic structures, particularly group theory, focussing on some work done in the UK. We will also point out some results and questions in automata theory arising from these interactions.

Francesca Toni (Imperial College London)
Argumentative Explainable AI

It is widely acknowledged that transparency of automated decision making is crucial for deployability of intelligent systems, and explaining the reasons why some outputs are computed is a way to achieve this transparency. The form that explanations should take, however, is much less clear. In this talk I will explore two classes of explanations, which I call 'lean' and 'mechanistic': the former focus on the inputs contributing to decisions given in output; the latter reflect instead the internal functioning of the automated decision making fed with the inputs and computing those outputs. I will show how both classes of explanations can be supported by forms of computational argumentation, and will describe forms of argumentative XAI in some settings (e.g. for multi-attribute decision making and machine learning).

John Tucker (Swansea University)
A (Personal) History of Logic in Computer Science

I will look at the emergence of logic as a central source of tools in computer science through the lens of my scientific development. I will concentrate on mathematical logic and its impact on programming and software engineering, starting in the 1960s.

Contributed Talks

Ruba Alassaf (University of Manchester)
Uniform interpolation in modal logic

A formula ϕ' is said to be a *uniform* Σ*−interpolant* of ϕ if, for any ψ over the signature Σ, ψ is implied by ϕ' if and only if ψ is implied by ϕ. Uniform interpolation has been the subject of many recent research papers due to its potential use in applications. In this talk, we introduce uniform interpolation and its variants, and present a saturation-based inference system that computes a *local* uniform interpolant for a formula and a "keep" signature.

Fahad Alhabardi (Swansea University)
Verification of bitcoin's smart contracts in Agda using weakest preconditions for access control

We address the verification of bitcoin smart contracts using the interactive theorem prover Agda, focusging on two standard smart contracts that govern the distribution of bitcoins: Pay to Public Key Hash (P2PKH) and Pay to MultiSig (P2MS). Both are written in bitcoin's low-level language script, and provide the security property of access control to the distribution of bitcoins. We introduce an operational semantics of the script commands used in P2PKH and P2MS, and formalise it in the Agda proof assistant using Hoare triples. We advocate weakest preconditions in the context of Hoare triples as the appropriate notion for verifying access control. Two methodologies for obtaining human-readable weakest preconditions are discussed in order to close the validation gap between user requirements and formal specification of smart contracts: (1) a step-by-step approach, which works backwards instruction by instruction through a script, sometimes stepping over several instructions in one go; (2) symbolic execution of the code and translation into a nested case distinction, which allows to read off weakest preconditions as the disjunction of accepting paths. A syntax for equational reasoning with Hoare Triples is defined in order to formalise those approaches in Agda.

Richard Allen (Swansea University)
Runtime verification for Android security

Users of computer systems face a constant threat of cyberattacks by malware designed to cause harm or disruption to services, steal information, or hold the user to ransom. Cyberattacks are becoming increasingly prevalent on mobile devices like Android. Attacks and countermeasures become more sophisticated in an ever-increasing arms race. A novel attack method is "collusion", where the attack gets hidden by distributing the steps through many malicious software actors. We investigate the use of runtime verification to detect collusion attacks on the end-users device. We have developed a novel algorithm called *RRH* that is a variation

of an existing algorithm by Grigore Roşu and Klaus Havelund. Our approach is computationally efficient enough to detect collusion in realtime on the Android device and does not require prior knowledge of malware source code. Thus, it can detect future malware without modification to the detection system or the software under scrutiny.

Peace Ayegba (University of Glasgow)
Resource allocation problems in wireless communications

Wireless communications is transforming every sector of the economy as it enables billions of people to keep in touch with work and family via their smart devices. However, the current technology cannot meet the demands of the future (i.e., explosive growth in number of devices, better coverage and connection rate). For future wireless communications, the key technology that has the potential to enhance connectivity for billions of users is referred to as Cell Free massive Multiple-Input Multiple-Output (CF-mMIMO). One of the many challenges in CF-mMIMO is how to efficiently manage limited resources (spectrum, energy, and power) – the so called resource allocation problem in wireless communications. This talk will give a survey of some of the problems arising in this context, as well as mathematical techniques that have been explored to tackle them.

