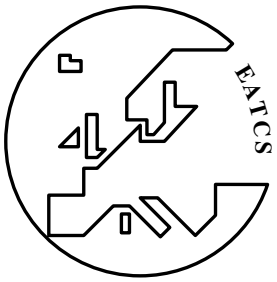
ISSN 0252–9742

Bulletin

of the

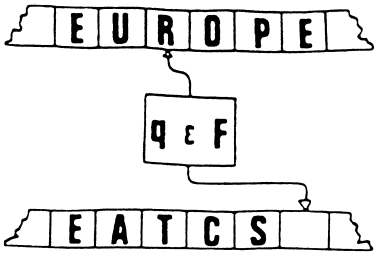
European Association for Theoretical Computer Science

EATCS



Number 145 February 2025

Council of the European Association for Theoretical Computer Science



President:	Giuseppe F. Italiano	Italy
Vice Presidents:	Antoine Amarilli	France
	Inge Li Gørtz	Denmark
Treasurer:	Gabriele Fici	Italy
Bulletin Editor:	Stefan Schmid	Germany

Antoine Amarilli	France	Thore Husfeldt	Sweden, Denmark
Ivona Bezakova	USA	Giuseppe F. Italiano	Italy
Tiziana Calamoneri	Italy	Emanuela Merelli	Italy
Thomas Colcombet	France	Anca Muscholl	France
Anne Driemel	Germany	Charles Paperman	France
Funda Ergün	USA	Eva Rotenberg	Denmark
Javier Esparza	Germany	Jiri Sgall	Czech Republic
Gabriele Fici	Italy	Jukka Suomela	Finland
Inge Li Goertz	Denmark	Szymon Torunczyk	Poland
Fabrizio Grandoni	Switzerland	Bianca Truthe	Germany

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)	Artur Czumaj	(2020–2024)

Secretary Office:	Dmitry Chistikov	UK
	Efi Chita	Greece

EATCS Council Members

email addresses

Antoine Amarilli . . . . . a3nm@a3nm.net
Ivona Bezakova . . . . . ib@cs.rit.edu
Tiziana Calamoneri . . . . . calamo@di.uniroma1.it
Thomas Colcombet . . . . . Thomas.Colcombet@irif.fr
Anne Driemel . . . . . driemel@cs.uni-bonn.de
Funda Ergün . . . . . chair.sigact@sigact.acm.org
Javier Esparza . . . . . esparza@in.tum.de
Gabriele Fici . . . . . gabriele.fici@unipa.it
Inge Li Goertz . . . . . inge@dtu.dk
Fabrizio Grandoni . . . . . fabrizio@idsia.ch
Thore Husfeldt . . . . . thore@itu.dk
Giuseppe F. Italiano . . . . . giuseppe.italiano@uniroma2.it
Emanuela Merelli . . . . . emanuela.merelli@unicam.it
Anca Muscholl . . . . . anca@labri.fr
Charles Paperman . . . . . charles.paperman@univ-lille.fr
Eva Rotenberg . . . . . eva@rotenberg.dk
Stefan Schmid . . . . . stefan.schmid@tu-berlin.de
Jiri Sgall . . . . . sgall@iuuk.mff.cuni.cz
Jukka Suomela . . . . . jukka.suomela@aalto.fi
Szymon Torunczyk . . . . . szymtor@mimuw.edu.pl
Bianca Truthe . . . . . bianca.truthe@informatik.uni-giessen.de

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.

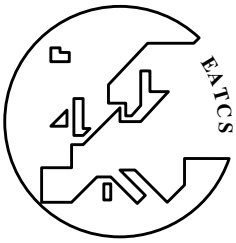
The Editor welcomes proposals for surveys, tutorials, and thematic issues of the Bulletin dedicated to currently hot topics, as well as suggestions for new regular sections.

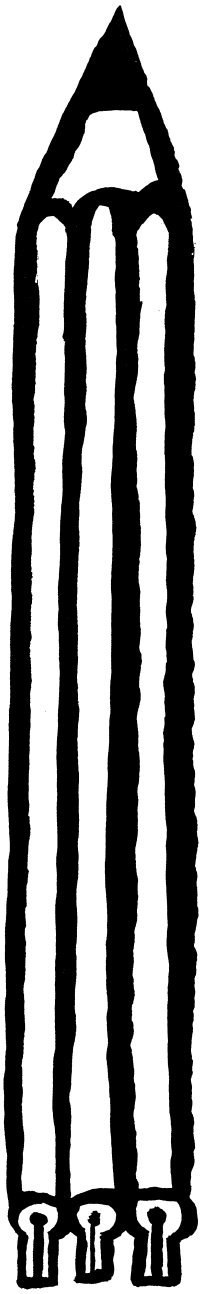
Table of Contents

News and Conference Reports

Miscellaneous

EATCS Matters





Dear EATCS members,

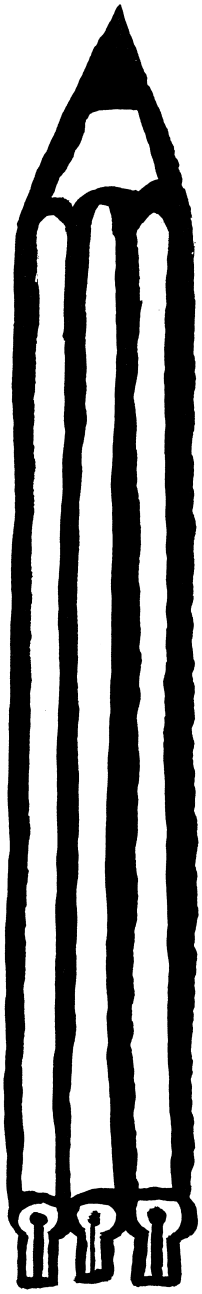
As we step into 2025, I would like to extend my warmest wishes to all of you. May this year bring you good health, inspiring research breakthroughs, and exciting opportunities to connect and collaborate within our vibrant theoretical computer science community.

I would also like to remind you of the ongoing calls for nominations for several prestigious awards sponsored by EATCS, including the EATCS Award, the Presburger Award, the EATCS Distinguished Dissertation Award, and the EATCS Fellows. Additionally, we are proud to be part of joint awards such as the Gödel Prize, the Alonzo Church Award, and the Dijkstra Prize. I encourage you to nominate deserving colleagues and outstanding contributions to our field.

I am particularly looking forward to the 52nd EATCS International Colloquium on Automata, Languages, and Programming (ICALP 2025), which will take place in Aarhus, Denmark, July 8-11. As the flagship conference of our association, ICALP continues to foster cutting-edge discussions and collaborations. The event will be preceded by a series of workshops on July 7, and I sincerely hope to see many of you there!

Beyond ICALP, several other exciting EATCS-affiliated conferences will take place over the year, including:

MFCS 2025 - The 50th International Symposium on Mathematical Foundations of Computer Science in Warsaw, Poland, August 25-29 (MFCS 2025).



ESA 2025 - The 33rd Annual European Symposium on Algorithms, also in Warsaw, Poland, September 15-19 (ESA 2025).

DISC 2025 - The 38th International Symposium on Distributed Computing in Berlin, Germany, October 27-November 1 (DISC 2025).

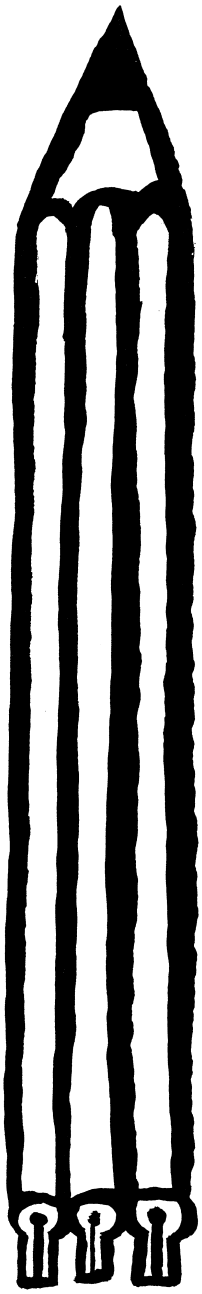
I look forward to meeting many of you at ICALP in Aarhus and at other conferences and workshops throughout the year. These occasions provide us with invaluable moments to exchange ideas and discuss how we can further enhance the impact of EATCS in our community.

As we move forward, I am eager to hear your thoughts and ideas on how we can best serve our scientific community. Your feedback and suggestions are always welcome - please do not hesitate to reach out. Together, we can continue pushing the boundaries of theoretical computer science and make meaningful contributions to the world.

Wishing you a fantastic 2025, and I hope to see you soon!

Giuseppe F. Italiano
Luiss University, Rome
President of EATCS
president@eatcs.org

February 2025



Dear EATCS member!

I hope you had a good start into the year and I wish you all the best for 2025.

First of all, I am very happy to welcome a new editor: Ioana-Oriana Bercea from KTH will be the editor of the Algorithmics Column, together with Thomas Erlebach.

This Winter issue of the Bulletin features several interesting researchers and articles. In the Interview Column, we talk with Antoine Amarilli, who currently also serves as the vice- president of EATCS.

In the TCS on the Web Column, Stefan Neumann talks with Rachel Cummings and Sumegha Garg, looking behind the scenes of TCS+, an online seminar series in theoretical computer science.

The Algorithmics Column features Nicola Cotumaccio, whose PhD thesis on data compression and automata theory was selected by the Italian Chapter of the EATCS for the Best PhD Thesis Award.

In the Distributed Computing Column, Francesco d'Amore surveys some recent exciting results in quantum distributed computing, focusing on the question of when a distributed quantum advantage is possible.

In the Computational Complexity Column, Oliver Korten provides a survey of range avoidance search problems, which lead to several exciting new results in derandomization and circuit complexity.

The Education Column, authored by Shriram Krishnamurthi and Tim Nelson, presents a proposal for increasing the accessibility

of formal methods to the large number of students who could benefit from it but may not be well-served by traditional introductions.

An obituary for Luca Trevisan, who was also our editor for the Theory Blogs Column, highlights his rich and diverse legacy of influential results and the profound impact he had on our community.

We also honor Arto Salomaa, a world-class Finnish mathematician and theoretical computer scientist, who passed away in January this year. He is renowned for his foundational contributions to automata theory and formal languages, among numerous other. He also played a key role in the establishment and early development of the EATCS and served as its president from 1979 to 1985.

Last but not least, Matthias Bentert reports on his conference experiences at ITCS 2025 and SODA 2025, and Vikraman Choudhury reports on POPL 2025. A big thank you to all editors and authors of this issue for their contributions. Enjoy the new Bulletin and I hope to see you sometime this year!

Stefan Schmid, Berlin
February 2025

The EATCS Award 2025

Call for Nominations

Deadline: March 15, 2025

The European Association for Theoretical Computer Science (EATCS) annually honours a respected scientist from our community with the prestigious EATCS Distinguished Achievement Award. The award is given to acknowledge extensive and widely recognized contributions to theoretical computer science over a life long scientific career. For the EATCS Award 2025, candidates may be nominated to the Awards Committee consisting of

- Valerie King (chair)
- Javier Esparza
- Amos Fiat

Nominations will be kept strictly confidential. They should include supporting justification and be signed by at least three co-signatories. Others that wish to be endorsers may send a one-page endorsement. Nominations will normally be kept active for three years. Nominators may choose to update a nomination or renew it beyond this period by resubmitting.

Nominations should be sent by e-mail to the chair of the EATCS Award Committee:

Valerie King
eatcs-award@eatcs.org

by March 15, 2025. The list of previous recipients of the EATCS Award is at http://eatcs.org/index.php/eatcs-award

The next award will be presented during ICALP 2025 in Aarhus, Denmark https://conferences.au.dk/icalp2025.

The Presburger Award for Young Scientists 2025

Call for Nominations

Deadline: 13 March 2025

The Presburger Award recognises outstanding contributions by a young scientist in theoretical computer science, documented by a published paper or a series of published papers. It is named after Mojzesz Presburger who accomplished his ground-breaking work on decidability of the theory of addition (known today as Presburger arithmetic) as a student in 1929. The award is conferred annually by the European Association for Theoretical Computer Science (EATCS) at the International Colloquium on Automata, Languages, and Programming (ICALP).

Nominated scientists have to be young scientists as of January 1st of the year of the award, by either being at most 35 years old (that is, for 2025, the nominee should be born in 1989 or later), or having finished their PhD at most 6 years ago. The eligibility is extended due to leaves of absence according to the same rules as the ERC starting grant: https://erc.europa.eu/apply-grant/starting-grant.

Nominations for the Presburger Award can be submitted by any member or group of members of the theoretical computer science community, but not by the nominee themselves nor the advisors for their master's thesis or doctoral dissertation.

The Presburger Award committee for 2025 consists of Tal Malkin (Columbia University, chair), Joël Ouaknine (Max Planck Institute for Software Systems) and Shiri Chechik (Tel-Aviv University). Nominations, consisting of a two page justification and (links to) the relevant publications, as well as additional supporting letters, should be sent by e-mail to:

`presburger-award@eatcs.org`

The subject line of every nomination should start with *Presburger Award 2025*, and the message must be received before **March 13th, 2025**.

The award includes an amount of 1000 Euro and an invitation to ICALP 2025 for a lecture.

Previous Winners:
Mikołaj Bojanczyk, 2010
Patricia Bouyer-Decitre, 2011
Venkatesan Guruswami and Mihai Patrascu, 2012
Erik Demaine, 2013
David Woodruff, 2014
Xi Chen, 2015
Mark Braverman, 2016
Alexandra Silva, 2017
Aleksander Madry, 2018
Karl Bringmann and Kasper Green Larsen, 2019
Dmitriy Zhuk, 2020
Shayan Oveis Gharan, 2021
Dor Minzer, 2022
Aaron Bernstein and Thatchaphol Saranurak, 2023
Justin Hsu and Pravesh Kothari, 2024

Official website: `http://www.eatcs.org/index.php/presburger`

EATCS-Fellows 2025

Call for Nominations

Deadline: March 7th, 2025

Proposals for Fellow consideration in 2025 should be submitted by MARCH 7th, 2025 by email to the EATCS Secretary - secretary@eatcs.org. The subject line of the email should read "EATCS Fellow Nomination - < surname of candidate >".
The EATCS Fellows Program is established by the Association 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.
In order to be considered by the EATCS Fellows Selection Committee, candidates must be nominated by at least four EATCS Members. Please verify your membership at http://www.eatcs.org/.
The EATCS Fellows Selection Committee consists of

- Luca Aceto
- Orna Kupferman
- Stefano Leonardi (chair)
- Paul Spirakis

INSTRUCTIONS: Please note that all nominees and nominators must be EATCS members. A nomination should consist of details on the items below. It can be co-signed by several EATCS members. Two nomination letters per candidate are welcomed. If you are supporting the nomination from within the

candidate's field of expertise, it is expected that you will be specific about the individual's technical contributions.

To be considered, nominations for 2025 must be received by March 07, 2025.

1. Name of candidate, candidate's current affiliation and position, candidate's email address, postal address and phone number. Nominator(s) relationship to the candidate.

2. Short summary of candidate's accomplishments (citation – 25 words or less).

3. Candidate's accomplishments: Identify the most important contributions that qualify the candidate for the rank of EATCS Fellow according to the following two categories:

A) Technical achievements,

B) Outstanding service to the TCS community.

Please limit your comments to at most three pages.

4. Nominator(s): Name(s) Affiliation(s), email and postal address(es), phone number(s)

EATCS Distinguished Dissertation Award 2024

Call for Nominations

Deadline: March 5, 2025

The EATCS establishes the Distinguished Dissertation Award to promote and recognize outstanding dissertations in the field of Theoretical Computer Science. Any PhD dissertation in the field of Theoretical Computer Science that has been successfully defended in 2024 is eligible. Up to three dissertations will be selected by the committee for year 2024. The dissertations will be 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 dissertation must be submitted by the author as an attachment to an email message sent to the address dissertation-award@eatcs.org with subject EATCS Distinguished Dissertation Award 2024 by March 7, 2025. The body of the message must specify: Name and email address of the candidate; Title of the dissertation; Department that has awarded the PhD and denomination of the PhD program; Name and email address of the thesis supervisor; Date of the successful defense of the thesis.

A five page abstract of the dissertation and a letter by the thesis supervisor certifying that the thesis has been successfully defended must also be included. In addition, an endorsement letter from the thesis supervisor, and possibly one more endorsement letter, must be sent by the endorsers as attachments to an email message sent to the address dissertation-award@eatcs.org with subject EATCS DDA 2024 endorsement. The name of the candidate should be clearly specified in the message.

The dissertations will be selected by the following committee:

- Standa Zivny (chair)
- Petra Berenbrink
- Veronique Bruyere
- Kasper Green Larsen
- Emanuela Merelli

The award committee will solicit the opinion of members of the research community as appropriate. Theses supervised by members of the selection committee are not eligible. The EATCS is committed to equal opportunities, and welcomes submissions of outstanding theses from all authors.

Alonzo Church Award 2025

Call for Nominations

Deadline: March 31, 2025

INTRODUCTION:

An annual award, called the Alonzo Church Award for Outstanding Contributions to Logic and Computation, was established in 2015 by the ACM Special Interest Group for Logic and Computation (SIGLOG), the European Association for Theoretical Computer Science (EATCS), and the European Association for Computer Science Logic (EACSL). The award is for an outstanding contribution represented by a paper or by a small group of papers published within the past 25 years. This time span allows the lasting impact and depth of the contribution to have been established. The award can be given to an individual, or to a group of individuals who have collaborated on the research. For the rules governing this award, see https://siglog.org/alonzo-church-award/, https://www.eatcs.org/index.php/church-award/, and https://www.eacsl.org/alonzo-church-award/.

THE 2024 AWARD

The 2024 Alonzo Church Award was given jointly to Thomas Ehrhard and Laurent Regnier for giving a logical and computational account of differentiation, bringing Taylor expansion to the Curry-Howard correspondence, which had a major impact on programming language semantics.

ELIGIBILITY AND NOMINATIONS:

The contribution must have appeared in a paper or papers published within the past 25 years. Thus, for the 2025 award, the cut-off date is January 1, 2000. When a paper has appeared in a conference and then in a journal, the date of the journal publication will determine the cut-off date. In addition, the contribution must not yet have received recognition via a major award, such as the Turing Award, the Kanellakis Award, or the Goedel Prize. (The nominee(s) may have received such awards for other contributions.) While the contribution can consist of conference or journal papers, journal papers will be given a preference.

Nominations for the 2025 award are now being solicited. The nominating letter must summarise the contribution and make the case that it is fundamental and outstanding. The nominating letter can have multiple co-signers. Self-nominations are excluded. Nominations must include: a proposed citation (up to 25 words); a succinct (100-250 words) description of the contribution; and a detailed statement (not exceeding four pages) to justify the nomination. Nominations may also be accompanied by supporting letters and other evidence of worthiness. Nominations for the 2025 award are automatically considered for all future editions of the award, until they receive the award or the nominated papers are no longer eligible.

PROCEDURE AND DEADLINE

Nominations to the 2025 Alonzo Church Award, taking the form of a single PDF file, should be sent by 2025 March 31 to thomas.colcombet@irif.fr.

PRESENTATION OF THE AWARD

The 2025 award will be presented at the Thirty-Fourth EACSL Annual Conference on Computer Science Logic 2026 (CSL) which is scheduled to take place during 23-28 February 2026 in Paris, France. The award winner, or one of the award winners, will be invited to give a lecture during the conference. The awardee(s) will receive a certificate and a cash prize of USD 1,500. If there are multiple awardees, this amount will be shared.

AWARD COMMITTEE

The 2025 Alonzo Church Award Committee consists of the following five members:

- Thomas Colcombet (chair),
- Anuj Dawar,
- Marcelo Fiore,
- Alexandra Silva
- and Igor Walukiewicz.

The Gödel Prize 2025

Call for Nominations

Deadline: April 11, 2025

The Gödel Prize for outstanding papers in the area of theoretical computer science is sponsored jointly by the European Association for Theoretical Computer Science (EATCS) and the Association for Computing Machinery, Special Interest Group on Algorithms and Computation Theory (ACM SIGACT). The award is presented annually, with the presentation taking place alternately at the EATCS International Colloquium on Automata, Languages, and Programming (ICALP) and the ACM Symposium on Theory of Computing (STOC). The 33rd Gödel Prize will be awarded at the 57th Annual ACM Symposium on Theory of Computing (STOC 2025) in Prague, Czech Republic, June 23 - 27, 2025.

The Prize is named in honor of Kurt Gödel in recognition of his major contributions to mathematical logic and of his interest, discovered in a letter he wrote to John von Neumann shortly before von Neumann's death, in what has become the famous "P versus NP" question. The Prize includes an award of USD 5,000.