Martin Barrere (Imperial College London)
Trustworthy critical infrastructure systems

Critical infrastructure systems are systems that are considered vital for the proper functioning and development of a society. Some examples are power plants, smart grids, and water distribution networks. These systems are often composed of diverse interconnected subsystems, thus creating a complex network of highly interdependent software, processes, and hardware components. From a security perspective, cyber attacks on these cyber-physical systems can have very serious consequences such as flooding, blackouts, or even nuclear disasters. In this talk, I will cover some of the main concepts regarding critical infrastructure systems, their importance, and related security management challenges. Afterwards, I will present two of the research areas that we are currently working on, namely, identification of critical cyber-physical components and cyber-physical attack graphs. I will conclude my talk by describing a number of open problems and exciting security challenges that require further investigation.

Arnold Beckmann (Swansea University)
Blockchain-based cyber-physical trust systems and their application

Cyber Physical Trust Systems (CPTS) are cyber physical systems enriched with trust as an explicit, measurable, testable and verifiable system component. In this talk we describe our research in relation to CPTS that are driven by blockchain.

We explain what blockchain is. We highlight some of the security properties of Blockchain-based CPTS and their formal proofs using the Tamarin Prover tool. We also describe the context of this research by highlighting the application of Blockchain-based CPTS in real world projects we are involved in.

Harry Bryant (Swansea University)
Exploring the IC3 algorithm to improve the Siemens-Swansea ladder logic verification tool

The last years have seen a radical transformation in railways. Among the new infrastructure, signalling software is a key technology to increase rail capacity and automation. However, verifying the safety of this software is a challenge for the railway industry. Siemens Mobility and Swansea University have developed an effective automated verifier for railway interlocking programs, which our project sets out to improve. The purpose of our research is to deal efficiently with the false-positive problem, where verification fails although the property under discussion holds. This problem arises due to the specific algorithmic approach taken. We are investigating if the false-positive problem can be addressed by Bradley's IC3 algorithm and its parallel and symbolic variations. The overall goal of our research is to offer a solution that scales to a challenge in rail industry.

Victor Cai (Swansea University)
Counterexample visualisation in the railway domain

In railway, the placement of equipment on a track plan is safety-critical and underlies many constraints (more than 300 are known in industry). For ETCS Level 2, our industrial partner has a design tool that lets engineers place equipment on track plans. The objective of our project is to visualise constraint violations in this tool, discovered through SMT solving. There is an approach under development that translates data from the design tool into a directed graph, which is then encoded in the SMTLib2 format, and finally checked against a set of design rules using the Z3 SMT solver. Our contribution is to take the unsatisfiable core of such a check and visualise it on the track plan in the design tool, by reversing the two model transformations that made the track data accessible to SMT solving. Our work provides a new instance of the challenge to translate from the language of a prover back into the language of the domain from which the problem originated.

Jamie Duell (Swansea University)
On explainable AI for medical diagnostics and its potential

Explainable AI (XAI) aims to provide intelligible explanations to users to support decision making. Current XAI methods, such as SHAP, LIME, and Scoped Rules, focus on calculating feature importance in predictions. Although XAI has attracted increasing attention, applying XAI techniques to healthcare to inform

clinical decision making is still challenging. Our experimental results show that XAI methods circumstantially generate different top features. The identified features, including shared top important features, demonstrate clinical significance in making diagnosis under different scenarios. Our work shows that medical diagnostics is a promising domain for applying XAI technologies. We aim to further this coalition of important features and the respective impact to surrounding features, by introducing peer-influence as a means of identifying underlying relations.

Hsuan Fu (Université Laval)
XAI in finance and economics

Abdul Ghani (Durham University)
Depth lower bounds in stabbing planes for combinatorial principles

This talk will introduce stabbing planes as a proof system, discuss its relevance, and indicate a few methods of finding lower bounds.

James Hinns (Swansea University)
Quantifying underspecification with feature attribution algorithms

For a given dataset and machine learning (ML) task, various predictors can be constructed with different reasoning that produce near-optimal test performance. However, due to this variance in reasoning, some predictors can generalise whilst some perform unexpectedly on unseen data. The existence of multiple equally performing predictors exhibits *underspecification* of the ML pipeline used for producing such predictors and presents challenges for credibility. In this talk, we propose identifying underspecification by estimating the variance of reasoning within a set of near-optimal predictors produced by a pipeline. We implement our approach by producing a set of well-performing predictors, computing their Shapley additive explanations and then measuring various distance and correlation metrics between them. We demonstrate our approach on multiple datasets drawn from the literature, and in a COVID-19 virus transmission case study.

Sahar Jahani (London School of Economics)
Automated equilibrium analysis of 2x2x2 games

In this project, we study the Nash equilibria in three-player non-cooperative games in which each player has two strategies. We present a program that outputs a full description of these equilibria even for degenerate games, which has not been done before. These are the simplest games with more than two players. The players' best-response functions are non-linear and a new formulation for these is proposed. Finding Nash equilibria is divided into two main algorithms: one to compute the partially mixed equilibria, and one compute the completely mixed equilibria. Both algorithms are implemented in Python with a 3D representation of

best-response surfaces. The program will be added to the Game Theory Explorer, which is a web application for analyzing different types of games.

Amin Karamlou (University of Oxford)
Logical aspects of non-local games

A non-local game consists of a group of spatially separated players who are trying to convince an adversary (the referee) that they know the answers to a set of questions. Ever since the pioneering work of Bell in the 1960s this rather abstract framework has proven to be one of our most useful tools for understanding quantum advantage, that is, the ability of a quantum system to outperform a classical one. We focus on analysing non-local games by using techniques traditionally associated with the field of logic in computer science. I present an overview of the questions in this area, and the progress made thus far in solving them.

Nina Klobas (Durham University)
The complexity of temporal vertex cover in small-degree graphs

Temporal graphs model graphs whose underlying topology changes over time. Recently, the *temporal vertex cover* (*TVC*) and *sliding window temporal vertex cover* (Δ-*TVC* for time-windows of a fixed-length Δ) have been established as natural extensions of the classic vertex cover problem on static graphs with connections to areas such as surveillance in sensor networks. In this talk we initiate a systematic study of the complexity of TVC and Δ-TVC on sparse graphs. Our main result shows that for every $\Delta \geq 2$, Δ-TVC is NP-hard even when the underlying topology is described by a path or a cycle. This resolves an open problem and shows a surprising contrast between Δ-TVC and TVC for which we provide a polynomial-time algorithm. To circumvent this hardness, we present a number of exact and approximation algorithms for temporal graphs whose underlying topologies are given by a path, that have bounded vertex degree in every time step, or that admit a small-sized temporal vertex cover.

Veera Raghava Reddy Kovvuri (Swansea University)
Understanding the influence of controllable factors in feature attribution algorithms

Feature attribution algorithms enable their users to gain insight into the underlying patterns of large data sets through their feature importance calculation. Existing feature attribution algorithms treat all features in a data set homogeneously, which may lead to misinterpretation of consequences of changing feature values. In this work, we consider partitioning features into controllable and uncontrollable features, and propose the Controllable fActor Feature Attribution (CAFA) approach to compute the relative importance of controllable features. CAFA is different from existing feature attribution algorithms in the following aspects. First, as

already mentioned, we distinguish between controllable and uncontrollable features instead of treating them homogeneously. Secondly, we compute the relative importance of controllable features through selective perturbation and global-for-local interpretation. We applied CAFA to two medical datasets, a lung cancer dataset and a breast cancer dataset. The experimental results show that after using CAFA, though the model is built over all features, the explanations of controllable features do not interfere with the uncontrollable ones. We further studied CAFA on the COVID-19 non-pharmaceutical control measures dataset, which was collected by ourselves. This study also showed that the controllable and uncontrollable features are relatively independent.