Award Committee: The 2025 Award Committee consists of Mikołaj Bojańczyk (University of Warsaw), Artur Czumaj (University of Warwick), Yuval Ishai (Technion), Anna Karlin (University of Washington), Marta Kwiatkowska (University of Oxford), Tim Roughgarden (chair, Columbia University).

Eligibility: The 2025 Prize rules are given below and they supersede any different interpretation of the generic rule to be found on websites of both SIGACT and EATCS. Any research paper or series of papers by a single author or by a team of authors is deemed eligible if:

- The main results were not published (in either preliminary or final form) in a journal or conference proceedings before January 1st, 2012.

- The paper was published in a recognized refereed journal no later than December 31, 2024.

The research work nominated for the award should be in the area of theoretical computer science. Nominations are encouraged from the broadest spectrum of

the theoretical computer science community so as to ensure that potential award winning papers are not overlooked. The Award Committee shall have the ultimate authority to decide whether a particular paper is eligible for the Prize.

Nominations: Nominations for the award should be submitted by email to the Award Committee Chair: tim.roughgarden@gmail.com. Please make sure that the Subject line of all nominations and related messages begin with "Goedel Prize 2025." To be considered, nominations must be received by April 11, 2025.

A nomination package should include:

1. A printable copy (or copies) of the journal paper(s) being nominated, together with a complete citation (or citations) thereof.

2. A statement of the date(s) and venue(s) of the first conference or workshop publication(s) of the nominated work(s) or a statement that no such publication has occurred.

3. A brief summary of the technical content of the paper(s) and a brief explanation of its significance.

4. A support letter or letters signed by at least two members of the scientific community.

Additional support letters may also be received and are generally useful. The nominated paper(s) may be in any language. However, if a nominated publication is not in English, the nomination package must include an extended summary written in English.

Those intending to submit a nomination should contact the Award Committee Chair by email well in advance. The Chair will answer questions about eligibility, encourage coordination among different nominators for the same paper(s), and also accept informal proposals of potential nominees or tentative offers to prepare formal nominations. The committee maintains a database of past nominations for eligible papers, but fresh nominations for the same papers (especially if they highlight new evidence of impact) are always welcome.

Selection Process: The Award Committee is free to use any other sources of information in addition to the ones mentioned above. It may split the award among multiple papers, or declare no winner at all. All matters relating to the selection process left unspecified in this document are left to the discretion of the Award Committee.

Recent Winners (all winners since 1993 are listed at `http://www.sigact.org/Prizes/Godel/` and `http://eatcs.org/index.php/goedel-prize`):

2024: Ryan Williams

2023: Samuel Fiorini, Serge Massar, Sebastian Pokutta, Hans Raj Tiwary and Ronald de Wolf: Exponential Lower Bounds for Polytopes in Combinatorial Optimization. STOC 2012. JACM, 62(2), 17:1-17:23, 2015. Thomas Rothvoss. The matching polytope has exponential extension complexity. STOC 2014. JACM, 64(6),1-19, 2017

2022: Brakerski, Zvika; Vaikuntanathan, Vinod : Efficient Fully Homomorphic Encryption from (Standard) LWE. SIAM Journal on Computing. **43** (2): 831–871 (preliminary version in Foundations of Computer Science, FOCS 2011). Brakerski, Zvika; Gentry, Craig; Vaikuntanathan, Vinod (2012: (Leveled) Fully Homomorphic Encryption without Bootstrapping. ACM Trans. Comput. Theory 6(3): 13:1-13:36 (preliminary version in Innovations in Theoretical Computer Science, ITCS 2012).

2021: Andrei Bulatov, The Complexity of the Counting Constraint Satisfaction Problem. J. ACM 60(5): 34:1–34:41 (2013). Martin E. Dyer and David Richerby: An Effective Dichotomy for the Counting Constraint Satisfaction Problem. SIAM J. Computing. 42(3): 1245–1274 (2013). Jin-Yi Cai and Xi Chen: Complexity of Counting CSP with Complex Weights. J. ACM 64(3): 19:1–19:39 (2017).

2020: Robin A. Moser and Gábor Tardos, *A constructive proof of the general Lovász Local Lemma*, Journal of the ACM (JACM), Volume Issue 2, 2010 (preliminary version in Symposium on Theory of Computing, STOC 2009)

2019: Irit Dinur, *The PCP theorem by gap amplification,* Journal of the ACM (JACM), Volume 54 Issue 3, 2007 (preliminary version in Symposium on Theory of Computing, STOC 2006)

2018: Oded Regev, *On lattices, learning with errors, random linear codes, and cryptography*, Journal of the ACM (JACM), Volume 56 Issue 6, 2009 (preliminary version in Symposium on Theory of Computing, STOC 2005).

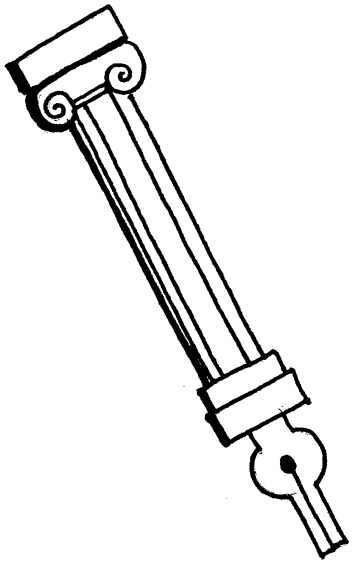
2017: Cynthia Dwork, Frank McSherry, Kobbi Nissim and Adam Smith**,** *Calibrating Noise to Sensitivity in Private Data Analysis,* Journal of Privacy and Confidentiality, Volume 7, Issue 3, 2016 (preliminary version in Theory of Cryptography, TCC 2006).

2016: Stephen Brookes, *A Semantics for Concurrent Separation Logic*. Theoretical Computer Science 375(1-3): 227-270 (2007). Peter W. O'Hearn, *Resources, Concurrency, and Local Reasoning*. Theoretical Computer Science 375(1-3): 271-307 (2007).

2015: Dan Spielman and Shang-Hua Teng, *Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems*, Proc. 36th ACM Symposium on Theory of Computing, pp. 81-90, 2004; *Spectral sparsification of graphs*, SIAM J. Computing 40:981-1025, 2011; *A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning*, SIAM J. Computing 42:1-26, 2013; *Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems*, SIAM J. Matrix Anal. Appl. 35:835-885, 2014.

Institutional Sponsors

EATCS
Columns



The Interview Column

BY

Chen Avin and Stefan Schmid

Ben Gurion University, Israel and TU Berlin, Germany
{chenavin,schmiste}@gmail.com

KNOW THE PERSON BEHIND THE PAPERS

Today: Antoine Amarilli

Bio: *Antoine Amarilli has an Advanced Research Position at Inria Lille in France, and is on leave from an associate professor position at Télécom Paris. He studied at École normale supérieure and received his PhD in 2016 from Télécom Paris, for which he was awarded a Beth Dissertation Award. His PhD focused on probabilistic databases and open-world query answering: his advisor was Pierre Senellart. He was the director of the ICPC SWERC programming contest from 2017 to 2020. He served as the inaugural managing editor for the TheoretiCS journal from 2021 to 2023, and was a local organiser in 2022 for Highlights of Logic, Games, and Automata, for which he is now environmental chair. His current research focuses on database theory and theoretical computer science; he received his habilitation from Institut Polytechnique de Paris in 2023. He co-authored works which received best paper awards at the ICDT conference in 2020 [2] and at the ICALP 2021 conference (track B) [3]. Since 2024, he is a vice-president of EATCS.*

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

Antoine: This is a selfie from a small hike near Lake Tahoe, which I did during my fall semester at the Simons Institute in 2023. This was my first overseas trip since COVID, so I wanted to make the most of it!

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

Antoine: I was passionate about computers as a kid – programming video games, hacking calculators, playing pranks on my classmates, etc. I was steered towards theory by the French education system, but for a long time I tried to resist – I was quite dismissive of teachers who called themselves computer scientists but couldn't even host their own mail server or compile their own kernel.

I eventually got interested in academic research during my internship with Pierre Senellart in 2012, but topic-wise I was still working on knowledge bases, in a futile attempt at practical relevance. It's only at the end of the internship that I embraced theory. I visited Fabian Suchanek at MPI Saarbrücken, and instead of knowledge bases I got nerdsniped by an abstract question of his. It was about tree labelings: if you label the vertices of a rooted tree with their preorder and postorder timestamp, then you can test in $O(1)$ if a node n_1 is a descendant of another node n_2, simply by retrieving their labels and checking if the interval formed by the timestamps of n_1 is included in that of n_2. Fabian observed that this approach could be generalized to certain DAGs, and I spent the visit trying to understand precisely for which DAGs this worked. Figuring it out was quite fun, and (I must admit) very rewarding in terms of ego. So I shifted towards database theory for my PhD with Pierre, and since then I have been continuously navigating towards the kind of theoretical questions that I like.

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

Antoine: Let me instead confess an embarrassing truth: I really don't read many papers in detail (except as a reviewer). I look at dozens of papers per month but only to skim them: I look for related work on topics that I'm interested in, I search for answers when I ask myself questions, I stay informed about what people in my area are working on, etc. But typically I don't read papers from beginning to end, unless it's a life-or-death necessity for the stuff I already want to do. My exposure to outside ideas mostly comes from talks and from people explaining their work in discussions.

Maybe it means I'm missing out on some things, but I'm not sure – I already feel like I'm stretched too thin... I also think we should be bolder and reach out to other researchers (at conferences or online) to chat about their work even if we haven't read it. Further, I suspect our papers are read by far fewer people than

we'd like – in the academic career game, you score points by writing papers, not reading them. But this doesn't mean that writing those papers isn't worthwhile!

EATCS: Is there a paper of your own you like to recommend the readers to study?

Antoine: It's hard to pick one, but I like our results in [4]. This paper is about enumerating the words of regular languages. Say we have a regular language L, and we want to write L as a (generally infinite) sequence of words: $w_1, w_2, \ldots$. We want to ensure that there is a uniform bound $c \in \mathbb{N}$ on the distance between any two consecutive words, i.e., each word w_{i+1} can be produced from the previous word w_i by changing at most c characters. For $L = a^*$ for instance you can simply write $\epsilon, a, aa, aaa, \ldots$. But for some languages it is impossible to construct such sequences, e.g., for $L = a^* + b^*$ (do you see why?). And sometimes we can build such sequences in non-obvious ways, e.g., for $L = (a + b)^*$ we can use a Gray code. Hence the question – for which regular languages do such sequences exist?

Our paper [4] gives a characterization of these languages based on a criterion on deterministic automata, which (much to our surprise) can be tested in PTIME. What I like about this paper is that I find the problem very easy to state, and the proofs are non-obvious but also do not use any sophisticated techniques. This makes me feel that theoretical CS is still a young field where you don't need many prerequisites to find fun questions to explore.

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

Antoine: I wish I had a recipe to be consistently productive! My ability to get the right things done varies a lot from one week to the next. My working hours have also changed a lot during my career. E.g., during my PhD I was a night owl – I would frantically write until around 2-3 AM, and on mornings I'd barely make it to the lab in time for lunch. Nowadays I have a more standard schedule.

That said, one thing I now understand is that good company is key for me to be productive and to have fun: I'm much more reliably focused and efficient when I work with others. This goes for research discussions, but also for writing – in a "pair programming" kind of way.

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

Antoine: Another confession: I spend very little time stuck on problems, because I'm very impatient and lose interest very quickly if I'm not making progress. At any given moment I have a lot of open directions on which it's feasible to move forward. So if I start thinking about a new question and run out of ideas to try, then I'll quickly move on to something else. The main exception is when I convinced myself (or, worse, convinced others) that a question should be solvable, but then the approach breaks – at this point I'll get very annoyed and work hard to save it.

I'm not sure my way of doing things is optimal, because it biases me towards easier questions – maybe I should learn to embrace failure more and spend more time on high-risk endeavors. Yet, I think it's reasonable to allocate one's efforts wisely, and work in priority on projects where you're in a good position to bring new techniques or new ideas and can reasonably hope to find a solution. It's true that you can sometimes break problems by relentless frontal attacks, but also it's often the case that you're just doing it wrong and the solution will have to come from elsewhere, e.g., using techniques from another field.

EATCS: How do you choose what to work on and what kind of impact are you hoping your work will have?

Antoine: Do we choose problems, or do problems choose us? I find there are some problems that immediately fascinate me [1] – they feel "natural". And I get more and more picky with time, and find it harder and harder to get interested in research unless I can see how it relates to the kind of problems I care about.

That said, in the past I have often been seduced by problems that seemed natural but turned out with hindsight to only make sense in a narrow community and at a specific time. So I try to shield my sense of naturalness from the appeal of trendy topics, by checking a few things: Can I explain in elementary terms what the question is and how we ended up there? Would the problem appeal to a more general audience? (Maybe not to random people on the street, but to a motivated undergrad, or to my friends with a CS/math background?) Is this problem the first question we should solve, or is there something simpler that we already don't understand?

In terms of impact, my hope is that I can contribute a little bit to the progress of theoretical research. I don't hope that my work will ever directly influence practical computing or even practical research, and I'm fine with that. Theory matters for its own sake – and I also believe it helps practice, but in indirect and global ways that are often hard to attribute to individual papers or researchers. I also think it's valuable and important to develop the use of advanced theory to solve practical problems, but personally I don't claim to do this – in my opinion this job must be done in collaboration with people who are actually getting their hands dirty to achieve something in the real world. I don't believe in papers that target hypothetical practical needs but didn't start as an attempt to solve a concrete problem – I find they are usually neither theoretically interesting nor practically relevant.

EATCS: What research topics or approaches do you wish the community did more of?

Antoine: I don't have specific topics in mind, but in terms of approaches I think TCS would benefit from a more open culture: we should be sharing our ideas

earlier and more broadly, instead of keeping them private until they can be posted as finished papers.

Here's how research currently seems to be done around me. You first find new directions to explore via discussions – with existing collaborators or with new people that you met at a conference or other event. Once the collaboration is started, everything happens in private, except maybe for some in-person gossiping at conferences or with colleagues, plus perhaps a few targeted questions sent by email to specific people. Nothing is visible to outside observers until the paper in finished and posted on arXiv.

All of this may have made sense in the pre-Internet age, but today it feels inefficient. Why aren't we publishing more things online than finished papers? To be clear: I'm not saying everything we do ought to be online, or that in-person discussions aren't valuable, or that major results shouldn't be written up as polished papers. But if only final results see the light of day, I think we're missing out, for several reasons. First, most projects just get postponed indefinitely. Lots of the time I spend on research ends up having zero tangible outcome because there's just not enough time to finish everything. So, if our research notes and drafts were in the open the entire time, then at least some usable trace of the effort would be available for others. Second, research discussions typically lead to many more open questions and ideas than what eventually appears in papers: why not post these questions (e.g., on `cstheory.stackexchange.com`) and see what it leads to? Third, online discussions are more inclusive than in-person conference gossip, and they scale better: they are open to everyone with a search engine, which includes researchers in other communities, people who don't often attend conferences, curious students you haven't met yet, etc. I think it's crucial that we reach out to people outside the core TCS community (not just the few students we directly supervise). So we should be making it easier for everyone to see what we are doing and get involved – and expensive onsite conferences and finished papers behind paywalls are not the best way to attract newcomers.

Overall, I think that much progress in research is made possible when ideas from different places can meet – and it's essentially impossible to anticipate what will have long-term impact and who it will inspire. So we should be throwing lots of our ideas around, to see what sticks. But currently we are blindly investing huge amounts of effort to write a small number of finished papers – faster and broader iteration only happens with our direct collaborators.

So why are we so hesitant about putting things online? Mostly for superficial reasons: we are worried about attracting parasitic co-authors, getting our ideas stolen, or embarrassing ourselves by posting something stupid or wrong. That said, I think it's mostly a question of social norms, given that so many successful Internet-native communities are open by default (e.g., Wikipedia, open-source projects...).

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

Antoine: I think our main challenge is that we rely too much on conferences – to validate results, and also to hand out community recognition. Conferences can be great, but we should attend them by choice, not by obligation. As it turns out, many of the historical functions of conferences have already been taken over by other channels: e.g., preprint servers (to make papers available), Google Scholar recommendations (to advertise them to relevant people), etc. But to get street cred as a TCS community member (and to make a living doing TCS), for now there is no alternative: you need papers in the big conferences. (Yes, I know about journals, but they operate on unpredictable timelines and are often less prestigious than conferences. So they are a not a good first option – at least in the common case when some co-authors are still looking for a permanent research position.)

The problem with most conferences is that they require in-person attendance. And sure, meeting people in-person is a fantastic opportunity when possible – in particular for new students. But I see two main problems with this culture of mandatory travel.

The first reason is inclusivity. Not everyone can easily travel to conferences, for various reasons: visa restrictions, lack of funding, health issues, childcare responsibilities... So, currently, we don't even see the many people that the conference publication system leaves out. (EATCS itself is a good example of this: people mostly join via in-person attendance to member conferences like ICALP.) I think our community should strive to include everyone who does TCS, not just the privileged subset of people who travel regularly.

The second reason is climate change. To reduce greenhouse gas emissions to sustainable levels, we must fly less – our initiative TCS4F [5] is a way to advocate for this. I'm passionate about TCS research and how it can focus on questions that will only bear fruit in the very long term; but I am also worried about our climate trajectory in the short term. Honestly, I'm not sure our research is worth the tons of CO_2 which are essentially required nowadays to build up an internationally competitive CV.

So here is what I think is our big challenge: decoupling community recognition and conference travel. I think it is also an opportunity: if we modernize our practices and make our community more accessible, we can stay relevant and attract talent that we would otherwise miss.

Please complete the following sentences?

- *Being a researcher is ... a job first and foremost. (At some point I thought it was a passion, but looking more closely: I do not see many hobbyists participating to the kind of research I do, and I see that we all spend a large part of our time on tasks that we would never do if we weren't professionally pressured to do them.)*
- *My first research discovery was ... complexity bounds on frequent itemset mining in taxonomies by posing queries on crowdsourcing platforms. Yes, my tastes have changed a bit since then.*
- *Enjoying research is ... not the easiest kind of pleasure in life!*
- *Theoretical computer scientists 100 years from now will ... possibly not exist anymore – unless we manage to do what's needed to steer our civilization towards a more sustainable trajectory (geopolitically, environmentally...)*

References

[1] A. Amarilli. List of open questions, 2025. `https://a3nm.net/work/research/questions/`.

[2] A. Amarilli and I. I. Ceylan. A dichotomy for homomorphism-closed queries on probabilistic graphs. In *ICDT*, 2020.

[3] A. Amarilli, L. Jachiet, and C. Paperman. Dynamic membership for regular languages. In *ICALP*, 2021.

[4] A. Amarilli and M. Monet. Enumerating regular languages with bounded delay. In *STACS*, 2023.

[5] TCS4F. Pledge for sustainable research in theoretical computer science, 2025. `https://tcs4f.org/`.

TCS ON THE WEB

BY

STEFAN NEUMANN

TU Wien
Erzherzog-Johann-Platz 1
1040 Vienna, Austria
stefan.neumann@tuwien.ac.at
https://neumannstefan.com

TCS+[1] is an online seminar series in theoretical computer science, existing since 2013, and by now there have been almost 200 talks. The seminar consists of online talks about the most important new results in algorithms and complexity, and also makes the talks available on YouTube[2]. Digging through the TCS+ YouTube channel is a real treasure trove of great talks and highly influential results of the past decade.

Recently, there have been a few changes to TCS+. Some of the time slots of the talks have been adjusted to better suit European audiences. Now, the talks usually commence at either 16:30 and 19:00 CET, and after the talks, there are more open discussions with the speaker.

Today I am more than happy to talk to two of the current TCS+ organizers, Rachel Cummings (Columbia University) and Sumegha Garg (Rutgers University). The other current organizers are Clément Canonne (University of Sydney), Shivam Nadimpalli (MIT), Yasamin Nazari (VU Amsterdam), and Erik Waingarten (University of Pennsylvania).

[1] https://sites.google.com/view/tcsplus/welcome
[2] https://www.youtube.com/@TCSplusSeminars

Behind the Scenes of TCS+: Talks, Research, and Community Efforts

A Conversation with Rachel Cummings and Sumegha Garg

Rachel and Sumegha, thank you for taking time for this interview. I think TCS+ is a great format that a lot of people in the community appreciate due to its timely and interesting topics. How do you decide which results you want to cover in TCS+ talks? Do you bias towards big new results or towards very active areas?

Rachel: We have a web form[1] where anyone can suggest papers, and we encourage people to use it—we'd love to get more suggestions! Organizers also add their own recommendations. We gather all the suggestions into a list and vote on them. We do have a bias towards topics that would be of broader interest rather than specific niche topics.