Pardeep Kumar (Swansea University)
Security in cyber physical smart environments

Cyber physical systems (CPSs) are envisioned as one of the building blocks for the internet of things (IoT), critical infrastructures, automations, smart healthcare, smart environments, etc. In a typical CPS network, a sensor works as an eye and an actuator works as a leg to cater their respective applications via several low-powered heterogenous communication technologies, protocols, and systems. However, such CPSs and their applications are largely distributed systems and subject to several attacks such as malicious attempts to hijack and denial of services. This talk will discuss some of the technical challenges and a lightweight security mechanism in cyber physical networks.

David Kutner (Durham University)
Payment scheduling in the interval debt model

Financial networks consist of banks (nodes) connected by debts (edges). We introduce the interval debt model, in which debts are due at integer time intervals on a temporal graph with lifetime T. We define valid payment schedules as ones consistent with the obligations of the nodes in the input, and study the problem of finding a schedule which minimizes (or maximizes) the number of nodes which go bankrupt. We also study the problem of bailout minimization, in which the aim is to minimize the bailout payments made by a financial authority to make possible a schedule with no bankruptcies. We find that bankruptcy minimization and maximization are NP-hard even on directed acyclic graphs of bounded degree; that bailout minimization is NP-hard on graphs of bounded degree; and that bankruptcy minimization remains weakly NP-hard on graphs with $O(1)$ vertices.

Laura Larios-Jones (University of Glasgow)
Reordering edges in temporal graphs to maximise reachability

Temporal graphs consist of an underlying graph (G, E) and an assignment t of timesteps to edges that specifies when each edge is active. This allows us to

model spread through a network which is time-sensitive. We will keep the set of timesteps the same and explore reordering them to optimise reachability. Previous work has mainly explored minimising spread for applications such as epidemiology. Here, we will be looking at the opposite problem of increasing movement through a graph. In particular, our goal is to reorder the timesteps assigned to the edges in our graph such that the minimum number of vertices reachable from any starting vertex is maximised. We will discuss which analogous structural and algorithmic results from the minimisation case hold in this case. Maximising spread can be useful in situations where we would like information or resources to be shared efficiently, such as advertising or even vaccine rollout.

Dimitrios Los (University of Cambridge)
Balanced allocations with incomplete information

In balanced allocations, we have to allocate m tasks (balls) sequentially into n servers (bins) so as to minimise the gap, i.e. the difference of the maximum load to the mean m/n. The simple process of allocating each ball to a random bin achieves whp (with high probability) a $\Theta(\sqrt{m/n \log n})$ gap when $m \gg n$. A great improvement over this process is the two-choice process, where each ball queries the load of two randomly chosen bins and is then placed in the least loaded bin. It achieves whp a $\Theta(\log_2 \log n)$ gap for $m \geq n$. In the first part of this talk we give some background on balanced allocation processes. In the second part, we investigate variants of the two-Choice process where allocations are made under incomplete information, giving insights into this dramatic improvement of two-choice over one-choice.

Michael McKay (University of Glasgow)
Envy-freeness in fixed size coalition formation games

In this talk we consider a variation of the three-dimensional stable roommates problem, which can be thought of as a coalition formation game. Agents specify preference lists over their peers, and the task is to partition the agents into sets of size 3 based on these preferences. Instead of seeking a "stable" partition, our aim is to construct a partition in which no agent has "envy" for another. There is a close relation to stability. In this talk we assume that agents have additive preferences. For three definitions of envy-freeness, we show that it is NP-hard in general to find envy-free partitions, but we present new results relating to restrictions of each problem in which an envy-free partition can be found in polynomial time. These results help us explore the boundary between NP-hardness and polynomial-time solvability in problems of coalition formation.