Sumegha: Exactly. The diversity of our organizing team also helps. Since we come from different research areas, we can make sure our selections appeal to a wide range of people. If a talk gets votes from across the team, it's usually a good sign it'll resonate broadly. We also keep an eye on accepted papers at recent theory conferences to identify impactful papers that haven't yet been suggested.

Rachel: Another factor is the speaker. We prioritize speakers known to give high-quality, engaging talks because that makes a huge difference, especially in an online format. If no one on the team has seen a speaker before, we'll check their previous talks online to make sure they'll connect well with the audience.

TCS+ has been around since 2013, long before the pandemic forced us all into online talks. What's changed over the years?

Rachel: Oh, a lot! I remember attending TCS+ talks as a grad student. Back then, it was hosted on Google+, which is where the "+" in TCS+ comes from. Google+ was like an early version of Zoom, but without as many features.

Sumegha: Right, and only 10–12 people could join a session because of platform limitations. So universities would organize watch parties in their conference rooms. Groups would gather, project the talk onto a screen, and watch together.

[1] `https://sites.google.com/view/tcsplus/welcome/suggest-a-talk`

Rachel: I have really fond memories of that era. I was a student at Caltech then, and the talks were in the morning for our timezone. We'd meet in a conference room, bring coffee and bagels, and watch the talks as a group. It felt special—a mix of learning and community building. But it was also competitive! Since only a few schools could join, there was a waitlist. If your school couldn't make it one week, another would take your slot.

Sumegha: Things are much more open now with Zoom. Anyone can join, which is great for accessibility. But we don't see as many of those group watch parties anymore, which is a bit nostalgic.

Rachel: We also upload all of our talks on YouTube[2] and so that people can watch them any time. Even if live attendance on Zoom has dipped, we get hundreds of views within a day of posting a new talk on YouTube.

It sounds like the format has evolved a lot. Have there been any challenges?

Sumegha: Definitely. During the pandemic, participation was high, but later, we noticed Zoom fatigue setting in. To address this, we've started encouraging more interactive discussions after the talks, which aren't recorded. It helps bring back some of that personal engagement.

Rachel: We've also adjusted the timing of some talks to make them more accessible to European audiences. Traditionally, our talks were in the evening for Europe, which isn't always convenient. Now we alternate between our usual time and an earlier slot to accommodate more people. The early talks start at 16:30 CET.

How did you both get involved in organizing TCS+?

Sumegha: I joined in 2020. I got an email from one of the organizers, and I was really excited to see how everything worked behind the scenes. At first, I focused on backend tasks, but over time, I became more involved in planning and organizing talks.

Rachel: For me, it was around 2021. Organizers are invited based on their reliability and expertise, and we try to ensure a good mix of research areas. People rotate in and out, depending on their availability, which keeps the team dynamic and diverse.

TCS+ is aimed at a technical audience. Do you see opportunities to engage broader communities, like applied researchers or mathematicians?

Rachel: Absolutely. Admittedly, TCS+ does focus on the TCS community, as our talks are really aimed at that audience. There's tremendous value in interdisciplinary engagement across academic communities or with practitioners, as

[2] `https://www.youtube.com/@TCSplusSeminars`

we've seen in the EconCS and CS+Law communities. Engaging broader audiences can help facilitate these connections, but requires a different approach. For example, talks should use less technical jargon and provide more background in order to be accessible to a broader audience who may be working with a different professional vocabulary.

Sumegha: Even within TCS, we try to make talks accessible. For example, we include one or two survey talks each semester, which are more about teaching a topic than presenting the latest advances. It's a way to help researchers learn about areas outside their specialization.

Can you tell us a bit about your research?

Rachel: My work focuses on differential privacy, which is a method for protecting individual privacy in data analysis. The goal is to ensure that changing one person's data in a dataset doesn't significantly alter the results. This usually involves adding random noise to the process.

I have two main areas of focus. On the theoretical side, I design and analyze privacy-preserving algorithms for tasks like optimization and statistics. On the practical side, I think about how to implement these algorithms in real-world settings. This includes translating complex privacy parameters into terms organizations care about, like compliance or risk mitigation, and communicating the privacy guarantees to end users.

Sumegha: My background is in computational complexity theory. I study models of computation with memory or information constraints, such as streaming algorithms or distributed computing. Recently, I've been exploring how memory limitations affect learning or estimation tasks, like online learning or regret minimization. Understanding the memory requirements of these tasks is becoming theoretically and practically important.

Any final thoughts you would like to share with our readers?

Rachel: We'd love to see more people attend talks and suggest topics for TCS+! Engagement is key to keeping this format alive and relevant.

Sumegha: Yes, please use our web form[3] to suggest speakers or topics you're excited about. It's a great way to make sure the talks reflect what the community cares about.

Thank you for this nice interview, Rachel and Sumegha!

[3] `https://sites.google.com/view/tcsplus/welcome/suggest-a-talk`

The Algorithmics Column

BY

Ioana O. Bercea and Thomas Erlebach

KTH, Stockholm, Sweden and Durham University, UK
bercea@kth.se and thomas.erlebach@durham.ac.uk

BEATCS no 145

Data Compression Meets Automata Theory

Nicola Cotumaccio
University of Helsinki, Finland
nicola.cotumaccio@helsinki.fi

Abstract

I received my PhD in Computer Science on January 31, 2024, under a Joint PhD agreement between Gran Sasso Science Institute (L'Aquila, Italy) and Dalhousie University (Halifax, Canada). I was supervised by Travis Gagie, Nicola Prezza and Catia Trubiani. My PhD thesis, *Data Compression Meets Automata Theory*, was selected by the Italian Chapter of the EATCS for the Best PhD Thesis Award.

The thesis introduces a new paradigm for studying regular languages, establishing a connection between classical results in automata theory, such as the powerset construction, and the most important data structures for solving pattern matching queries on compressed strings, such as the Burrows-Wheeler transform. The results and the open problems should be of interest to both the algorithmic community and the formal language theory community.

In 2020, Alanko et al. introduced *Wheeler automata* [3]. Intuitively, a nondeterministic finite automaton (NFA) is Wheeler if there exists a total order on the set of all nodes that is consistent with the co-lexicographic order on the strings reaching each node (see Figure 1). We say that a regular language is Wheeler if it is recognized by some Wheeler NFA. Wheeler automata were initially motivated by classical results on compressed data structures [18] but, as a matter of fact, the class of Wheeler languages is a surprisingly rich and stable subclass of regular languages. Basic results in automata theory show that, if a language is recognized by a nondeterministic finite automaton (NFA), then the language is also recognized by a deterministic finite automaton (DFA), and up to isomorphism there exists a unique (state)-minimal DFA recognizing the language. Moreover, regular languages admit an algebraic characterization in terms of equivalence relations, and the minimal automaton can be described through the Myhill-Nerode equivalence. Analogous results hold for Wheeler languages: determinism and non-determinism have the same expressive power, there exists a unique minimal Wheeler DFA, and there exists a characterization in terms of *convex* equivalence relations [4].

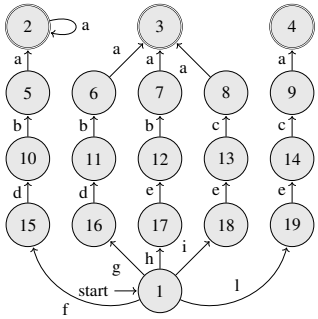


Figure 1: A Wheeler automaton. States are numbered according to their positions in the corresponding total order. Node 2 is reached by the string $fdba$, node 3 is reached by the string $gdba$, we have $2 < 3$ and consistently the string $fdba$ is co-lexicographically smaller than $gdba$.

Most automata are not Wheeler: every Wheeler language must be star-free. In [13], we show how to extend the idea behind Wheeler automata to *arbitrary* automata and *arbitrary* regular languages. The key idea is that we should not use a total order, but a *partial order* (see Figure 2): in this way, it is possible to capture every automaton. We can measure how far a partial order is from being a total order by the notion of *width*: a partial order has width p is it can be decomposed into p total orders, but not into $p-1$ total orders. Dilworth's theorem [14] shows that the width of a partial order is equal to the size of a largest set of pairwise incomparable elements. Wheeler automata correspond to the case $p = 1$.

The parameter p is a complexity parameter with multiple interpretations and applications.

- First, p is a nondeterminism parameter. The powerset construction [23] is a classical construction that converts an NFA into an equivalent DFA (see Figure 3). If the original NFA has n states, in the worst case the equivalent DFA can have up to $2^n - 1$ states. This exponential blow-up is unavoidable: there exist regular languages for which, if N is the number of states of a state-minimal NFA recognizing the language and D is the number of states of the minimal DFA for the language, then $D = 2^N - 1$ [20]. In the paper, we show that, in fact, the powerset construction is exponential only in p: if we start from an NFA with n states, then the equivalent DFA has at most $2^p(n-p+1)-1$ states. In the worst case, we have $p = n$ and we retrieve the bound $2^n - 1$ but, for example, a Wheeler NFA ($p = 1$) with n states can be converted into an equivalent DFA with at most $2n - 1$ states. This implies that several classical algorithmic problems that are computationally hard on

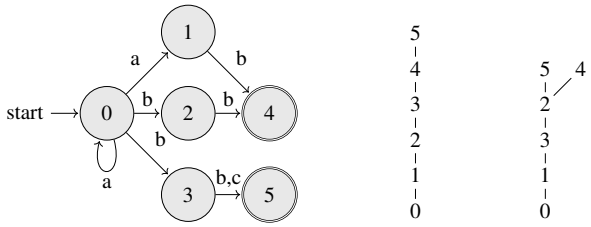


Figure 2: An automaton with the Hasse diagrams of two partial orders (on the set of all states), both respecting the co-lexicographic order on the strings reaching each state. Note that the first partial order is, in fact, a total order.

NFAs but easy on DFA are fixed-parameter tractable with respect to p. For example, the problem of deciding whether two NFAs recognize the same languages (*equivalence problem*) is PSPACE-complete [25], but the same problem can be solved efficiently if the input automata are DFAs. This implies that the equivalence problem is fixed-parameter tractable with respect to p, because one can convert the input NFAs into equivalent DFAs by the powerset construction and then test the resulting DFAs for equivalence.

- Second, p is a compression parameter. An NFA with n states and e edges on an alphabet of size σ can be stored by using only $e(2\log p + \log\sigma)(1 + o(1)) + O(e)$ bits. This is surprising because at the very least we need $e\log\sigma$ bits to store the edge labels, so with a small overhead we can also store the topology of the automaton. This result extends the Burrows-Wheeler transform [6] from strings to arbitrary automata.

- Third, p is an algorithmic parameter. By only querying our compressed representation, we can solve pattern matching on an NFA (including deciding whether a string in accepted by the NFA) in $O(mp^2\log\log(p\sigma))$ time, where m is the length of the pattern. Our algorithm matches and parameterizes well-known conditional lower bounds on the problem [15]. This result extends the FM-index [17] from strings to arbitrary automata.

Since we can capture all automata, we can also capture all regular languages. In particular, we can define the deterministic width of a regular language as the minimum width of some DFA that recognizes the language. In our journal extension [8], we show that the width of a regular language is related to the notion of *entanglement*: intuitively, some states of an automaton (and in particular of the minimal automaton) are entangled if the strings reaching these states are inherently incomparable in *every* DFA recognizing the language. Based on the notion of entanglement, we define a new canonical automaton for each regular language,

the *Hasse automaton* of the language, and we show that the problem of computing the deterministic width of a language is decidable. The Hasse automaton captures the propensity of a regular language to be sorted.

The journal extension [8] contains the main ideas of the thesis. A short initial section presents the new paradigm; then, the readers can skip the automata theory results or the data compression results, based on their interests and preferences.

Computing p is NP-hard. In [10], I show that the hardness can be overcome by building a quotient automaton obtained by collapsing some states in the original automaton. The quotient automaton and the original automaton recognize the same language, and some correspondence theorems ensure that solving pattern matching queries on the original automaton is equivalent to solving pattern matching queries on the quotient automaton. This paper received the Best Student Paper Award at the 2022 Data Compression Conference (DCC).

In the deterministic case, the problem of computing p can be interpreted as a highly non-trivial extension of the problem of computing the suffix array of a string, a well-studied problems in string processing (see [21] for a survey). In the original paper [13] we gave an $O(m^2)$ algorithm, where m is the number of edges. In [11], I improve this running time to $O(m + n^2)$, where m is the number of edges and n is the number of states, by proposing a recursive algorithm based on induced sorting (a powerful and elegant algorithmic technique for sorting the suffixes of a string). This paper received the Best Student Paper Award at the 2023 International Symposium on Algorithms and Computation (ISAAC). Interestingly, it is an open problem to determine whether it is possible to achieve $O(m)$ time. There exists an alternative algorithm based on Paige and Tarjan's partition refinement algorithm [22], and the $O(m + n^2)$ algorithm suggests that induced sorting may outperfom approaches based on partition refinement. This could lead to unexpected consequences, because the most efficient algorithm for DFA minimization is still Hopcroft's algorithm [19], which is a partition refinement algorithm.

The same paradigm can be extended to more general computation models. In his monumental work on automata theory [16], Eilenberg proposed a natural generalization of NFAs where edges can be labeled not only with characters but with finite strings, the so-called *generalized automata*. String-labeled graphs appear in some classical data structures (such as suffix trees) and in the emerging field of pangenomics. In [12], I show that our new paradigm can be extended to generalized automata, and I describe a full Myhill-Nerode theorem, the first structural result for the class of generalized automata, which includes a sound notion of minimal DFA for generalized automata.

As mentioned earlier, Wheeler languages always admit a minimal Wheeler automata. A Wheeler DFA can be minimized in $O(n \log n)$ time by adapting Hopcroft's algorithm for DFA minimization. In [2], we show that, in the Wheeler case, we can do better: it is possible to achieve linear time minimization by ex-

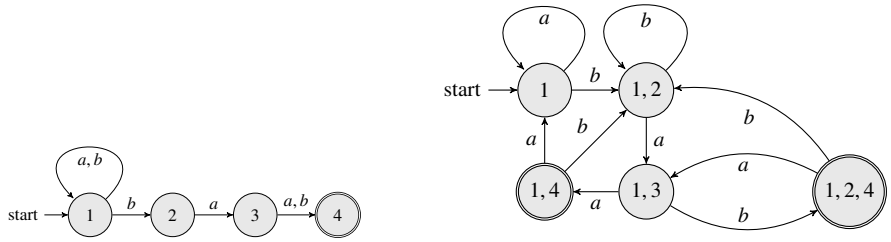


Figure 3: An NFA (left) and the equivalent DFA obtained by the powerset construction (right). Each state of the DFA corresponds to a nonempty set of states of the original NFA.

ploiting the co-lexicographic structure of a Wheeler DFA.

If we want to solve more complicated pattern matching queries on graphs and automata (for example, approximate pattern matching queries) we need a more powerful data structure. In the string setting, the most versatile data structure is the suffix tree [26], so we should extend suffix tree functionality to automata. In a chain of papers [7, 9, 1], we provide some partial results in this direction by showing how to extend the longest common prefix array to Wheeler DFAs. The longest common prefix array is a key data structure for simulating a traversal of a suffix tree [5].

In the last chapter of my thesis, I discuss some additional open problems. We outlined how Wheeler languages enjoy several properties, but two characterizations are missing: one in terms of regular expressions, and one through logic. Wheeler languages are star-free, so they capture a fragment of first-order logic: to obtain a logical characterization it may be necessary to go through difficult results such as the Schützenberger theorem for aperiodic monoids [24]. At the same time, the thesis introduces a very natural hierarchy of regular languages parameterized by p, so by exploring these characterizations and extending them to each level of the hierarchy we may shed new light on famous important open problems, such as the generalized star-height problem.

References

[1] Jarno N. Alanko, Davide Cenzato, Nicola Cotumaccio, Sung-Hwan Kim, Giovanni Manzini, and Nicola Prezza. Computing the LCP Array of a Labeled Graph. In Shunsuke Inenaga and Simon J. Puglisi, editors, *35th Annual Symposium on Combinatorial Pattern Matching (CPM 2024)*, volume 296 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 1:1–1:15,

Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[2] Jarno Alanko, Nicola Cotumaccio, and Nicola Prezza. Linear-time minimization of Wheeler DFAs. In *2022 Data Compression Conference (DCC)*, pages 53–62, 2022.

[3] Jarno Alanko, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. Regular languages meet prefix sorting. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, page 911–930, USA, 2020. Society for Industrial and Applied Mathematics.

[4] Jarno Alanko, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. Wheeler languages. *Information and Computation*, 281:104820, 2021.

[5] Mohamed Ibrahim Abouelhoda, Stefan Kurtz, and Enno Ohlebusch. Replacing suffix trees with enhanced suffix arrays. *Journal of Discrete Algorithms*, 2(1):53–86, 2004. The 9th International Symposium on String Processing and Information Retrieval.

[6] Michael Burrows and David J Wheeler. A block-sorting lossless data compression algorithm. Technical Report 124, Digital Equipment Corporation, 1994.

[7] Alessio Conte, Nicola Cotumaccio, Travis Gagie, Giovanni Manzini, Nicola Prezza, and Marinella Sciortino. Computing matching statistics on Wheeler DFAs. In *2023 Data Compression Conference (DCC)*, pages 150–159, 2023.

[8] Nicola Cotumaccio, Giovanna D'Agostino, Alberto Policriti, and Nicola Prezza. Co-lexicographically ordering automata and regular languages-part i. *Journal of the ACM*, 70(4):1–73, 2023.

[9] Nicola Cotumaccio, Travis Gagie, Dominik Köppl, and Nicola Prezza. Space-time trade-offs for the LCP array of Wheeler DFAs. In Franco Maria Nardini, Nadia Pisanti, and Rossano Venturini, editors, *String Processing and Information Retrieval*, pages 143–156, Cham, 2023. Springer Nature Switzerland.

[10] Nicola Cotumaccio. Graphs can be succinctly indexed for pattern matching in $O(|E|^2+|V|^{5/2})$ time. In *2022 Data Compression Conference (DCC)*, pages 272–281, 2022.

[11] Nicola Cotumaccio. Prefix Sorting DFAs: A Recursive Algorithm. In Satoru Iwata and Naonori Kakimura, editors, *34th International Symposium on Algorithms and Computation (ISAAC 2023)*, volume 283 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:15, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[12] Nicola Cotumaccio. A Myhill-Nerode Theorem for Generalized Automata, with Applications to Pattern Matching and Compression. In Olaf Beyersdorff, Mamadou Moustapha Kanté, Orna Kupferman, and Daniel Lokshtanov, editors, *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*, volume 289 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:19, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[13] Nicola Cotumaccio and Nicola Prezza. On indexing and compressing finite automata. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2585–2599. SIAM, 2021.

[14] RP Dilworth. A decomposition theorem for partially ordered sets. *The Annals of Mathematics*, 51(1):161, 1950.

[15] Massimo Equi, Roberto Grossi, Veli Mäkinen, and Alexandru I. Tomescu. On the Complexity of String Matching for Graphs. In Christel Baier, Ioannis Chatzigiannakis, Paola Flocchini, and Stefano Leonardi, editors, *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 55:1–55:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[16] Samuel Eilenberg. *Automata, Languages, and Machines*. Academic Press, Inc., USA, 1974.

[17] Paolo Ferragina and Giovanni Manzini. Indexing compressed text. *J. ACM*, 52(4):552–581, July 2005.

[18] Travis Gagie, Giovanni Manzini, and Jouni Sirén. Wheeler graphs: A framework for BWT-based data structures. *Theoretical Computer Science*, 698:67–78, 2017. Algorithms, Strings and Theoretical Approaches in the Big Data Era (In Honor of the 60th Birthday of Professor Raffaele Giancarlo).

[19] John Hopcroft. An n log n algorithm for minimizing states in a finite automaton. In *Theory of machines and computations*, pages 189–196. Elsevier, 1971.

[20] F.R. Moore. On the bounds for state-set size in the proofs of equivalence between deterministic, nondeterministic, and two-way finite automata. *IEEE Transactions on Computers*, C-20(10):1211–1214, 1971.

[21] Simon J. Puglisi, W. F. Smyth, and Andrew H. Turpin. A taxonomy of suffix array construction algorithms. *ACM Comput. Surv.*, 39(2):4–es, July 2007.

[22] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[23] M. O. Rabin and D. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[24] M.P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.

[25] L. J. Stockmeyer and A. R. Meyer. Word problems requiring exponential time(preliminary report). In *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing*, STOC '73, page 1–9, New York, NY, USA, 1973. Association for Computing Machinery.

[26] Peter Weiner. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 1–11. IEEE, 1973.

THE DISTRIBUTED COMPUTING COLUMN

Seth Gilbert
National University of Singapore
seth.gilbert@comp.nus.edu.sg

This month, in the Distributed Computing Column, Francesco d'Amore surveys some recent exciting results in quantum distributed computing, focusing on the question of when a distributed quantum advantage is possible. That is, imagine you have a network of quantum computers connected via quantum communication channels; when can such a network solve a problem faster than in the classical LOCAL model? While general versions of the questions remain open, there has been significant and compelling progress in the last several years.

To answer this question of quantum advantage, d'Amore begins by examining the fundamental principles that enable fast quantum distributed computation, e.g., issues related to causality and independence of output distributions. He explores the landscape of intermediate models, including those both stroger and weaker than the "quantum-LOCAL" model, including discussions of "non-signalling" models, "bounded dependence" models, and online local models. These models provide key technical tools for excluding quantum advantage—or showing when it may help. The survey then describes a recently discovered concrete example of a local problem that is provably faster in the quantum models: it can be solved in $O(1)$ time in quantum-LOCAL, but requires $\Omega(\Delta)$ time in classical LOCAL (where Δ is the maximum degree of the network graph).

Overall, then, this article provides a comprehensive overview of the state-of-the-art for quantuam distributed algorithms today, along with some nice insight into the tools and techniques needed to understand the performance of these algorithms.

The Distributed Computing Column is particularly interested in contributions that propose interesting new directions and summarize important open problems in areas of interest. If you would like to write such a column, please contact me.

BEATCS no 145

ON THE LIMITS OF DISTRIBUTED QUANTUM COMPUTING

Francesco d'Amore*

*Bocconi University, BIDSA, Italy

Abstract

Quantum advantage is well-established in centralized computing, where quantum algorithms can solve certain problems exponentially faster than classical ones. In the distributed setting, significant progress has been made in bandwidth-limited networks, where quantum distributed networks have shown computational advantages over classical counterparts. However, the potential of quantum computing in networks that are constrained only by large distances is not yet understood. We focus on the LOCAL model of computation (Linial, FOCS 1987), a distributed computational model where computational power and communication bandwidth are unconstrained, and its quantum generalization. In this brief survey, we summarize recent progress on the quantum-LOCAL model outlining its limitations with respect to its classical counterpart: we discuss emerging techniques, and highlight open research questions that could guide future efforts in the field.

1 Introduction

Since the advent of quantum computing, extensive research has been conducted to explore its potential, revealing its advantage over classical computing, at least from a theoretical point of view: there are problems that classical algorithms solve in super-polynomial time, while quantum algorithms can solve them in polynomial time [Aar22]. But what about *distributed* quantum advantage?

There is a large body of research investigating this question in *bandwidth-limited* networks [Cen+22; LM18; WY22; WWY21; IL19; ILM20; AV22; Fra+24; MN22]. Such networks are captured by (possibly variants of) the CONGEST model of computing. In essence, the question is: can a synchronous network of quantum computers that send b quantum qubits per time unit to each neighbor outperform a synchronous network of classical machines that send b classical bits

per time unit? It turns out that in many cases, the answer is *yes*. There are computational tasks that are asymptotically easier to solve in the quantum-CONGEST model (and related variants): see, e.g., [IL19; ILM20; LM18].

However, if we consider instead networks that are constrained only by *large distances*, the scenario is much less understood. Such networks typically model distributed systems where network latency and the time required for information to propagate play key roles. We emphasize that large distances are a fundamental physical limitation of distributed networks: information, whether classical or quantum, cannot travel faster than the speed of light. This limitation cannot be overcome by technological progress, unlike bandwidth constraints, which can benefit from innovations (say, the installation of multiple parallel communication channels by, e.g., increasing the number of fiber-optic links between nodes).

Such distance-constrained networks are modeled by the famous LOCAL model of computation, first introduced in the seminal work by Linial [Lin87] (see [Lin92] for the journal version). In the LOCAL model, a distributed network is represented as a graph $G = (V, E)$, where the nodes of G are processors capable of *unbounded local computation*, and the edges represent communication links. Time proceeds synchronously: in each round, nodes send and receive messages of *arbitrarily large size* from their neighbors and perform local computations to update their state variables. Eventually, all nodes announce their outputs, marking the end of the computation. The complexity measure in this model is the number of communication rounds required to solve a problem, captured by the notion of *locality*. Specifically, T rounds of communication allow a node to gather information about the topology and input within its radius-T neighborhood. Thus, T is also called the *locality* of the algorithm, reflecting why distances are the only limitation in this model.

The LOCAL model has been extensively studied [Lin92; Nao91; NS95; CKP16; CKP19; GKM09; CP19; GHK18; FG17; GKM17; Bra+16; Hir+17; KMW04; KSV13]. A specific class of problems of particular interest in distributed computing is *locally checkable labeling* (LCL) problems, introduced by Naor and Stockmeyer [NS95]. LCL problems are defined via local constraints (e.g., graph coloring). For LCL problems, a solution may be *hard to find* but is *easy to verify* with a distributed algorithm. As such, LCL problems can bee seen as the distributed analogue of the FNP class in centralized computation. Nowadays, we have a good understanding of complexity landscape of LCL problems in the LOCAL model [NS95; CKP19; CKP16; CP19; Bal+18; Bal+19a; Bal+19b; Bal+21a; Bal+20; Bal+21b; Bal+22b; Bal+22a; Bal+23a; Bal+23b; Bra+17; Bra19; Akb+23].

To date there is no clear understanding of the impact of quantum computation and communication in the LOCAL model, especially concerning LCL problems. The main challenge lies in the lack of tools for directly tackling quantum-LOCAL, particularly for establishing lower bounds. Nevertheless, general arguments based

on physical principles, such as *causality* and *independence* of output distributions, provide some insights.

In this brief survey, we summarize previous knowledge and recent results, introducing new techniques for investigating the role of quantum computation and communication in distributed settings and shedding light on potential research directions.

2 Preliminaries

In order to proceed, we need to provide the mathematical framework in which we work. We start with some basic graph notations and definitions, introducing the class of problems we consider and the computational model that is the heart of our investigation. Section by section, we will give other definitions that are useful for that section and the subsequent ones. We denote the set of natural numbers (starting from 0) by $\mathbb{N}$, and also define $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$.

Graphs. We work with simple graphs unless otherwise specified. Given any set S, we denote by $\binom{S}{k}$ the set whose elements are all sets of k different elements of S. We will consider both undirected and directed graphs. A graph is a pair $G = (V, E)$ where V is the set of nodes and E is the set of edges. In case of undirected graphs, $E \subseteq \binom{V}{2}$, whereas $E \subseteq V \times V$ in case of directed graphs. For any graph G, we also denote its set of nodes by $V(G)$ and its set of edges by $E(G)$.

The distance between two nodes u, v of any graph G is the number of edges in any shortest path between u and v (note that the shortest path is not necessarily an oriented path), and is denoted by $\mathrm{dist}_G(u, v)$. The notion of distance can be easily extended to subset of nodes: Given any node $u \in V$ and any two subsets $A, B \subseteq V$, the distance between u and A is $\mathrm{dist}_G(u, A) = \min_{v \in A}\{\mathrm{dist}_G(u, v)\}$ and the distance between A and B is $\mathrm{dist}_G(A, B) = \min_{u \in A, v \in B}\{\mathrm{dist}_G(u, v)\}$. When the graph is clear from the context, we omit the suffix and write only dist() instead of $\mathrm{dist}_G()$. Through the notion of distance, we define the diameter of a graph G to be $\mathrm{diam}(G) = \max_{u,v \in V(G)} \mathrm{dist}_G(u, v)$.

For any non-negative integer T, the radius-T (closed) neighborhood of a node u in a graph G is the set $\mathcal{N}_T[u] = \{v \in V \mid \mathrm{dist}(u, v) \leq T\}$. Throughout this survey, we will only make use of closed neighborhoods. More in general, the radius-T neighborhood of any subset of node $A \subseteq V$ is $\mathcal{N}_T[A] = \cup_{u \in A} \mathcal{N}_T[u]$. We also define the *ring neighborhood* between T_1 and T_2 of a node $u \in V(G)$ (for $T_1 \leq T_2$) as $\mathcal{N}_{T_2}^{T_1}[u] = \mathcal{N}_{T_2}[u] \setminus \mathcal{N}_{T_1}[u]$. We use an analogous notation for subsets of nodes.

The degree of a node v in an undirected graph G is the number of edges the node belongs to, i.e., $\deg_G(v) = |\{\{u, v\} \in E(G) \mid u \in V(G)\}|$. In a directed graph G, we define the *indegree* and the *outdegree* of a node v as follows: $\mathrm{indeg}_G(v) =$

$|\{(u,v) \in E(G) \mid u \in V(G)\}|$ and $\text{outdeg}_G(v) = |\{(v,u) \in E(G) \mid u \in V(G)\}|$. Then the degree of v in G is just $\deg_G(v) = \text{indeg}_G(v) + \text{outdeg}_G(v)$. In all these notations, we omit the suffix when the graph is clear from the context. Finally, the degree of a graph G is just the maximum degree of any node, i.e., $\deg(G) = \max_{v \in V(G)}\{\deg_G(v)\}$.

For any graph $G = (V, E)$ and any subset of nodes $A \subseteq V$, the subgraph of G induced by A is denoted by $G[A]$. Consider any node $u \in V$ (or any subset $S \subseteq V$): with an abuse of notation, we define the *open induced subgraph* as the set $\mathring{G}[\mathcal{N}_T[u]] = G[\mathcal{N}_T[u]] \setminus G[\mathcal{N}_T^{T-1}[u]]$. (or $\mathring{G}[\mathcal{N}_T[S]] = G[\mathcal{N}_T[S]] \setminus G[\mathcal{N}_T^{T-1}[S]]$):. In practice, in this definition we are removing from the classical notion of neighborhood the edges that connect nodes that are at distance T from u (or S) as the graph that u (or S) *sees* by moving T hops away does not include them.

Now we define some graph operations. Given any two graphs G and H, the intersection of G and H is the graph $G \cap H = (V(G) \cap V(H), E(G) \cap E(H))$. The union of G and H is the graph $G \cup H = (V(G) \cup V(H), E(G) \cup E(H))$, while the difference between G and H is the graph $G \setminus H = (V(G) \setminus V(H), E(G) \setminus E(H))$.

An isomorphism between two graphs G and H is a function $\varphi : V(G) \to V(H)$ that is bijective and such that $\{u, v\} \in E(G)$ if and only if $\{\varphi(u), \varphi(v)\} \in E(H)$. In case of directed graphs, we also require the isomorphism to keep edge orientation.

Labeling problems. In this brief survey we consider labeling problems, namely, graph problems that ask to output some labels on the nodes of the graph. A formal definition follows.

Definition 2.1 (Labeling problem)**.** Let $\Sigma_{\text{in}}, \Sigma_{\text{out}}$ be two alphabets, and I a set of indices. A labeling problem Π is a mapping $(G, \text{in}) \mapsto \{\text{out}_i\}_{i \in I}$ that maps every input graph $G = (V, E)$ where nodes are labelled by any input function $\text{in} : V \to \Sigma_{\text{in}}$ to a family of suitable output functions $\text{out}_i : V \to \Sigma_{\text{out}}$ indexed by I. The mapping is closed under graph isomorphism, that is, for any isomorphism $\varphi : V(G) \to V(H)$ between two graphs G and H, $\text{out} \in \Pi((H, \text{in}))$ if and only if $\text{out} \circ \varphi \in \Pi((G, \text{in} \circ \varphi))$.

The reader may notice that we defined labeling problems for *any input graph*, despite the fact that some labeling problems might be defined only for some specific graph family like, for example, 3-coloring bipartite graphs. However, Definition 2.1 is general enough to capture all such problems, because one can just say that, for graphs that are outside the right graph family (e.g., non-bipartite graphs), all outputs are admissible. Examples of graph labeling problems are leader election, consensus, diameter approximation, etc.

A subclass of labeling problems that is of particular interest in the distributed computing community is that of locally checkable labeling (LCL) problems, already mentioned in the introduction. LCL problems were first introduced by Naor

and Stockmeyer [NS93] (for the journal version we refer to [NS95]). Here, we report the original definition.

For any function $f : A \to B$ and any subset $S \subseteq A$, we denote the restriction of f to S by $f \restriction_S : S \to B$. We define a *centered graph* to be any pair (G, v_G), where G is a graph and $v_G \in V(G)$ is a node of G that we call the *center* of G. The *radius* of a centered graph (G, v_G) is the maximum distance between v_G and any other node of G. We are now ready to state the definition of LCL problems.

Definition 2.2 (Locally Checkable Labeling problem)**.** Let r, Δ be non-negative integers. Let $\Sigma_{\text{in}}, \Sigma_{\text{out}}$ be two finite alphabets, and I a finite set of indices. Consider a labeling problem Π defined on $\Sigma_{\text{in}}, \Sigma_{\text{out}}, I$. Π is *locally checkable* with checking radius r and maximum degree Δ if there exists a family $\mathcal{S} = \{((H, v_H), \text{in}, \text{out})_i\}_{i \in I}$ where each tuple $((H, v_H), \text{in}, \text{out})_i$ contains a centered graph (H, v_H) of radius at most r and degree at most Δ, an input labeling function $\text{in} : V(H) \to \Sigma_{\text{in}}$ and an output labeling function $\text{out} : V(H) \to \Sigma_{\text{out}}$ with the following property:

- For every input (G, in) to Π with $\deg(G) \leq \Delta$, an output vector $\text{out} : V(G) \to \Sigma_{\text{out}}$ is admissible (i.e., $\text{out} \in \Pi((G, \text{in}))$) if and only if, for each node $v \in V(G)$, the tuple $\big((G[\mathcal{N}_T[v]], v), \text{in} \restriction_{\mathcal{N}_T[v]}, \text{out} \restriction_{\mathcal{N}_T[v]}\big)$ belongs to $\mathcal{S}$.

$\mathcal{S}$ is also called the family of *permissible outputs*, and is finite (up to graph isomorphisms) since Δ, r are finite and also $\Sigma_{\text{in}}, \Sigma_{\text{out}}$ are finite sets. Notice that in Definitions 2.1 and 2.2 we assumed that input and output labels are given to and from nodes. We might similarly assume that they are given to and from edges, and come up with other problems. In this survey, we will interchangeably use both possibilities when it is convenient. LCL problems capture all problems defined via local constraints, e.g., graph coloring, maximal independent set, maximal matching, sinkless orientation, triangle freeness, etc.

We now introduce the models of computations we are interested in. We begin with the port-numbering model, and on top of it we define the LOCAL model of computation [Lin92].

The port-numbering model. A port-numbered network is a triple $N = (V, P, p)$ where V is the set of nodes, P is the set of *ports*, and $p : P \to P$ is a function specifying connections between ports. Each element $x \in P$ is a pair (v, i) where $v \in V$, $i \in \mathbb{N}_+$. The connection function p between ports is an involution, that is, $p(p(x)) = x$ for all $x \in P$. If $(v, i) \in P$, we say that (v, i) is port number i in node v. With an abuse of notation, we say that the degree of a node v in the network N is the number of ports in v and is denoted by $\deg_N(v)$. We assume that port numbers are consecutive, i.e., the ports of any node $v \in V$ are $(v, 1), \ldots, (v, \deg_N(v))$. Clearly, a port-numbered network identifies an *underlying graph* $G = (V, E)$ where, for any two nodes $u, v \in V$, $\{u, v\} \in E$ if and only if there

exists ports $x_u, x_v \in P$ such that $p(x_u) = x_v$. Here, the degree of a node $\deg_N(v)$ corresponds to $\deg_G(v)$.

In the port-numbering model we are given a distributed system consisting of a port-numbered network of $|V| = n$ *processors* (or *nodes*) that operates in a sequence of synchronous rounds. In each round the processors may perform unbounded computations on their respective local state variables and subsequently exchange messages of arbitrary size along the links given by the underlying input graph. Nodes identify their neighbors by using ports as defined before, where the port assignment may be done adversarially. Barring their degree, all nodes are identical and operate according to the same local computation procedures. Initially all local state variables have the same value for all processors; the sole exception is a distinguished local variable $\mathrm{x}(v)$ of each processor v that encodes input data (that is, port numbers, degree, possible input from the problem itself, etc.). Usually, we assume that $\mathrm{x}(v)$ also encodes the number of nodes n composing the distributed system.

Let Σ_{in} be a set of input labels. The input of a problem is defined in the form of a labeled graph (G, in) where $G = (V, E)$ is the system graph, V is the set of processors (hence it is specified as part of the input), and $\text{in}\colon V \to \Sigma_{\text{in}}$ is an assignment of an input label $\text{in}(v) \in \Sigma_{\text{in}}$ to each processor v and is encoded in $\mathrm{x}(v)$. The output of the algorithm is given in the form of a function of output labels $\text{out}\colon V \to \Sigma_{\text{out}}$, and the algorithm is assumed to terminate once all labels $\text{out}(v)$ are definitely fixed. We assume that nodes and their links are fault-free. The local computation procedures may be randomized by giving each processor access to its own set of random variables; in this case, we are in the *randomized* port-numbering model as opposed to the *deterministic* port-numbering model.

The running time of an algorithm is the number of synchronous rounds required by all nodes to produce output labels. If an algorithm running time is T, we also say that the algorithm has locality T. Notice that T can be a function of the size (or other parameters) of the input graph. We say that a problem Π over some graph family $\mathcal{F}$ has complexity (or locality) T in the port-numbering model if there is a port-numbering algorithm running in time T that solves Π over $\mathcal{F}$, and $T = T(n)$ is the minimum running time (among all possible algorithms that solve Π over $\mathcal{F}$) in the worst case instance of size n. If the algorithm is randomized, we also require that the failure probability is at most $1/\text{poly}(n)$, where n is the size of the input graph.

We remark that the notion of an (LCL) problem is a graph problem, and does not depend on the specific model of computation we consider (hence, the problem definition cannot depend on, e.g., port numbers).

The LOCAL model. The LOCAL model of computation is just the port-numbering model augmented with an assignment of unique identifiers to nodes. Let $c \geq 1$ be a constant. The nodes of the input graph $G = (V, E)$ are given as input also unique identifiers specified by an injective function $\mathrm{id} : V \to [n^c]$. This assignment might be adversarial and is stored in the local state variable $\mathrm{x}(v)$, and nodes can exploit these values during their local computation.

The local computation procedures may be randomized by giving each processor access to its own set of random variables; in this case, we are in the *randomized* LOCAL (rand-LOCAL) model as opposed to the *deterministic* LOCAL (det-LOCAL) model. If the algorithm is randomized, we also require that the failure probability while solving any problem is at most $1/\mathrm{poly}(n)$, where n is the size of the input graph. The definition of running time, locality and complexity easily extends from the port-numbering model to the LOCAL model.

On top of the LOCAL model, it is easy to describe its quantum generalization. In order to avoid the math of quantum mechanics, we only provide an informal definition of the quantum-LOCAL model. For a formal definition, we defer the reader to [GKM09].

The quantum-LOCAL model. The quantum-LOCAL of computing is similar to the deterministic LOCAL model above, but now with quantum computers and quantum communication links. More precisely, the quantum computers manipulate local states consisting of an unbounded number of qubits with arbitrary unitary transformations, the communication links are quantum communication channels (adjacent nodes can exchange any number of qubits), and the local outputs can be the result of any quantum measurement.

Relations between models. We say that a computational model A is *stronger* than a computational model B if an algorithm with locality T running in A can be simulated by an $O(T)$-round algorithm in B. Clearly, det-LOCAL is stronger than the port-numbering model, rand-LOCAL is stronger than det-LOCAL, and quantum-LOCAL is stronger than rand-LOCAL. We suggest the reader has Fig. 1 at hand to keep track of the models and their relations while we introduce them. We will define the other models present in Fig. 1 later in the related sections.