Steffan Mon (Swansea University)
Routing as a game

The internet can be viewed as a game played on a graph, where each vertex makes forwarding decisions based on selfish preferences over the path the data takes. Changes in forwarding decisions by one vertex may cause others to change their decisions, causing a ripple effect through the graph, resulting in degraded performance. Path formation games formalise and model this scenario along with other problems of interest. Although not every instance of path formation games has a solution, we prove how some special cases are guaranteed to have an efficiently computable solution. We also show how lacking a structure known as a dispute wheel is a sufficient condition for a game to have a solution.

Yoàv Montacute (University of Cambridge)
The path of hidden pebbles

Pebble games are a type of Spoiler-Duplicator game used in finite model theory to compare models and examine expressive powers of logical systems. Recently, these games were shown to provide an equivalent intermediary between fragments of different logical languages and categorical structures named comonads. This realises a unified language that can be used to study combinatorics categorically and to introduce alternative consolidated proofs for results with a common general form. This talk will begin with a current overview on this research project, originating from the seminal work of Abramsky, Dawar and Wang. We then consider a new type of pebble game which utilises the idea of a hidden pebble and yields a novel homomorphism-counting theorem via a comonad whose coalgebras characterise pathwidth.

Kiana Samadpour Motalebi (University of Manchester)
Tableau methods for modal logic

Logics of confluence, defined by generalisations of the axiom of confluence, cover a wide range of standard modal logics including modal logic K and single axiom extensions with the axioms T, B, 5, BAN, alt1, G0111, G and D, which are simple instances of a generalised pattern of axioms of confluence. Two kinds of tableau calculi for this class of logics using respectively structural rules and propagation rules are studied. Structural rules ensure that the constructed models satisfy the characteristic frame conditions of the logic, whereas propagation rules only construct pre-models which are sufficient to discover unsatisfiability and can be completed or adapted to give concrete models.

Tamio-Vesa Nakajima (Oxford University)
Linearly ordered colouring of hypergraphs

Linearly ordered colouring is a type of special hypergraph colouring. In normal hypergraph colouring, one must colour each vertex with one of K different colours such that no hyperedge is monochromatic. In linearly ordered colouring,

one must colour each edge so that it has a unique maximum colour. In this talk we discuss several interesting results about such colourings – both tractability of certain cases, and hardness of others. In particular, we will discuss "promise" problems: rather than trying to find a linearly ordered colouring with K colours for arbitrary hypergraphs, we will assume that the hypergraph admits a linearly ordered colouring with fewer than K colours.

Hoang Nga Nguyen (Coventry University)
Towards systematic threat assessment and security testing for automotive OTA

Modern cars host numerous special-purpose computing and connectivity devices facilitating the correct functioning of various in-vehicle systems. These devices host complex software systems with over 100 million lines of code, requiring regular and timely updates for functional and security improvements. Addressing the shortcomings of the legacy update system, over-the-air (OTA) software update systems have emerged as an efficient, cost-effective, and convenient solution for delivering updates to automobiles remotely. While OTA offers several benefits, it introduces new security challenges requiring immediate attention, as attackers can abuse these update systems to undermine vehicle security and safety. This talk will present our model-based approaches to addressing the security assessment problem for automotive OTA systematically.

Uchenna Nnawuchi (Middlesex University)
Investigating the right to explainable AI in the context of the GDPR

The growth of Artificial Intelligence (AI) in various aspects of human society has affected the landscape of human cognition, social order and political power. Machine Learning (ML) algorithms are increasingly becoming responsible for decision-making in many industry sectors owing to their efficiency in performing tasks. However, the growing worry is that ML algorithms are fraught with opacity and bias. Some legal scholars and data scientists have claimed that the right to explanation exists and is a persuasively appealing approach for challenging and providing redress to ML opacity. However, the existence or otherwise of such a right is at the centre of global debate. This paper investigates whether there is a legal right to explanation of ML algorithms in the context of the GDPR. The paper analyses the argument from various scholars on both sides of the divide with a view to ascertaining whether the right actually exists.