3 The non-signaling model

As mentioned in the introduction, to date we do not have direct ways to prove lower bounds in quantum-LOCAL. Specific procedures that are commonly used in the LOCAL model such as round elimination [Bra19] do not generalize to quantum-LOCAL (regarding this, we argue more later in Section 6.2). However,

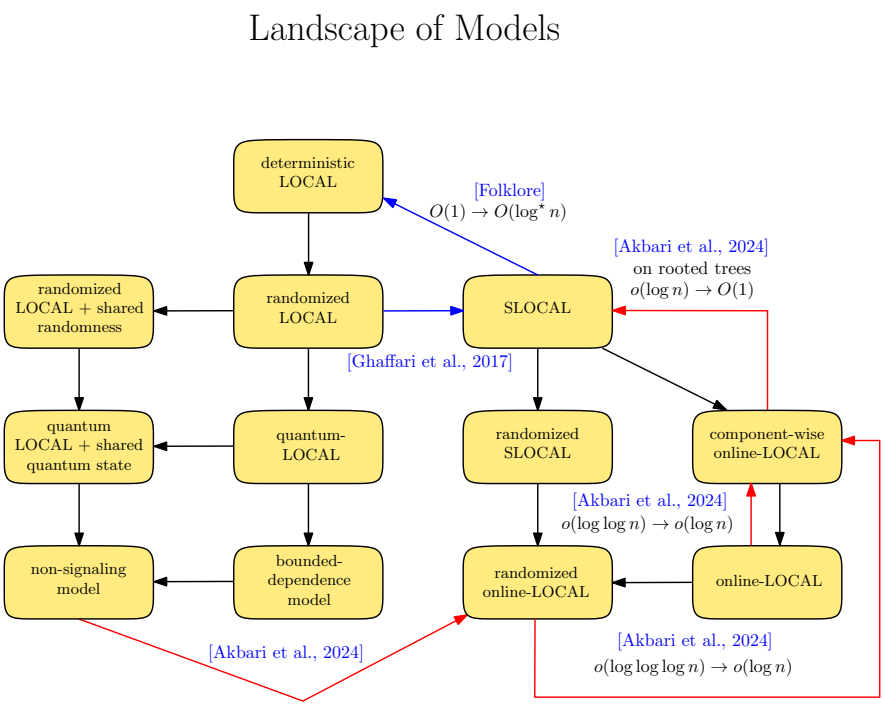


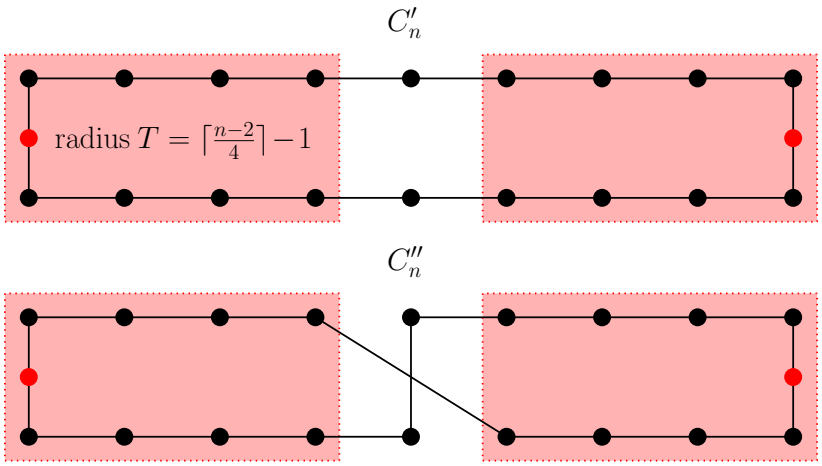
Figure 1: Landscape of computational models. An arrow between model X and Y, that is $X \to Y$ means that model Y is stronger than model X, unless otherwise specified. Black arrows are trivial implications (by construction), blue arrows are known results, and red arrows are recent results.

some general arguments based on the *non-signaling* principle do generalize to quantum-LOCAL and actually holds more in general. Before going through specific definition, let us introduce the no-signaling principle via two examples on the same problem: 2-coloring even cycles both in det-LOCAL and in rand-LOCAL.

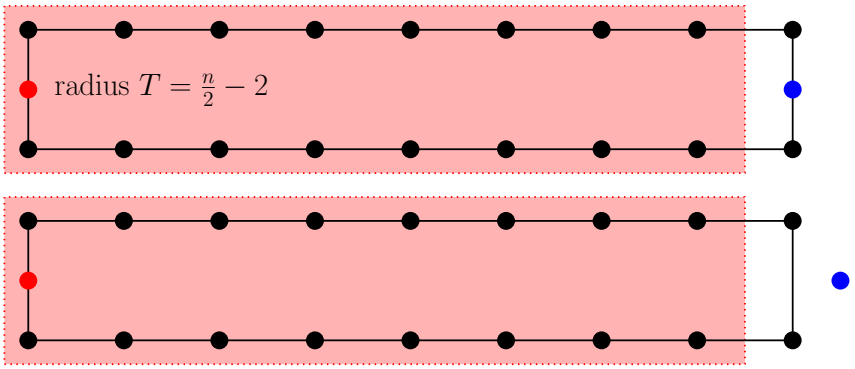
3.1 Warm-up: 2-coloring cycles in classical LOCAL

Let $n \in \mathbb{N}$ be an even number and consider a cycle C_n with n nodes.

First example: indistinguishability argument (*within* the input graph family). The indistinguishability argument in the LOCAL model relies on the fact that a node, by accessing its local view, cannot distinguish between *proper* inputs that are indistinguishable in the local neighborhood but differ outside and, for which,



(a) Indistinguishability argument *within* the same input graph family. The two highlighted nodes cannot distinguish between the case in which they have even (in C'_n) or odd (in C''_n) distance between each other.



(b) Graph-existential argument: *outside* the input graph family. The red nodes cannot distinguish whether they are in an even or an odd cycle.

Figure 2: The no-signaling principle illustrated in the problem of 2-coloring cycles.

the node must behave in different ways. For example, the bipartitness of an even cycle is a very rigid property: if the radius-T neighborhoods of two distinct nodes in a cycle do not intersect, the nodes cannot guess if they are in the same set of the bipartition or not. More formally, assume that there is a det-LOCAL algorithm $\mathcal{A}$ that 2-colors C_n in time $T = \lceil (n-2)/4 \rceil - 1$. Consider two nodes u_1, u_2 that are at distance at least $2\lceil (n-2)/4 \rceil - 1$ between each other. Their radius-T neighborhoods do not intersect, hence there is no way to coordinate in case their distance is odd (which implies different colors) or even (which implies the same color). One can instantiate two input graphs C'_n and C''_n in which the distance between u_1 and u_2 is even and odd, respectively. However, when running $\mathcal{A}$ the output must be the same, leading to a contradiction (see Fig. 2a). This argument gives a lower bound of $\lceil (n-2)/4 \rceil - 1$ to the problem. As for rand-LOCAL, simply notice that, in the worst case, the two nodes u_1 and u_2 cannot produce the correct output with probability more than $1/2$.

Second example: graph-existential argument (*outside* the input graph family). Graph-existential arguments are another key technique to prove lower bounds in classical LOCAL. The idea is the following: Suppose we have an LCL problem that assume that the input graph family satisfies some key-property (in this case, bipartite cycles). Assume also that one can find a graph that *locally* looks like a proper input, but lies *outside* the input graph family (e.g., an odd cycle). In this way, one can exploit the fact that, if the locality T of the algorithm solving the problem is not large enough to detect that the graph is not proper, then the algorithm must run and produce some local failure (e.g., two monochromatic nodes). We can now take a copy of the radius-T neighborhood of the failing nodes and construct a proper input that contains this copy. Since the nodes do not distinguish in which input (proper or not) they are in, they must produce the same output, leading to the failure of the algorithm in a proper instance (see Fig. 2b). Formally, assume that there is a det-LOCAL algorithm $\mathcal{A}$ that 2-colors C_n in time $T = n/2 - 2$. Consider now a second graph G of n nodes that is the disjoint union of a cycle C_{n-1} with $n-1$ nodes and a single node with no edges. Run $\mathcal{A}$ in G: the nodes in C_{n-1} will not notice that they are in an odd cycle since the radius-T neighborhood of any node leaves out 2 nodes. Hence, the nodes must output some color and, since C_{n-1} is not 2-colorable, there must be a failure somewhere, that is, two adjacent nodes u_1, u_2 share the same color. We can now construct an even cycle C_n that contains a subgraph isomorphic to $C_{n-1}[\mathcal{N}_T[\{u_1, u_2\}]]$, with the same input data (that is, identifiers and port numbers). Since the isomorphic copies of u_1, u_2 in C_n cannot distinguish if they are in C_n or in C_{n-1} when running $\mathcal{A}$, they must produce the same failure, contradicting the hypothesis that $\mathcal{A}$ was correct. This argument gives a lower bound of $n/2 - 2$ for 2-coloring even cycles with n

nodes.

When we allow randomness, things are a bit more complex. Indeed, the failure in C_{n-1} is random and when we look at two specific adjacent nodes, in the worst case, the probability of a failure (i.e., a monochromatic edge) can be as small as $1/\text{poly}(n)$. However, we can do something different. Let $u_1, \ldots, u_{n-1}$ be the nodes of C_{n-1}, where $\{u_i, u_{i+1}\}$ is an edge for $i = 1, \ldots, n-2$, and $\{u_1, u_{n-1}\}$ is another edge. Consider two subgraphs G, H of C_{n-1} defined as follows: $V(G) = \{u_1, \ldots, u_{n/2}\}$, $V(H) = \{u_{n/2}, \ldots, u_{n-1}\}$. Now assume $\mathcal{A}$ is a rand-LOCAL algorithm that 2-colors even cycles of n nodes in time $T = \lceil (n-2)/4 \rceil - 1$, and run $\mathcal{A}$ in C_{n-1} (plus one disjoint singleton node to make the number of nodes equal to n). Again, since locality T is not sufficient for the nodes of C_{n-1} to understand that they are not in an even cycle, there must be a failure in at least two adjacent nodes u, v at every run of $\mathcal{A}$. Since $G \cup H = C_{n-1}$, in at least one of them there is probability no less than $1/2$ that a failure takes place. Wlog, assume G is such subgraph. Then, one can construct an even cycle of n nodes that contains as induced subgraph a copy of $\mathring{C}_{n-1}[\mathcal{N}_T[V(G)]]$ and give as input the same identifiers and port numbers. Since the view of the nodes of the copy of G in C_n is indistinguishable from the view of the nodes of G in C_{n-1}, they must reproduce the same failure probability. Hence, we get that any algorithm that 2-colors cycles in rand-LOCAL with locality at most $T = \lceil (n-2)/4 \rceil - 1$ fails with probability at least $1/2$. Notice that we could consider more subgraphs of C_n and get lower bounds with higher locality but smaller failure probability.

Randomized LOCAL: boosting the failure probability. In rand-LOCAL, we can also boost the failure probability at the cost of worsening the locality of the lower bound, by simply repeating the experiment many times. We only focus on the graph-existential argument, but a similar approach holds for the other argument as well. We assume that the overall amount of nodes is now $n = m \cdot N$ for two positive integers m, N where m is odd and N is even. Assume now that we have an algorithm $\mathcal{A}$ that 2-colors even cycles of n nodes with locality $T = \lceil (m-2)/4 \rceil - 1 = \lceil (n/N - 2)/4 \rceil - 1$. Now consider N disjoint copies $C_m^{(1)}, \ldots, C_m^{(N)}$ of an odd cycle C_m with m nodes. For each $C_m^{(i)}$, the same argument as before holds, and we can identify a subgraph G_i of each $C_m^{(i)}$ where a failure takes place with probability at least $1/2$ independently of the others, and such that the radius-T neighborhood of $V(G)$ still leaves at least one node of $C_m^{(i)}$ out. Now we can construct a proper input C_n that contains, as induced subgraphs, a copy of each $C_m^{(i)}[\mathcal{N}_T(V(G_i))]$. By independence, the failure probability here is at least $1 - \frac{1}{2^N}$. Hence, any algorithm 2-coloring even cycles with locality $T = \lceil (n/N - 2)/4 \rceil - 1$ fails with probability at least $1 - \frac{1}{2^N}$. See Fig. 3 for a visual explanation.

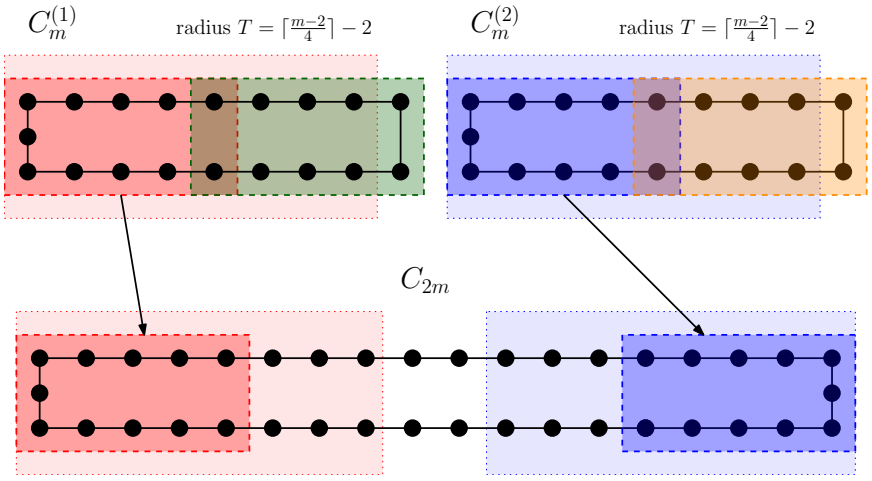


Figure 3: Visual representation of the failure probability boost: We take two odd cycles $C_m^{(1)}$ and $C_m^{(2)}$ and subdivide the nodes in two slightly overlapping regions (the colored dashed regions). For $T = \left\lceil \frac{m-2}{4} \right\rceil - 2$, a T-round algorithm $\mathcal{A}$ must fail both in $C_m^{(1)}$ and $C_m^{(2)}$ (it does not catch that we are in odd cycles). For each cycle, in at least one region $\mathcal{A}$ must fail with probability at least 1/2. Wlog, we assume that this happens in the red and the blue regions. Now we create a new cycle C_n with $n = 2m$ nodes where we copy the radius$T$ neighborhoods of the red and the blue region, and we add the remaining nodes. The failing probability of $\mathcal{A}$ over C_n is, by independence, at least 3/4, and we get a lower bound on the locality of magnitude $T = \left\lceil \frac{n-4}{8} \right\rceil - 2$.

3.2 The no-signaling principle

The lower bound techniques in Section 3.1 rely on a crucial assumption, which is quite intuitive when dealing with classical LOCAL. First, let us introduce the notion of *view* of a subset of nodes.

For any input distributed network (G, x) to any problem, and any subset of nodes $S \subseteq V(G)$, the radius-T view of S is $\mathcal{V}_T(S) = (\mathring{G}[\mathcal{N}_T[S]], \mathrm{x} \restriction_{\mathcal{N}_T[S]})$. Basically, $\mathcal{V}_T(S)$ includes everything that can be *seen* by the nodes in S with T rounds of communication, including input data (degree, ports, identifiers and input labels—if any, etc.). Suppose G has n nodes, and fix any subset of nodes $S \subseteq V(G)$. Given any two graphs G, H with inputs $\mathrm{x}_G, \mathrm{x}_H$ and any two subset of nodes $S_G \subseteq V(G)$ and $S_H \subseteq V(H)$, it is natural to define the notion of *isomorphism between views*. We say that $\mathcal{V}_T(S_G)$ is isomorphic to $\mathcal{V}_T(S_H)$ if there exists a function $\varphi : V(G) \to V(H)$ such that the folowwing holds:

1. $\varphi \restriction_{S_G}$ is an isomorphism between $G[S_G]$ and $H[S_H]$
2. $\varphi \restriction_{\mathcal{N}_T[S_G]}$ is an isomorphism between $\mathring{G}[\mathcal{N}_T[S_G]]$ and $\mathring{H}[\mathcal{N}_T[S_H]]$;
3. $\mathrm{x}_G(u) = \mathrm{x}_H(\varphi(u))$.

Consider any T-round (deterministic or randomized) LOCAL algorithm $\mathcal{A}$ run by the nodes of G. Imagine that the distributed network is split in two different laboratories, Alice's and Bob's. Alice's lab contains $\mathcal{V}_0(S)$, while Bob's lab contains everything that is not contained in $\mathcal{V}_T(S)$. When performing T rounds of communication, Alice's lab receive no information from Bob's lab. Hence, the behavior of the output distribution over nodes in Alice's lab cannot change whatever Bob does in his lab, including rearranging links between nodes or manipulating inputs. The *cause* of outputs in Alice's lab *cannot be influenced* by any action of Bob in his lab. See Fig. 4 for a visual representation of the property.[1] This property is the so-known *no-signaling from the future* principle in physics, which states that no signals can be sent from the future to the past, and is equivalent to the *causality principle* [DCP16]. Such principle holds in *every physical distributed network* running any kind of synchronous distributed algorithm, including quantum ones.

To see how this principle formally translates in our setting, let us define the notion of *outcome*, which is some kind of generalization of an algorithm. Here, we restrict to finite sets of input and output labels since we focus on LCL problems.

Definition 3.1 (Outcome)**.** Let $\Sigma_{\text{in}}, \Sigma_{\text{out}}$ be two finite sets of labels, and I a finite set of indices. An *outcome* is a mapping $\mathrm{O} : (G, \mathrm{x}) \mapsto \{(\mathrm{out}_i : V(G) \to \Sigma_{\text{out}}, p_i)\}_{i \in I}$ that assigns to every input distributed network (G, x) (with any input labeling in : $V(G) \to \Sigma_{\text{in}}$) a distribution over output labelings $\{(\mathrm{out}_i : V(G) \to \Sigma_{\text{out}}, p_i)\}_{i \in I}$, where p_i is the probability that out_i occurs; in particular, $0 \leq p_i \leq 1$ and $\sum_{i \in I} p_i = 1$.

This definition is easily generalizable to the case of infinite label sets, but we avoid it for the sake of simplicity. Notice that all synchronous distributed algorithms gives r outcomes: it is just the assignment of an output distribution (the one that is the result of the algorithmic procedure) to the input graph. We remark we have defined the domain set of outcomes to include *all possible graphs*. This is not restrictive: Even classical (or quantum) algorithms can run on every possible graph. One can just introduce some garbage output label so that whenever a node

[1] Formally, Bob's lab also contains edges that are in $E(G[\mathcal{N}_T[S]]) \setminus E(\mathring{G}[\mathcal{N}_T[S]])$, but does not contain the nodes in $\mathcal{N}_T^{T-1}[S]$. We can imagine that Bob sees the ports of edges that are not in $\mathcal{V}_T(S)$, but nothing else, and is constrained to assign edges to those ports. In Fig. 4 Bob would have the freedom to modify such edges in the red region as well, but we avoid representing this aspect for the sake of simplicity.

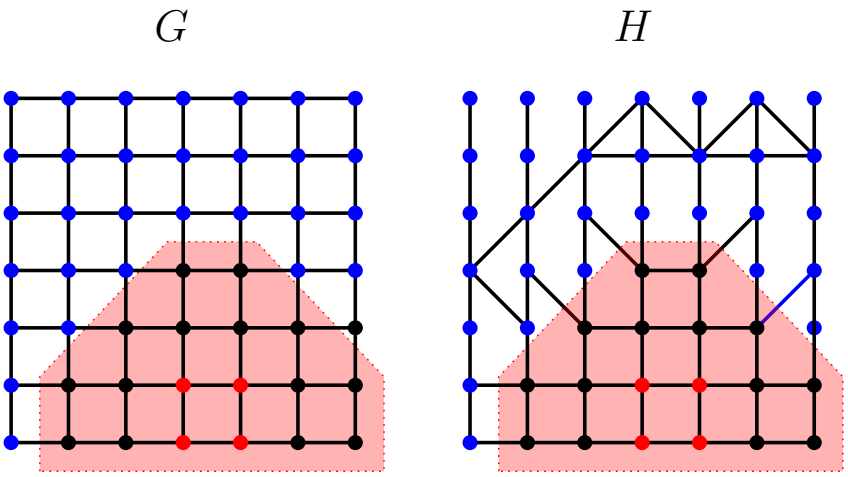


Figure 4: No-signaling property. Alice's lab contains the red nodes, Bob's lab contains the blue nodes. By running any 2-rounds synchronous distributed algorithm, the red nodes cannot distinguish between G and H: Bob has freedom to change the topology outside the red region without being detected by Alice.

running an algorithm needs to do something that is not well-defined (given its neighborhood), it can just output the garbage label.

We say that an outcome O solves a problem Π over a family of graphs $\mathcal{F}$ with probability $q > 0$ if, for every $G \in \mathcal{F}$ and every input data x, it holds that

$$\sum_{\substack{i\in I:\\ \text{out}_i\in\Pi((G,\text{x}))}} p_i \geq q.$$

Let $\text{O} : (G, \text{x}) \mapsto \{(\text{out}_i, p_i)\}_{i\in I}$ be any outcome and fix an input (G, x). Consider any subset of nodes $S \subseteq V(G)$. The restriction of the output distribution $\text{O}((G, \text{x})) = \{(\text{out}_i : V(G) \to \Sigma_{\text{out}}, p_i)\}_{i\in I}$ to S is the distribution $\{(\text{out}_j : S \to \Sigma_{\text{out}}, p'_j)\}_{j\in J}$ such that

$$p'_j = \sum_{\substack{i\in I:\\ \text{out}_j=\text{out}_i\restriction_S}} p_i,$$

and is denoted by $\text{O}((G, \text{x}))[S]$ or $\{(\text{out}_i, p_i)\}_{i\in I}[S]$. We also say that two labeling distributions $\{(\text{out}_i : V(G) \to \Sigma, p_i)\}_{i\in I}, \{(\text{out}_j : V(H) \to \Sigma, p_j)\}_{j\in J}$ over two graphs G, H are isomorphic if there is an isomorphism $\varphi : V(G) \to V(H)$ between G and H such that $\{(\text{out}_i : V(G) \to \Sigma, p_i)\}_{i\in I} = \{(\text{out}_j \circ \varphi : V(G) \to \Sigma, p_j)\}_{j\in I}$.