Alpay Ozkeskin (University of Liverpool)
Scheduling with precedence constraints for electricity cost in smart grid

We study a scheduling problem arising in demand response management in smart grid. Consumers send in power requests with a flexible feasible time interval during which their requests can be served. The grid controller, upon receiving power

requests, schedules each request within the specified interval. The electricity cost is measured by a convex function of the load in each time slot. The objective is to schedule all requests with the minimum total electricity cost. Previous work has studied a special case where the power requirement and the duration a request demands are both unit-size. We extend this case by allowing jobs to have precedence constraints. We present a polynomial time offline algorithm that gives an optimal solution when the feasible time intervals for the jobs in order of precedence are the same.

Filippos Pantekis (Swansea University)
Towards massively parallel SAT

Parallelisation of SAT is an area that faces many challenges, including the sharing of information between threads causing restrictions to performance. In this work, we present steps in massively parallelising SAT checking using new GPGPU architectures in a manner that is both loosely coupled and fully scalable. We demonstrate techniques for successful parallelisation of SAT in these special environments and present early results in favour of their use.

Francis Southern (Swansea University)
Singular DP-reduction

It is useful to be able to reduce one set of boolean constraints to another, which is simpler relative to some complexity measure, in such a way that solutions can be translated back to the original problem. One operation which achieves this and has been utilised in both theoretical and practical work on the propositional satisfiability problem is Davis-Putnam reduction, which eliminates a variable v from a clause-set F by replacing the clauses containing v with their resolvents (on v). In general, this may increase the number of clauses. However, the number of clauses also decreases in the special case of singular DP-reduction, in which the eliminated variable occurs in one of its polarities only once. My talk will cover some structural properties of singular DP-reduction on minimally unsatisfiable clause-sets.

Theofilos Triommatis (University of Liverpool)
On maximising the visibility area with a rotating field of view

Imagine we have a polygon-shaped platform P and only one static spotlight outside of P. Which direction should the spotlight face to light most of the platform? More formally, we define and provide an algorithm for the problem of finding the maximum intersection between a polygon and a rotating field of view. If we were to guess the direction that yields the maximum intersection, we would have to go through an infinite number of guesses. We provide an FPTAS that approximates the rotation that yields the maximum intersection.

Galen Wilkerson (Imperial College London / University of Surrey)
Spontaneous emergence of computation in network cascades

Neuronal network computation and computation by avalanche supporting networks are of interest to the fields of physics, computer science (computation theory as well as statistical or machine learning) and neuroscience. Here we show that computation of complex Boolean functions on inputs arise spontaneously in threshold networks as a function of connectivity and antagonism (inhibition), computed by logic motifs. We also obtain lower bounds on function probabilities, and show that these results agree with those obtained by others.

Adam Wyner (Swansea University)
Explainable AI in law

Tansholpan Zhanabekova (University of Liverpool)
Deciding what is good-for-MDPs

Nondeterministic Good-for-MDP (GFM) automata are for MDP model checking and reinforcement learning what good-for-games automata are for synthesis: a more compact alternative to deterministic automata that displays nondeterminism, but only so much that it can be resolved locally, such that a syntactic product can be analysed. GFM has recently been introduced as a property for reinforcement learning, where the simpler Büchi acceptance conditions it allows to use is key. However, while there are classic and novel techniques to obtain automata that are GFM, there has not been a decision procedure for checking whether or not an automaton is GFM. We show that GFM-ness is decidable and provide an EXPTIME decision procedure as well as a PSPACE-hardness proof.