We are now ready to define *non-signaling outcomes*.

Definition 3.2 (Non-signaling outcome)**.** Let O be any outcome. Fix any two graphs G, H of n nodes and any two input data functions $\mathrm{x}_G, \mathrm{x}_H$. Suppose there exists a non-negative integer $T \geq 0$ with the following property: For any two subsets $S_G \subseteq V(G), S_H \subseteq V(H)$ such that $\varphi : V(G) \to V(H)$ is an isomorphism between $\mathcal{V}_T(S_G)$ and $\mathcal{V}_T(S_H)$, then the restrictions $\mathrm{O}((H, \mathrm{x}_H))[S_H]$ and $\mathrm{O}((G, \mathrm{x}_G))[S_G]$ are isomorphic under φ. We say that O is *non-signaling beyond distance T* or, alternatively, that O has locality T.

With this notion, we can define the non-signaling model.

The non-signaling model. The non-signaling model is a computational model which produces non-signaling outcomes. More specifically, the distributed network in input (G, x) is as in the definition of the deterministic LOCAL model, with $\mathrm{x}(v)$ encoding the degree of a node, port numbers, the identifier and (possibly) an input label expected by the problem of interest. The model can produce non-signaling outcomes where one wants to minimize the locality T to solve the problem. Usually, we require that the success probability of an outcome that solves a problem is at least $1 - 1/\mathrm{poly}(n)$.

The non-signaling model was first introduced by Gavoille, Kosowski, and Markiewicz [GKM09] with a slightly different definition, and then formalized by Arfaoui and Fraigniaud [AF14] in the current form. It is stronger than any *physical* synchronous distributed computing model, as T-rounds synchronous distributed algorithms (both classical and quantum) obey the no-signaling principle and must produce outcomes that are non-signaling beyond distance T. [GKM09] was the first to observe that lower bounds in the non-signaling model must hold in all weaker models, and noticed that lower bound techniques based on the indistinguishability argument *withing* the input graph family still hold in non-signaling. More specifically, [GKM09] revisited some previous lower bound results and noticed that they hold also in the non-signaling model, and it also established a new lower bound for 2-coloring even cycles, revisiting the first argument on indistinguishability shown in Section 3.1.

Theorem 3.3 (Gavoille, Kosowski, and Markiewicz [GKM09])**.** *In the non-signaling model, the following holds:*

1. *Maximal independent requires locality $\Omega\left(\sqrt{\frac{\log n}{\log\log n}}\right)$ [KMW04].*
2. *Locally minimal (greedy) coloring requires locality $\Omega\left(\frac{\log n}{\log\log n}\right)$ [Gav+07; Gav+09].*
3. *Finding a connected subgraph with $O\left(n^{1+1/k}\right)$ edges requires locality $\Omega(k)$ [Der+08; Elk07].*

4. *Finding a 2-coloring in even cycles requires locality* $\Omega\left(\left\lceil \frac{n-2}{4} \right\rceil\right)$.

We remark that [GKM09] had a slight different definition of the non-signaling model. In the paper, the model was called φ-LOCAL and defined outcomes to exist only on a specific input graph family. When considering, e.g., the problem of 2-coloring even cycles, the outcome produced by φ-LOCAL were defined *only* for even cycles, leaving out the possibility to play with graphs outside the input graph family. However, as previously mentioned, (classical or quantum) algorithms can be run also on network graphs that lie outside the input graph family: If computation is at any point undefined, we can let a node output some garbage label. This freedom that we have with algorithms is what we exploit in the second lower bound argument in Section 3.1, the graph-existential one. By defining outcomes on *every possible input graph*, we have access to new lower bound techniques. In [Coi+24], we proved that the graph-existential argument can be extended all the way up to the non-signaling model (under some general hypotheses), and used it to prove the following lower bounds.

Theorem 3.4 (Coiteux-Roy et al. [Coi+24])**.** *In the non-signaling model, the following holds:*

1. *The problem of c-coloring χ-chromatic graphs requires locality* $\Omega\left(n^{1/\left\lfloor \frac{c-1}{\chi-1} \right\rfloor}\right)$.
2. *Finding a* 3*-coloring $n \times m$ grids requires locality* $\min\{n, m\}$.
3. *Finding a c-coloring of trees requires locality* $\log_c n$.

The three results of Theorem 3.4 make use of *cheating graphs*, that is, of graphs that are locally everywhere indistinguishable from proper inputs, but globally they are not in the input family. E.g., for c-coloring χ-chromatic graphs we need to find a graph such that the radius-T neighborhood of any node induces a graph that is χ-colorable, for $T = \Theta\left(n^{1/\left\lfloor \frac{c-1}{\chi-1} \right\rfloor}\right)$, but globally the graph has chromatic number strictly greater than c. The existence of such graph immediately implies that deterministic LOCAL algorithms require locality $\Omega\left(n^{1/\left\lfloor \frac{c-1}{\chi-1} \right\rfloor}\right)$ to solve the problem. Such graph is given, for all combinations of c and χ, by Bogdanov [Bog13]: this was the first time that Bogdanov's construction found application in distributed computing and, more in general, theoretical computer science. Interestingly, in [Coi+24] the authors proved an almost matching upper bound for the problem in det-LOCAL, thus excluding any significant quantum advantage over det-LOCAL for this problem. As for 3-coloring grids, we made use of odd quadrangulations of Klein-bottles [MS02], which are everywhere locally indistinguishable from grids, but have global chromatic number, while for c-coloring trees we

revisited Linial's argument in [Lin92] that made use of Ramanujan graphs, which are high-girth, high-chromatic graphs [MST13]. When looking at rand-LOCAL and the non-signaling model, we need to make sure that the graph can be nicely covered by a small amount of subgraphs (whose union form the whole graph) that are slightly overlapping, in order to identify a subgraph where the failure probability is high enough. In general, boosting the failure probability is possible also in the non-signaling model. However, we cannot rely on the same argument of Section 3.1 as a non-signaling outcome does not guarantee *independence* between the output distributions of far-away subsets of nodes (which instead is guaranteed by rand-LOCAL algorithms). The reason is that non-signaling outcomes include the possible use of *global* resources, such as *shared randomness* or *shared quantum states*. Such resources make output distributions of distant nodes dependent of each other, even if their distance is greater than the locality of the algorithm itself. Nonetheless, boosting the failure probability is still possible provided that the cheating graph meets some properties. For more details, we defer the reader to the original article [Coi+24].

All this discussion might suggest that non-signaling argument are sufficient to exclude quantum advantage, and hence gives rise to the following question:

Question 1. *Can we exclude quantum advantage for all LCL problems using non-signaling arguments?*

Unfortunately, the answer to this question is *no*. Indeed, the non-signaling model is too strong to compare with classical LOCAL. For example, very recently Balliu et al. [Bal+24] solved a longstanding open problem, proving that shared randomness gives advantage over private randomness in classical LOCAL when restricting to LCL problems (otherwise, the thesis is trivial). More specifically, there is an LCL problem that requires $\Omega\left(\sqrt{n}\right)$ rounds in rand-LOCAL but can be solved in $O(\log n)$ rounds when nodes have access to shared randomness (e.g., an infinite random bit string). Since the non-signaling model is strong enough to simulate LOCAL with shared randomness, we already know that it is too strong with respect to classical LOCAL and there is no hope to prove something like Question 1. However, it is worth investigating to what extent lower bound arguments based on the non-signaling property apply, since we still miss a characterization.

Question 2 (Open)**.** *For which LCL problems can we exclude quantum advantage using non-signaling arguments?*

4 The bounded-dependence model

To avoid dealing with shared resources, we can *weaken* the non-signaling model by introducing further restrictions on the non-signaling outcomes. Running T-

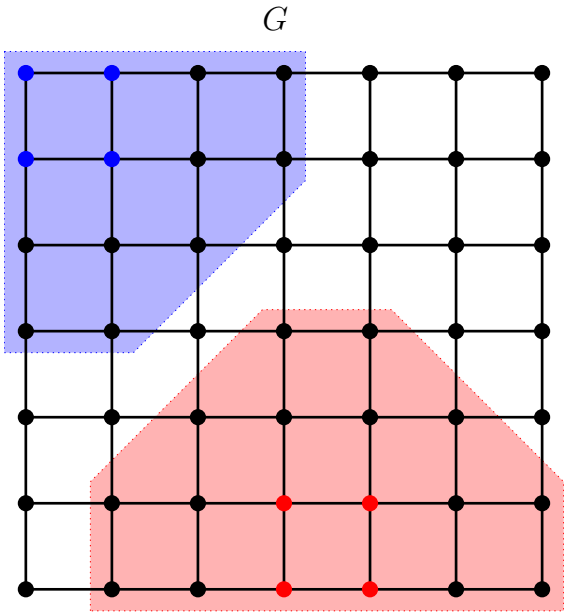


Figure 5: Bounded-dependence property. When running any 2-round synchronous distributed algorithm (which does not rely on shared resources), the output labeling distribution of the red nodes is independent of that of the blue nodes.

rounds classical or quantum-LOCAL algorithms without shared resources, we obtain the following property on the output labeling distribution: for every two subset of nodes A, B such that $\mathrm{dist}(A, B) > 2T$, then the output distributions restricted to A and B are independent (see Fig. 5). Let us formalize this notion.

Definition 4.1 (T-dependent distribution)**.** Let Σ_{out} and I be two sets, and let (G, x) be an input for some labeling problem Π. An output labeling distribution $\{(\text{out}_i : V(G) \to \Sigma_{\text{out}}, p_i)\}_{i\in I}$ is T-dependent if, for any two subsets of nodes $A, B \subseteq V(G)$, we have that $\{(\text{out}_i, p_i)\}_{i\in I}[A]$ is independent of $\{(\text{out}_i : V(G) \to \Sigma_{\text{out}}, p_i)\}_{i\in I}[B]$.

We can think now of non-signaling outcomes that produces such distributions.

Definition 4.2 (Bounded-dependent outcome)**.** Let O be any outcome that is non-signaling beyond distance T. We say that O is bounded-dependent with locality T if for any input (G, x) we have that $\mathrm{O}((G, \mathrm{x}))$ is $2T$-dependent. Furthermore, when $T = O(1)$ (i.e., it does not depend on the input graph), we say that $\mathrm{O}((G, \mathrm{x}))$ is a finitely-dependent distribution. Moreover, if for all inputs (G, x) it holds that $\mathrm{O}((G, \mathrm{x}))$ does not depend on identifiers and port numbers, we say that O is *invariant under subgraph isomorphism*.

With the addition of this further property, we can define the *bounded-dependence model*, first formalized in [Akb+25].

The bounded-dependence model. Similarly to the introduction of the non-signaling model, we can define the *bounded-dependence model* as a model that produces bounded-dependent outcomes. Again, the required success probability then solving a problem should be at least $1 - 1/\mathrm{poly}(n)$.

One might hope that we can prove lower bounds in the bounded-dependence model and matching upper bounds in classical LOCAL.

Question 3. *Can we always rule out quantum advantage for LCLs using bounded-dependent outcomes?*

Unfortunately, the answer to Question 3 is *no*: Holroyd and Liggett [HL16] and Holroyd, Hutchcroft, and Levy [HHL18] proved it for us. It is well-known that 3-coloring paths and cycles requires locality $\Theta(\log^\star n)$ both in det-LOCAL and in rand-LOCAL [Lin92; CKP16]. [HL16; HHL18] showed that these problems admit finitely-dependent distributions, that is, it is possible to 3-color path and cycles with an $O(1)$-dependent outcome that is invariant under subgraph isomorphism and under permutations of the colors.

Apart from 3-coloring paths and cycles, there are other famous problems in the complexity class $\Theta(\log^\star n)$, such as computing an MIS, $(\Delta + 1)$-coloring graphs of maximum degree Δ, etc.

All such problems are also called *symmetry-breaking* problems, in the sense that they are easily solvable by an $O(1)$-round port-numbering algorithm (and without knowledge of n) if symmetry is locally broken with a constant amount of labels. This fact implies that, given any two LCLs Π_1, Π_2 that have complexity $\Theta(\log^\star n)$ in some graph G, a solution to any of those problems can be converted into a solution of the other in a constant number of rounds. In [Akb+25], the authors proved it is possible to compose port-numbering algorithms (that do not make use of n) and bounded-dependent outcomes obtaining a bounded-dependent outcome (without significant loss in locality). Hence, the results in [HL16; HHL18] immediately imply that in paths and cycles all LCL problems that have classical complexity $\Theta(\log^\star n)$ are solvable by an $O(1)$-dependent outcome in the bounded-dependence model as well: but what about other graphs?

Question 4. *Is there any LCL problem with classical complexity* $\Theta(\log^\star n)$ *for which we can rule out quantum advantage using the bounded-dependence model?*

Unfortunately the answer is, again, *no*. Akbari et al. [Akb+25] proved that all such problems are solvable with locality $O(1)$ in the bounded-dependence model, and the resulting bounded-dependent outcomes are also invariant under subgraph isomorphism.

Theorem 4.3 (Akbari et al. [Akb+25])**.** *Let* Π *be an LCL over some input graph family* $\mathcal{F}$ *that has complexity* $O(\log^\star n)$ *in classical LOCAL. Then, there exists an* $O(1)$*-dependent outcome* O *that solves* Π *over* $\mathcal{F}$*. Furthermore,* O *is invariant under subgraph isomorphism.*

Theorem 4.3 is powerful result that shows the power of finitely-dependent distributions: such distributions are able to break symmetry with constant locality, that is something that classical LOCAL cannot do. This leaves us with one of the major open question in the field.

Question 5 (Open)**.** *Is quantum-LOCAL able to break symmetry in LCLs? That is, can quantum-LOCAL solve in time* $o(\log^\star n)$ *any LCL* Π *that has classical complexity* $\Theta(\log^\star n)$*?*

We currently lack tools to analyze directly quantum-LOCAL (especially regarding lower bounds). What we can do is instead focusing on specific graph families and/or specific complexity classes in the bounded-dependence or non-signaling model. In classical (both deterministic and randomized) LOCAL, LCL complexities belong to the following three classes: $O(1)$, $\Theta(\log^\star n)$, and $\Omega(\log\log n)$. After Theorem 4.3, we might wonder what happens in the complexity class $\Omega(\log\log n)$.

Question 6 (Open)**.** *In the bounded-dependence model and the non-signaling, what can we infer on the complexity of LCL problems that have classical complexity* $\Omega(\log\log n)$*?*

One of the most prominent problems is maybe sinkless orientation (SO), a problem that asks each node v to orient its adjacent edges so that $\mathrm{outdeg}(v) \geq 1$. It is known that SO has complexity $\Theta(\log n)$ in det-LOCAL and $\Theta(\log\log n)$ in rand-LOCAL [Bal+23b]. Hence, we can formulate the following question, which is nowadays open.

Question 7 (Open)**.** *What is the complexity of sinkless orientation in the bounded-dependence model or the non-signaling model?*

Currently, we have little insight on these questions, but we managed to give some small partial answers to Question 6, which we address in the following section.

5 The online-LOCAL model

In this section, we describe other models of computation that, at a first glance, seem completely unrelated from the (super)quantum world we have been describing so far, but after a deeper look turn out to be related and extremely useful. In [GKM17], Ghaffari, Kuhn, and Maus introduced the SLOCAL model, that is, a sequential version of the LOCAL model.

The SLOCAL model. The SLOCAL model is similar to the LOCAL model, but sequential. Here, the nodes of the input graph $G = (V, E)$ with $|V| = n$ are processed according to an adversarial order $\sigma = v_1, \ldots, v_n$. While processing a node v_i, a T-round algorithm collects all data contained in and the topology of the radius-T neighborhood of v_i (including the states and the outputs of previously processed nodes in $\mathcal{N}_T[v_i]$, i.e., any node $v_j \in \mathcal{N}_T[v_i]$ for $j < i$): We say that such an algorithm has locality/complexity/running time T. Note that the algorithm might store the whole data contained in $\mathcal{V}_T(v_i)$ inside the memory of v_i (and we always assume this happens, since it can only make the algorithm stronger). This phenomenon gives to a node v_i access to the data contained in $\mathcal{V}_T(v_j)$ if and only if there is a subsequence of nodes $\{v_{h_k}\}_{k\in[m]}$ with $j = h_k < h_{k+1} < \cdots < h_m = i$ such that $v_{h_k} \in \mathcal{N}_T[v_{h_{k+1}}]$ for all $k \in [m-1]$.

If the algorithm is given in input an infinite random bit string, we talk about the randomized SLOCAL model, as opposed to the deterministic SLOCAL model. Notice that the adversarial order in which node are processed is assumed to be *oblivious* to the random bit string. In this case, we require the success probability to be at least $1 - 1/\mathrm{poly}(n)$, with n being the number of nodes of the input graph.

Clearly, the SLOCAL model is stronger than the LOCAL model, since any deterministic T-round LOCAL algorithm can be converted into a deterministic T-round SLOCAL algorithm. Surprisingly, Ghaffari, Harris, and Kuhn [GHK18] proved that also any randomized T-round LOCAL algorithm can be converted into a deterministic $O(T)$-round SLOCAL algorithm through some derandomization technique. Interestingly, [GKM17] proved that, under certain hypotheses, (both randomized and deterministic) SLOCAL algorithms can be converted in (respectively, randomized and deterministic) LOCAL algorithms with some overhead in the complexity. However, for our purposes it is sufficient to know that $O(1)$-round SLOCAL algorithms for LCLs can be turned into $O(\log^\star n)$-round LOCAL algorithms: this is folklore, but a proof can be found in [Akb+25]. See Fig. 1 for a representation of the relations among models.

On top of the SLOCAL model, Akbari et al. [Akb+23] introduced the online-LOCAL model. The online-LOCAL model is simply the SLOCAL model equipped with global memory.

The online-LOCAL model. The (deterministic) online-LOCAL model is basically equivalent to the SLOCAL model with global memory. More specifically, the online-LOCAL model is a centralized model of computing where the algorithm initially knows only the set of nodes of the input graph G. The nodes are processed with respect to an adversarial input sequence $\sigma = v_1, v_2, \ldots, v_n$. The output of v_i depends on $G_i = \cup_{j=1}^{i} \mathcal{V}_T(v_j)$, i.e., the radius-$T$ views of of $v_1, v_2, \ldots, v_i$ (which includes all input data), plus all the outputs of previously processed nodes

(we can imagine that the views get updated at each step of the algorithm).

In [Akb+25], the authors defined the randomized online-LOCAL model as a randomized variant of the online-LOCAL model where the label assigned by the algorithm to v_i might depend on arbitrarily large portions of an infinite random bit string. Note that this model is oblivious to the randomness used by the algorithm. In particular this means that the graph outside G_i cannot be changed depending on the label assigned to v_i. One could also define the randomized online-LOCAL model in an adaptive manner, but it turns out that this is equivalent to the deterministic online-LOCAL model (as proved in [Akb+25]). The notion of complexity of a problem can be easily extended from the LOCAL model. If the algorithm is randomized, we also require that the failure probability is at most $1/\mathrm{poly}(n)$, where n is the size of the input graph.

Interestingly, [Akb+23] proved that in rooted regular trees LCLs have roughly the same complexity across the three deterministic models LOCAL, SLOCAL, and online-LOCAL.

So, why should we care about the online-LOCAL model? Akbari et al. [Akb+25] found a very important connection with the non-signaling model, which we report here with the following theorem: in [Akb+25] the result is stated only for LCL problems, but the proof holds also for any labeling problem.

Theorem 5.1 (Akbari et al. [Akb+25])**.** *Let* Π *be any labeling problem and* O *any non-signaling outcome with locality* T *solving* Π *with probability* $p > 0$*. Then, there is a randomized online-LOCAL algorithm* $\mathcal{A}$ *with locality* T *that solves* Π *with probability* p*. Furthermore, the output distribution of* $\mathcal{A}$ *over any input* (G, x) *is exactly* $\mathrm{O}((G, \mathrm{x}))$*.*

Theorem 5.1 is very powerful, as it allows us to focus on classical, centralized models of computing: every lower bound in randomized online-LOCAL holds also in non-signaling and, hence, in quantum-LOCAL. Surprisingly, [Akb+25] proved even more results regarding the online-LOCAL model that connects it with the classical LOCAL model.

In order to introduce it, we need to define *component-wise* online-LOCAL algorithms.

The component-wise online-LOCAL model. The component-wise online-LOCAL model is exactly the same as the deterministic online-LOCAL model but when the algorithm processes a node v_i according to the adversarial order $\sigma = v_1, \ldots, v_n$, v_i does not have access to the whole G_i. Rather, it has access only to its connected component in G_i. Clearly a component-wise online-LOCAL algorithm is also a standard online-LOCAL algorithm, hence the model is weaker than the deterministic online-LOCAL model (see Fig. 1 for a landscape of all the computational models). However, we have the following result.

Theorem 5.2 (Akbari et al. [Akb+25])**.** *Let* Π *be any LCL problem, and let* $\mathcal{A}$ *be any online-LOCAL algorithm solving* Π *with locality* $T(n)$ *for graphs of n nodes. Then the following holds:*

1. *If* $\mathcal{A}$ *is deterministic, then there exists a deterministic component-wise online-LOCAL algorithm solving* Π *with locality* $T(2^{O(n^3)})$*.*
2. *If* $\mathcal{A}$ *is randomized and has success probability* $p(n) > 0$*, then there exists a deterministic component-wise online-LOCAL algorithm solving* Π *with locality* $T\left(2^{O(n^3)} + 2^{O(2^{n^2})} \cdot \log \frac{1}{p(n)}\right)$*.*

5.1 Implications in rooted trees

Why is Theorem 5.2 so meaningful? The difference between online-LOCAL and SLOCAL is *local* vs *global memory*. Component-wise online-LOCAL algorithms lie somewhere in the middle: a node gets access only to the memory contained in the currently explored connected component. Interestingly, in some topologies, the SLOCAL model is able to "simulate" component-wise online-LOCAL algorithms. We remind the reader that a rooted tree is a directed tree where all nodes have outdegree 1 except for a single node v that has outdegree 0 and is called the *root* of the tree.

Theorem 5.3 (Akbari et al. [Akb+25])**.** *Let* Π *be any LCL problem over rooted trees. Assume* $\mathcal{A}$ *is a component-wise online-LOCAL algorithm solving* Π *with locality* $T(n)$*. Then, there exists a deterministic SLOCAL algorithm solving* Π *with locality* $O(1) + T(O(n))$*.*

Now we can go from (deterministic or randomized) online-LOCAL to SLOCAL in rooted trees. Interestingly, [Akb+25] also proved that LCLs over rooted trees in SLOCAL have complexity either $O(1)$ or $\Omega(\log n)$.

Theorem 5.4 (Akbari et al. [Akb+25])**.** *Let* Π *be any LCL problem over rooted trees. Assume* $\mathcal{A}$ *is an SLOCAL algorithm solving* Π *with locality* $o(\log n)$*. Then, there exists another SLOCAL algorithm* $\mathcal{B}$ *solving* Π *with locality* $O(1)$*.*

It is folklore that LCLs that have complexity $O(1)$ in SLOCAL over any topology translate in complexity $O(\log^\star n)$ in classical LOCAL. Altogether, we have the following.

Corollary 5.5 (Akbari et al. [Akb+25])**.** *Let* Π *be any LCL problem over rooted trees. Then the following holds:*

1. *If* Π *has complexity* $o(\log \log n)$ *in deterministic online-LOCAL, then it has complexity* $O(\log^\star n)$ *in LOCAL.*

2. *If* Π *has complexity* $o(\log\log\log n)$ *in randomized online-LOCAL, then it has complexity* $O(\log^\star n)$ *in LOCAL.*

See also Fig. 1 for a drawing of all the implications among models. Corollary 5.5 is very powerful in a twofold sense. On one hand, it implies that the LCL complexity class $O(\log^\star n)$ in quantum-LOCAL and in LOCAL coincide over rooted trees, excluding significant quantum advantage (it might still be that $O(1)$ locality in quantum-LOCAL becomes $\Theta(\log^\star n)$ in LOCAL). Also, the LCL complexity class $O(1)$ in the bounded-dependence and non-signaling models becomes $O(\log^\star n)$ in LOCAL over rooted trees, while we know that $O(\log^\star n)$ in LOCAL becomes $O(1)$ in the bounded-dependence and non-signaling models over any topology by Theorem 4.3. On the other hand, Corollary 5.5 allows us to obtain lower bounds in online-LOCAL (and, hence, quantum-LOCAL, bounded-dependence, and non-signaling) through lower bounds in LOCAL: We know that if an LCL over rooted trees cannot be solved in time $O(\log^\star n)$ in LOCAL, then it needs locality $\Omega(\log\log n)$ in deterministic online-LOCAL and $\Omega(\log\log\log n)$ in randomized online-LOCAL. Hence, in rooted trees we also know that the LCL complexity class $\omega(\log^\star n)$ in LOCAL becomes $\Omega(\log\log\log n)$ in quantum-LOCAL, bounded-dependence, and non-signaling.

5.2 LCL complexity landscape in general trees

Recently, Dhar et al. [Dha+24] analyzed more carefully the relation between the randomized online-LOCAL model and the det-LOCAL model in various kind of trees: rooted, unrooted, regular, etc. They proved results for the LCL complexity class $\omega(\log n)$ by extending previous results all the way up to the randomized online-LOCAL model [Akb+23; Bal+22a; Bal+23a; Bal+21a; Cha20; CP19; GRB22]. By these previous works, it was known that in the LOCAL model, LCL problems over regular trees that have complexity $\omega(\log n)$ fall into one of the following classes: $\Theta\left(n^{1/k}\right)$ for some $k \in \mathbb{N}_+$. Furthermore, if the tree is also rooted, we know that the only possible complexities in det-LOCAL are $O(1)$, $\Theta(\log^\star n)$, and $\Theta(\log n)$. In [Akb+23] this result was extended all the way up to deterministic online-LOCAL (but only for the case of rooted trees, and did not find an *exact* correspondence between LOCAL and online-LOCAL). The authors of [Dha+24] proved the following two theorems.

Theorem 5.6 (Dhar et al. [Dha+24])**.** *Let* Π *be an LCL problem on unrooted regular trees. If* Π *has complexity* $\Theta\left(n^{1/k}\right)$ *for any* $k \in \mathbb{N}_+$ *in det-LOCAL, then it has complexity* $\Theta\left(n^{1/k}\right)$ *in randomized online-LOCAL, and vice-versa.*

Theorem 5.7 (Dhar et al. [Dha+24])**.** *Let* Π *be an LCL problem on rooted regular trees. The following holds:*

1. *If* Π *has complexity* $\Theta\left(n^{1/k}\right)$ *for any* $k \in \mathbb{N}_+$ *in det-LOCAL, then it has complexity* $\Theta\left(n^{1/k}\right)$ *in randomized online-LOCAL, and vice-versa.*

2. *If* Π *has complexity* $\Theta(\log n)$ *in det-LOCAL, then it has complexity* $\Theta(\log n)$ *in randomized online-LOCAL, and vice-versa.*

Combining Theorem 5.7 with Corollary 5.5, we obtain that in the LCL complexity region $O(\log n)$ over rooted regular trees, the following classes contain the same problems in LOCAL and randomized online-LOCAL, and are the only possible complexity classes: $O(\log^\star n)$ in det-LOCAL and $O(1)$ in randomized online-LOCAL, or $\Theta(\log n)$ both in det-LOCAL and randomized online-LOCAL. We remind the reader that all these equivalences between complexity classes also hold between LOCAL and quantum-LOCAL, as well as the bounded-dependence and the non-signaling models, as they are "sandwiched" between det-LOCAL and randomized online-LOCAL. [Dha+24] also provided a general result on trees extending speedup arguments in [Bal+21a].

Theorem 5.8 (Dhar et al. [Dha+24])**.** *Let* Π *be an LCL problem on general trees. Then, either* Π *has complexity* $\Theta(n)$ *in the (deterministic or randomized) LOCAL model and in the (deterministic or randomized, respectively) online-LOCAL model, or the complexity in both models is* $O\left(\sqrt{n}\right)$.

Theorem 5.8 trivially holds for any intermediate model instead of randomized online-LOCAL. We invite the reader to have a look at Fig. 6 for a representation of the LCL complexity landscape in trees after the results of [Akb+25; Dha+24].[2] We conclude the section observing that, perhaps, the most difficult complexity range is that between $\omega(\log^\star n)$ (hence, by known results, $\Omega(\log\log n)$) and $O(\log n)$ in general (regular or non-regular) trees. The reason is that we don't have techniques that we can refer to which we can extend to these (super)quantum models.

Question 8 (Open)**.** *Let* Π *be an LCL problem over (regular or non-regular) trees that has classical complexity between* $\Omega(\log\log n)$ *and* $O(\log n)$*. Can we find non-trivial upper or lower bounds on the complexity of* Π *in randomized online-LOCAL?*

Again, sinkless orientation is an example of such problem. The lower bound of sinkless orientation (and often of other LCLs with similar complexity) is proved in classical LOCAL via a famous lower bound technique known as round elimination [Bra+16; Bra19]. As we argue in the next section, round elimination is

[2] We reproduced Figure 2 in [Dha+24], for which we give credits to the authors.

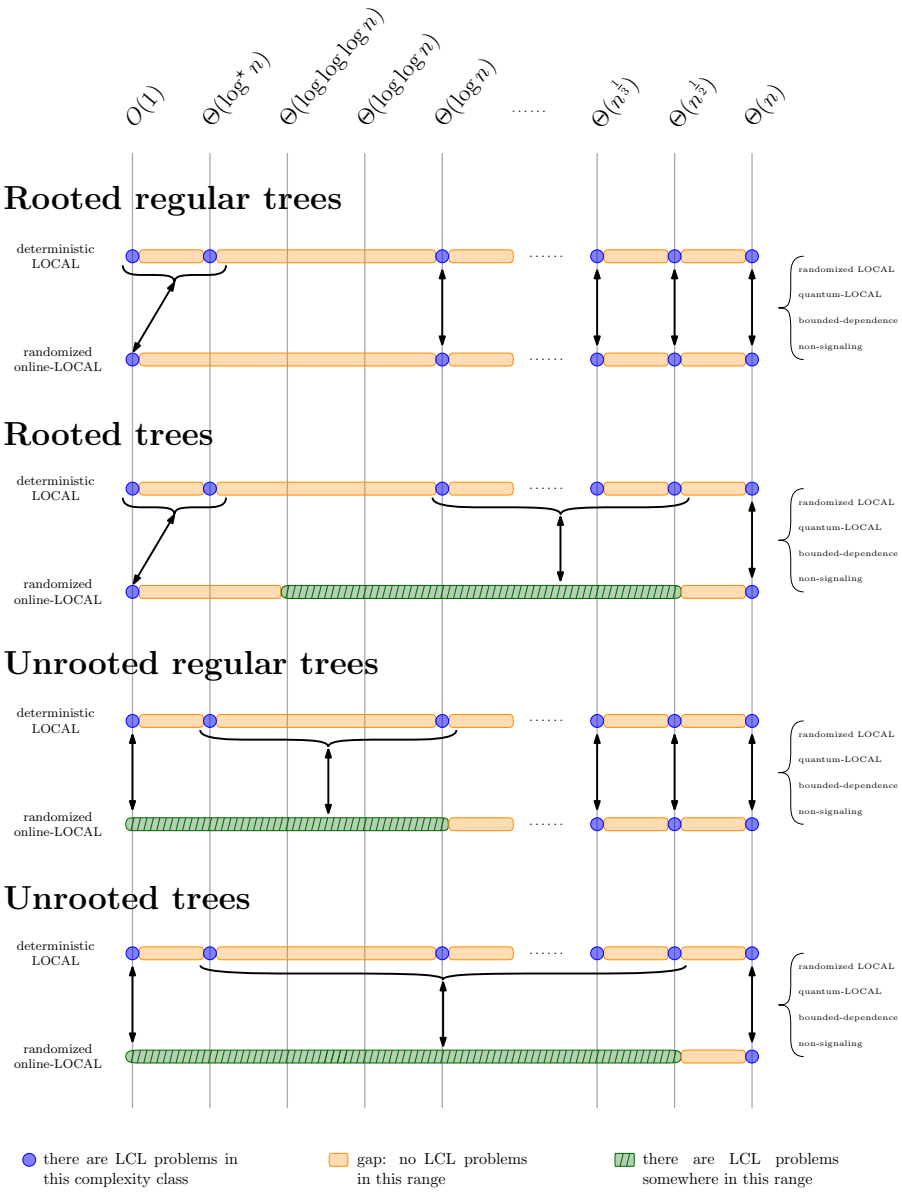


Figure 6: Landscape of LCL complexities in trees.

unfortunately not generalizable to (super)quantum models. Other candidate problems are: 3-coloring trees, 2-2-3 in 3-regular trees (that is, 2-coloring trees so that each node of any color x has at least 2 neighbors colored with a color that is different from x), Δ-coloring trees of maximum degree Δ, etc.

6 Quantum advantage for a local problem

What do we know about quantum advantage in the LOCAL model? Before last year, there was only one example of quantum advantage: Le Gall, Nishimura, and Rosmanis [LNR19] showed that there exists a problem that requires $\Omega(n)$ communication rounds in input graphs with n nodes in classical LOCAL, but can be solved in $O(1)$ rounds in quantum-LOCAL. This problem, however, has an inherent global and artificial definition, and the winning condition depends on the joint output of nodes that are at distances $\Omega(n)$ between one another, which makes it very far from problems that are usually interesting for the distributed computing community, especially from LCLs. Last year, Balliu et al. [Bal+25] exhibited the first *local problem* Π that admits quantum advantage in the LOCAL model. The authors proved that Π is solvable in $O(1)$ rounds in quantum-LOCAL, but requires $\Theta(\Delta)$ rounds in classical LOCAL, where Δ is the degree of the input graph. Before describing the problem in details, let us remark that Π is locally checkable in all senses except that the maximum degree of the input graph is not bounded, hence it is not an LCL in the strict sense of Definition 2.2. Indeed, in case $\Delta = O(1)$, then there would be no asymptotic difference between the complexities in LOCAL and quantum-LOCAL. Let us now introduce the problem in multiple steps: first we introduce the GHZ game between three players, then we use it to build Π.

6.1 The GHZ game

The GHZ game is a game between three players that works as follows: Alice, Bob, and Charlie all receive by an adversary one input bit. The input (x, y, z) is drawn from the set $\{(0, 0, 0), (0, 1, 1), (1, 0, 1), (1, 1, 0)\}$, that is, the promise is that $x \oplus y \oplus z = 0$ ($\oplus$ is the notation for the XOR logical operator). The players must produce one output bit each, resulting in a tuple (a, b, c) such that $a \oplus b \oplus c = 0$ if and only if $(x, y, z) = (0, 0, 0)$ (see Table 1). Now, the game allows players to agree on a strategy *before* receiving the input (x, y, z), but *after* it they cannot communicate anymore. The best classical strategy for the three players to win this game is to deterministically output a tuple that wins in case $(x, y, z) \neq (0, 0, 0)$. However, in the quantum world there is a strategy that *always* wins the game. An uninterested reader might just skip the rest of this subsection, as the only notion that is useful for the rest of this article is that there exists a quantum strategy

that always wins the game. Otherwise, we assume the reader is familiar with basic notions of quantum computing: if not, we defer the reader to introductory surveys like, e.g., [BS98]. The players can share a tripartite entangled state $|\psi\rangle = \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle)$, known as the GHZ state. If a player receives 0 as input, it makes a measurement of the entangled state in the basis $\{|+\rangle, |-\rangle\}$. Otherwise, it makes a measurement in the basis $\{\frac{1}{\sqrt{2}}(|0\rangle + i\,|1\rangle), \frac{1}{\sqrt{2}}(|0\rangle - i\,|1\rangle)\}$. In both cases, the player outputs 0 if the result of the measurement is the first state, and 1 if it is the second state [BBT05].

Alice's (x) Input (x)	**Bob's (y) Input (y)**	**Charlie's (z) Input (z)**	**Winning Condition (Outputs)**
0	0	0	$a \oplus b \oplus c = 0$
0	1	1	$a \oplus b \oplus c = 1$
1	0	1	$a \oplus b \oplus c = 1$
1	1	0	$a \oplus b \oplus c = 1$

Table 1: GHZ game definition: the tuple (x, y, z) is the input and the output (a, b, c) must satisfy the winning condition.

6.2 Iterated GHZ games

The GHZ game usually comes with a promise that $x \oplus y \oplus z = 0$. In order to define our labeling problem, we need to remove this promise. We do so by relaxing the definition of the game: when $x \oplus y \oplus z = 1$ we allow any combination of outputs from Alice, Bob, and Charlie: clearly,this modification cannot make the problem harder. Now, our input is a bipartite graph where the two sets of the bipartition are the set W of *white nodes* and the set B of *black nodes*. White nodes are the real players of the games, while black nodes represent the games that the players are playing. More specifically, we assume that each white node has degree Δ and each black node has degree 3. We also assume that black nodes are colored with colors from $[\Delta]$, and that each white node has a unique neighbor colored with color c for all $c \in [\Delta]$. The colors of the black nodes specify the order according to which a white node must play the games. Every white node v must output a vector $[v_1, \ldots, v_\Delta]$ of Δ bits, where $v_i \in \{0, 1\}$ is its output to the game represented by its neighboring black node of color i. Let v be any black node colored with color $c \geq 1$, and s, t, u its three neighboring white nodes. The problem is defined as follows:

1. If $c = 1$, then exactly one among s_1, t_1, and u_1 must be equal to 1, and the others must be equal to 0.

2. If $c > 1$, then:

   (a) If $s_{c-1} \oplus t_{c-1} \oplus u_{c-1} = 0$, then s_c, t_c, and u_c must solve the GHZ game with input $(s_{c-1}, t_{c-1}, u_{c-1})$, that is, $s_c \oplus t_c \oplus u_c = 0$ if $s_{c-1}, t_{c-1}, u_{c-1} = (0, 0, 0)$, and $s_c \oplus t_c \oplus u_c = 1$ otherwise.

   (b) If $s_{c-1} \oplus t_{c-1} \oplus u_{c-1} = 1$, then s_c, t_c, and u_c can be arbitrarily chosen.

Clearly, in 1 communication round the problem can be solved in quantum-LOCAL as follows: The black nodes of color $c = 1$ inform their three neighbors s, t, and u on how to set s_1, t_1, and u_1 so that exactly one of these outputs is equal to 1. Black nodes of color $c > 1$ prepare 3 entangled GHZ states as described in Section 6.1 and send them to their three white neighbors, along with their colors. The white node will now have $\Delta - 1$ entangled states $|\psi_2\rangle, \ldots, |\psi_\Delta\rangle$, where $|\psi_c\rangle$ comes from the black neighbor of color c. Then, it can locally measure the entangled states, in order, setting all correct outputs without communicating further, as output number i depends only on $|\psi_i\rangle$ and output number $i - 1$.

In the classical setting, a trivial strategy would be the following: In round 1, black nodes of color 1 inform their white neighbors about which one should set its output to 1, while the others set their output to 0. In round 2, the white nodes inform their neighbors of color 2 about their output that has been set in the previous round. In this way, the three white neighbors s, t, and u of a black node v of color 2 have specified the inputs s_1, t_1, and u_1 for another GHZ game. Now v can locally solve the GHZ game with the new inputs and inform s, t, and u about their new outputs s_2, t_2, and u_2. Iterating this procedure, we have a solution to the problem in 2Δ rounds.

Interestingly, Balliu et al. [Bal+25] proved the following result.

Theorem 6.1 (Balliu et al. [Bal+25])**.** *The iterated GHZ problem can be solved in* 1 *round in quantum-LOCAL but requires* $\Omega(\min\{\Delta, \log_\Delta \log n\})$ *rounds in classical (deterministic or randomized) LOCAL.*

The classical lower bound in Theorem 6.1 is obtained through *round elimination* (RE), one of the most prominent lower bound techniques. Round elimination was formalized by Brandt [Bra19] but already used in a less general form to get an $\Omega(\log^\star n)$ lower bound for 3-coloring cycles by Linial [Lin92]. Nowadays, we even have an automated software that guides us into the round elimination procedure (i.e., REtor [Oli19]). Round elimination works as follows: Suppose an LCL problem Π has some complexity T, that is, there exists a T-round LOCAL algorithm $\mathcal{A}$ that solves Π, and T is the minimum integer with such property. Imagine

running $\mathcal{A}$ for $T-1$ rounds on a graph G. Nodes of G now will contain enough knowledge to be able to solve Π with just one more round of communication. Hence, one can describe exactly what this knowledge is and come up with the most general LCL problem Π_1 nodes can solve in $T-1$ rounds with $\mathcal{A}$. Now suppose we iterate this procedure for T rounds: We end up with an LCL Π_T that is solvable in 0 rounds of communication. However, T is usually unknown and the object of our investigation. Hence, we can guess the magnitude of T (say, T') and analyze $\Pi_{T'}$. If the description of $\Pi_{T'}$ is simple enough, it might be incredibly easier to understand if $\Pi_{T'}$ is solvable in 0 rounds: if not, then we get a lower bound T' on the complexity of Π. The usual challenge while performing round elimination is that the description of Π_i grows exponentially fast at each iteration, and thus it is fundamental to find *relaxations* of the problem that make Π_i easier to solve but hopefully with a simpler description. Note that while relaxing the description of a problem, one might exaggerate and end up with a problem that is just trivial to solve thwarting all the efforts to understand the complexity of the original problem. The whole procedure specific to the iterated GHZ problem can be found in [Bal+25].