Report on CPM 2022

The 33rd Annual Symposium on Combinatorial Patter Matching

Nadia Pisanti
University of Pisa, Italy

The 33rd Annual Symposium on Combinatorial Patter Matching was held in Prague, Czech Republic, from June 27th to June 29th, 2022. Each year, CPM gathers scientists of many areas related to word combinatorics, discrete algorithms, string algorithms, that address problems such as text searching and indexing, data compression, pattern discovery, and that can be applied to bioinformatics, data mining, information retrieval, natural language processing, just to mention a few. The 2022 edition of CPM was held at the Faculty of Civil Engineering of the Czech Technical University in Prague, organised by Jan Holub, Jan Trávníček, Ondřej Guth, Tomáš Pecka, and Eliška Šestáková, Dominika Draesslerová, Štepán Plach y, Lucie Procházková, Regina Šmidová. The scientific program consisted of 3 invited talks, 2 highlights talks, and 26 regular talks of accepted papers which had been chosen by the Program Committee out of 43 submissions (coming from authors from 20 different countries and 4 different continents) on the basis of three reviews for each submission. The Program Committee consisted of 28 members (24 men, 4 women) from 18 different countries.
The detailed program, as well as some pictures can be found on the website https://www.stringology.org/event/CPM2022/.
The proceedings, edited by the program committee co-chairs Hideo Bannai and Jan Holub, have been published in volume 223 of LIPIcs. They are open access and can be found here:
https://drops.dagstuhl.de/opus/portals/lipics/index.php?semnr=16232.
The invited talks covered several interesting topics and were given by:

1. ***Takehiro Ito*** (Tohoku University, Japan):
   "Invitation to Combinatorial Reconfiguration"
2. ***Jeffrey Shallit*** (University of Waterloo, Canada):
   "Using automata and a decision procedure to prove results in pattern matching"
3. ***Sharma V. Thankachan*** (University of Central Florida, USA):
   "Compact Text Indexing for Advanced Pattern Matching Problems: Parametrized, Order-isomorphic, 2D, etc."

The highlights talks, introduced for the first time in CPM 2019, are special sessions dedicated to as many presentations of the highlights of recent results and

developments in combinatorial pattern matching topics, that have been recently published in other venues.
This year, CPM featured the following two highlight talks:

1. ***Tomasz Kociumaka*** ((MPI, Germany). "Small space and streaming pattern matching with k edits", paper presented at FOCS 2021.
2. ***Moses Ganardi*** (MPI, Germany). "Compression by Contracting Straight-Line Programs", paper presented at ESA 2021, and extended in J.ACM 2021.

The conference had 50 on-line and 59 in-person participants. The business meeting of CPM 2022 was chaired by the Steering Commitee and took place on June 27th, at the end of the afternoon session. In the business meeting, the PC chair and local organiser Jan Holub gave briefly an overview on the conference organisation. At the end of the meeting, Laurent Bulteau presented the conference edition of CPM 2023 which will take place in Paris, France. The social program took place the second day of CPM 2022 and started with a sightseeing ride through the city center on board of one of Prague's historical trams that used to roam Prague streets in the first half of the 20th century. The tram ride was followed by a guided tour through beautiful Prague's Old Town which started with the Charles Bridge and ended with a conference dinner at the Tiskarna Restaurant.
Thank you to the local organizers for their excellent work and all participants for the nice and successful conference.
Looking forward to see you at CPM 2023 in Paris!

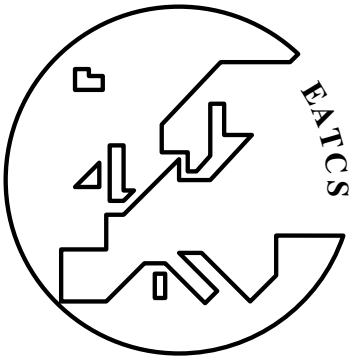
European

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
- Glasgow, UK (virtual conference) 2021

(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. Artur Czumaj,
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