We remark that round elimination cannot be used in the quantum world: apart from our result (which separates quantum-LOCAL and classical LOCAL), the *no-cloning* principle (that states that quantum states cannot be cloned [DCP16]) forbids us any kind of generalization of RE.

6.3 Networks of non-signaling games

After we established quantum advantage via Theorem 6.1, one may wonder whether with more refined games we could achieve even a stronger separation. In this section we show that, unfortunately, *this is not possible*.

Definition 6.2 (Game)**.** Let Σ be a finite set, and let $m \in \mathbb{N}_+$ be the number of players. We call $\mathfrak{g} \subseteq \Sigma^m \times \Sigma^m$ a *game*. Each player i receives one input $x_i \in \Sigma$ and produces one output $y_i \in \Sigma$. A *move* $\mu = (x, y) \in \Sigma^m \times \Sigma^m$ is valid if $\mu \in \mathfrak{g}$. We overload the notation so that $\mathfrak{g}(x) = \{y \in \Sigma^m \mid (x, y) \in \mathfrak{g}\}$. We say $\mathfrak{g}$ is *solvable* if, for every x, $\mathfrak{g}(x)$ is non-empty.

Non-signaling games. Let $\mathfrak{g}$ be any game of m players. We can define the following labeling problem Π on any graph $G = (V, E)$ with $|V| \geq m$. In input, a subset $P \subseteq V$ with $|P| = m$ is chosen to represent the set of players. Those nodes receive an input bit and must output another bit so that their joint output solves $\mathfrak{g}$. All other nodes can output any bit. We say that $\mathfrak{g}$ is non-signaling if there exists a non-signaling outcome that solves Π with locality 0. Clearly, Π is not locally checkable, but if we add the further constraint that the diameter of $G[P]$ is

bounded by a constant, then we can make it locally checkable (even if the degree of the graph might be unbounded).

Network of non-signaling games. We are given a bipartite graph where the two sets of the bipartition are the set W of *white nodes* and the set B of *black nodes*. White nodes are the real players of the games, while black nodes represent the games that the players are playing. We assume that each white node has degree Δ and each black node has degree m. Each black node represents a non-signaling game that its white neighbors must play. Also, white nodes can contain arbitrarily complex arithmetic circuits according to which they need to play the games: in a sense, the input of a game might depend arbitrarily complex arithmetic operations on the outputs of previous games. Such a network is called *network of non-signaling games*: it includes networks that one can build with quantum games such as the network in the iterated GHZ problem, and is locally checkable (possibly with unbounded degree).

[Bal+25] proved the following theorem.

Theorem 6.3 (Balliu et al. [Bal+25])**.** *Let $\Delta \in \mathbb{N}_+$ be a constant. For any network of non-signaling games of maximum degree Δ, there exists a classical LOCAL algorithm solving the games in time $O(1)$, where the constant might depend on Δ.*

The key ingredient for Theorem 6.3 is that any non-signaling game is completable: for example, for a 2-player game this means that for any Alice's input x there is some Alice's output a such that for any Bob's input y there is still a valid Bob's output b, and vice versa. We can exploit completability to solve any network of non-signaling games in a distributed manner: Each white node starts to process its own arithmetic circuit in a topological order. As soon as it encounters a step that involves a game, it sends a message to the black neighbor responsible for that specific game, together with its own input for that game. The black nodes keep track of the inputs they have seen so far, and they always pick safe outputs for those players. In this way, in two rounds of communication all white nodes can learn their own output for the game that appears first in their own circuit. We can then repeat this for each game in a sequential order—thanks to completability, while black nodes will be always able to assign valid outputs also for players that join the game late. The running time of this algorithm is proportional to the size of the circuit held by a single white node: for a fixed LCL (with a finite set of possible local circuits) it will be bounded by some constant. With this kind of arguments all we can hope for is a separation that is a function $f(\Delta)$ of the degree of the graph, but we still do not have an LCL problem (in the strict sense) separating LOCAL and quantum-LOCAL by a function $f(n)$ of the number of nodes of the graph n. We conclude this brief survey with the major open question in the field.

Question 9 (Open)**.** *Is there any LCL problem that quantum-LOCAL can solve asymptotically faster (as a function of the number of nodes) than classical LOCAL?*

References

[Aar22] Scott Aaronson. "How Much Structure Is Needed for Huge Quantum Speedups?" In: *CoRR* abs/2209.06930 (2022). DOI: 10.48550/ARXIV.2209.06930.

[Akb+25] Amirreza Akbari, Xavier Coiteux-Roy, Francesco D'Amore, François Le Gall, Henrik Lievonen, Darya Melnyk, Augusto Modanese, Shreyas Pai, Marc-Olivier Renou, Václav Rozhon, and Jukka Suomela. "Online Locality Meets Distributed Quantum Computing". In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, CZ, Czech Republic, June 23-27, 2025*. ACM, 2025.

[Akb+23] Amirreza Akbari, Navid Eslami, Henrik Lievonen, Darya Melnyk, Joona Särkijärvi, and Jukka Suomela. "Locality in Online, Dynamic, Sequential, and Distributed Graph Algorithms". In: *50th International Colloquium on Automata, Languages, and Programming, ICALP 2023, July 10-14, 2023, Paderborn, Germany*. Ed. by Kousha Etessami, Uriel Feige, and Gabriele Puppis. Vol. 261. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023, 10:1–10:20. DOI: 10.4230/LIPICS.ICALP.2023.10.

[AV22] Joran van Apeldoorn and Tijn de Vos. "A Framework for Distributed Quantum Queries in the CONGEST Model". In: *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*. Ed. by Alessia Milani and Philipp Woelfel. ACM, 2022, pp. 109–119. DOI: 10.1145/3519270.3538413.

[AF14] Heger Arfaoui and Pierre Fraigniaud. "What can be computed without communications?" In: *SIGACT News* 45.3 (2014), pp. 82–104. DOI: 10.1145/2670418.2670440.

[Bal+19a] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. "The Distributed Complexity of Locally Checkable Problems on Paths is Decidable". In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. Ed. by Peter Robinson and Faith Ellen. ACM, 2019, pp. 262–271. DOI: 10.1145/3293611.3331606.

[Bal+22a] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studený, and Jukka Suomela. "Efficient Classification of Locally Checkable Problems in Regular Trees". In: *36th International Symposium on Distributed Computing, DISC 2022, October 25-27, 2022, Augusta, Georgia, USA*. Ed. by Christian Scheideler. Vol. 246. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 8:1–8:19. DOI: 10.4230/LIPICS.DISC.2022.8.

[Bal+23a] Alkida Balliu, Sebastian Brandt, Yi-Jun Chang, Dennis Olivetti, Jan Studený, Jukka Suomela, and Aleksandr Tereshchenko. "Locally checkable problems in rooted trees". In: *Distributed Computing* 36.3 (2023), pp. 277–311. DOI: 10.1007/S00446-022-00435-9.

[Bal+25] Alkida Balliu, Sebastian Brandt, Xavier Coiteux-Roy, Francesco D'Amore, Massimo Equi, François Le Gall, Henrik Lievonen, Augusto Modanese, Dennis Olivetti, Marc-Olivier Renou, Jukka Suomela, Lucas Tendick, and Isadora Veeren. "Distributed Quantum Advantage for Local Problems". In: (2025).

[Bal+19b] Alkida Balliu, Sebastian Brandt, Juho Hirvonen, Dennis Olivetti, Mikaël Rabie, and Jukka Suomela. "Lower Bounds for Maximal Matchings and Maximal Independent Sets". In: *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*. Ed. by David Zuckerman. IEEE Computer Society, 2019, pp. 481–497. DOI: 10.1109/FOCS.2019.00037.

[Bal+20] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. "How much does randomness help with locally checkable problems?" In: *PODC '20: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, August 3-7, 2020*. Ed. by Yuval Emek and Christian Cachin. ACM, 2020, pp. 299–308. DOI: 10.1145/3382734.3405715.

[Bal+21a] Alkida Balliu, Sebastian Brandt, Dennis Olivetti, and Jukka Suomela. "Almost global problems in the LOCAL model". In: *Distributed Computing* 34.4 (2021), pp. 259–281. DOI: 10.1007/S00446-020-00375-2.

[Bal+21b] Alkida Balliu, Keren Censor-Hillel, Yannic Maus, Dennis Olivetti, and Jukka Suomela. "Locally Checkable Labelings with Small Messages". In: *35th International Symposium on Distributed Computing, DISC 2021, October 4-8, 2021, Freiburg, Germany (Virtual Conference)*. Ed. by Seth Gilbert. Vol. 209. LIPIcs. Schloss Dagstuhl

- Leibniz-Zentrum für Informatik, 2021, 8:1–8:18. DOI: 10.4230/LIPICS.DISC.2021.8.

[Bal+24] Alkida Balliu, Mohsen Ghaffari, Fabian Kuhn, Augusto Modanese, Dennis Olivetti, Mikaël Rabie, Jukka Suomela, and Jara Uitto. "Shared Randomness Helps with Local Distributed Problems". In: *CoRR* abs/2407.05445 (2024). DOI: 10.48550/ARXIV.2407.05445.

[Bal+18] Alkida Balliu, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. "New classes of distributed time complexity". In: *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*. Ed. by Ilias Diakonikolas, David Kempe, and Monika Henzinger. ACM, 2018, pp. 1307–1318. DOI: 10.1145/3188745.3188860.

[Bal+22b] Alkida Balliu, Juho Hirvonen, Darya Melnyk, Dennis Olivetti, Joel Rybicki, and Jukka Suomela. "Local Mending". In: *Structural Information and Communication Complexity - 29th International Colloquium, SIROCCO 2022, Paderborn, Germany, June 27-29, 2022, Proceedings*. Ed. by Merav Parter. Vol. 13298. Lecture Notes in Computer Science. Springer, 2022, pp. 1–20. DOI: 10.1007/978-3-031-09993-9_1.

[Bal+23b] Alkida Balliu, Janne H. Korhonen, Fabian Kuhn, Henrik Lievonen, Dennis Olivetti, Shreyas Pai, Ami Paz, Joel Rybicki, Stefan Schmid, Jan Studený, Jukka Suomela, and Jara Uitto. "Sinkless Orientation Made Simple". In: *2023 Symposium on Simplicity in Algorithms, SOSA 2023, Florence, Italy, January 23-25, 2023*. Ed. by Telikepalli Kavitha and Kurt Mehlhorn. SIAM, 2023, pp. 175–191. DOI: 10.1137/1.9781611977585.CH17.

[BS98] Charles H. Bennett and Peter W. Shor. "Quantum Information Theory". In: *IEEE Trans. Inf. Theory* 44.6 (1998), pp. 2724–2742. DOI: 10.1109/18.720553.

[Bog13] Ilya I. Bogdanov. "Examples of topologically highly chromatic graphs with locally small chromatic number". In: *CoRR* abs/1311.2844 (2013). DOI: 10.48550/arXiv.1311.2844. arXiv: 1311.2844.

[Bra19] Sebastian Brandt. "An Automatic Speedup Theorem for Distributed Problems". In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. Ed. by Peter Robinson and Faith Ellen. ACM, 2019, pp. 379–388. DOI: 10.1145/3293611.3331611.

[Bra+16] Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. "A lower bound for the distributed Lovász local lemma". In: *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*. Ed. by Daniel Wichs and Yishay Mansour. ACM, 2016, pp. 479–488. DOI: 10.1145/2897518.2897570.

[Bra+17] Sebastian Brandt, Juho Hirvonen, Janne H. Korhonen, Tuomo Lempiäinen, Patric R. J. Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemyslaw Uznanski. "LCL Problems on Grids". In: *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*. Ed. by Elad Michael Schiller and Alexander A. Schwarzmann. ACM, 2017, pp. 101–110. DOI: 10.1145/3087801.3087833.

[BBT05] Gilles Brassard, Anne Broadbent, and Alain Tapp. "Quantum Pseudo-Telepathy". In: *Foundations of Physics* 35 (2005), pp. 1877–1907. DOI: 10.1007/s10701-005-7353-4.

[Cen+22] Keren Censor-Hillel, Orr Fischer, François Le Gall, Dean Leitersdorf, and Rotem Oshman. "Quantum Distributed Algorithms for Detection of Cliques". In: *13th Innovations in Theoretical Computer Science Conference, ITCS 2022, January 31 - February 3, 2022, Berkeley, CA, USA*. Ed. by Mark Braverman. Vol. 215. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 35:1–35:25. DOI: 10.4230/LIPICS.ITCS.2022.35.

[Cha20] Yi-Jun Chang. "The Complexity Landscape of Distributed Locally Checkable Problems on Trees". In: *34th International Symposium on Distributed Computing, DISC 2020, October 12-16, 2020, Virtual Conference*. Ed. by Hagit Attiya. Vol. 179. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 18:1–18:17. DOI: 10.4230/LIPIcs.DISC.2020.18.

[CKP16] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. "An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model". In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. Ed. by Irit Dinur. IEEE Computer Society, 2016, pp. 615–624. DOI: 10.1109/FOCS.2016.72.

[CKP19] Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. "An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model". In: *SIAM Journal on Computing* 48.1 (2019), pp. 122–143. DOI: 10.1137/17M1117537.

[CP19] Yi-Jun Chang and Seth Pettie. "A Time Hierarchy Theorem for the LOCAL Model". In: *SIAM Journal on Computing* 48.1 (2019), pp. 33–69. DOI: 10.1137/17M1157957.

[Coi+24] Xavier Coiteux-Roy, Francesco D'Amore, Rishikesh Gajjala, Fabian Kuhn, François Le Gall, Henrik Lievonen, Augusto Modanese, Marc-Olivier Renou, Gustav Schmid, and Jukka Suomela. "No Distributed Quantum Advantage for Approximate Graph Coloring". In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*. Ed. by Bojan Mohar, Igor Shinkar, and Ryan O'Donnell. ACM, 2024, pp. 1901–1910. DOI: 10.1145/3618260.3649679.

[DCP16] Giacomo Mauro D'Ariano, Giulio Chiribella, and Paolo Perinotti. *Quantum Theory from First Principles: An Informational Approach*. Cambridge University Press, 2016. DOI: 10.1017/9781107338340.

[Der+08] Bilel Derbel, Cyril Gavoille, David Peleg, and Laurent Viennot. "On the locality of distributed sparse spanner construction". In: *Proceedings of the Twenty-Seventh Annual ACM Symposium on Principles of Distributed Computing, PODC 2008, Toronto, Canada, August 18-21, 2008*. Ed. by Rida A. Bazzi and Boaz Patt-Shamir. ACM, 2008, pp. 273–282. DOI: 10.1145/1400751.1400788.

[Dha+24] Anubhav Dhar, Eli Kujawa, Henrik Lievonen, Augusto Modanese, Mikail Muftuoglu, Jan Studený, and Jukka Suomela. "Local Problems in Trees Across a Wide Range of Distributed Models". In: *28th International Conference on Principles of Distributed Systems, OPODIS 2024, December 11-13, 2024, Lucca, Italy*. Ed. by Silvia Bonomi, Letterio Galletta, Etienne Rivière, and Valerio Schiavoni. Vol. 324. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024, 27:1–27:17. DOI: 10.4230/LIPICS.OPODIS.2024.27.

[Elk07] Michael Elkin. "A near-optimal distributed fully dynamic algorithm for maintaining sparse spanners". In: *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*. Ed. by Indranil Gupta and Roger Wattenhofer. ACM, 2007, pp. 185–194. DOI: 10.1145/1281100.1281128.

[FG17] Manuela Fischer and Mohsen Ghaffari. "Sublogarithmic Distributed Algorithms for Lovász Local Lemma, and the Complexity Hierarchy". In: *31st International Symposium on Distributed Computing, DISC 2017, October 16-20, 2017, Vienna, Austria*. Ed. by Andréa W. Richa. Vol. 91. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 18:1–18:16. DOI: 10.4230/LIPICS.DISC.2017.18.

[Fra+24] Pierre Fraigniaud, Maël Luce, Frédéric Magniez, and Ioan Todinca. "Even-Cycle Detection in the Randomized and Quantum CONGEST Model". In: *Proceedings of the 43rd ACM Symposium on Principles of Distributed Computing, PODC 2024, Nantes, France, June 17-21, 2024*. Ed. by Ran Gelles, Dennis Olivetti, and Petr Kuznetsov. ACM, 2024, pp. 209–219. DOI: 10.1145/3662158.3662767.

[Gav+09] Cyril Gavoille, Ralf Klasing, Adrian Kosowski, Lukasz Kuszner, and Alfredo Navarra. "On the complexity of distributed graph coloring with local minimality constraints". In: *Networks* 54.1 (2009), pp. 12–19. DOI: 10.1002/NET.20293.

[Gav+07] Cyril Gavoille, Ralf Klasing, Adrian Kosowski, and Alfredo Navarra. "On the Complexity of Distributed Greedy Coloring". In: *Distributed Computing, 21st International Symposium, DISC 2007, Lemesos, Cyprus, September 24-26, 2007, Proceedings*. Ed. by Andrzej Pelc. Vol. 4731. Lecture Notes in Computer Science. Springer, 2007, pp. 482–484. DOI: 10.1007/978-3-540-75142-7_37.

[GKM09] Cyril Gavoille, Adrian Kosowski, and Marcin Markiewicz. "What Can Be Observed Locally?" In: *Distributed Computing, 23rd International Symposium, DISC 2009, Elche, Spain, September 23-25, 2009. Proceedings*. Ed. by Idit Keidar. Vol. 5805. Lecture Notes in Computer Science. Springer, 2009, pp. 243–257. DOI: 10.1007/978-3-642-04355-0_26.

[GHK18] Mohsen Ghaffari, David G. Harris, and Fabian Kuhn. "On Derandomizing Local Distributed Algorithms". In: *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*. Ed. by Mikkel Thorup. IEEE Computer Society, 2018, pp. 662–673. DOI: 10.1109/FOCS.2018.00069.

[GKM17] Mohsen Ghaffari, Fabian Kuhn, and Yannic Maus. "On the complexity of local distributed graph problems". In: *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*. Ed. by Hamed Hatami,

Pierre McKenzie, and Valerie King. ACM, 2017, pp. 784–797. DOI: 10.1145/3055399.3055471.

[GRB22] Christoph Grunau, Václav Rozhon, and Sebastian Brandt. "The Landscape of Distributed Complexities on Trees and Beyond". In: *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*. Ed. by Alessia Milani and Philipp Woelfel. ACM, 2022, pp. 37–47. DOI: 10.1145/3519270.3538452.

[Hir+17] Juho Hirvonen, Joel Rybicki, Stefan Schmid, and Jukka Suomela. "Large Cuts with Local Algorithms on Triangle-Free Graphs". In: *The Electronic Journal of Combinatorics* 24.4 (2017), p. 4. DOI: 10.37236/6862.

[HHL18] Alexander E. Holroyd, Tom Hutchcroft, and Avi Levy. "Finitely dependent cycle coloring". In: *Electronic Communications in Probability* 23 (2018). DOI: 10.1214/18-ecp118.

[HL16] Alexander E. Holroyd and Thomas M. Liggett. "Finitely Dependent Coloring". In: *Forum of Mathematics, Pi* 4, e9 (2016). DOI: 10.1017/fmp.2016.7.

[IL19] Taisuke Izumi and François Le Gall. "Quantum Distributed Algorithm for the All-Pairs Shortest Path Problem in the CONGEST-CLIQUE Model". In: *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*. Ed. by Peter Robinson and Faith Ellen. ACM, 2019, pp. 84–93. DOI: 10.1145/3293611.3331628.

[ILM20] Taisuke Izumi, François Le Gall, and Frédéric Magniez. "Quantum Distributed Algorithm for Triangle Finding in the CONGEST Model". In: *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*. Ed. by Christophe Paul and Markus Bläser. Vol. 154. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 23:1–23:13. DOI: 10.4230/LIPICS.STACS.2020.23.

[KSV13] Amos Korman, Jean-Sébastien Sereni, and Laurent Viennot. "Toward more localized local algorithms: removing assumptions concerning global knowledge". In: *Distributed Computing* 26.5-6 (2013), pp. 289–308. DOI: 10.1007/S00446-012-0174-8.

[KMW04] Fabian Kuhn, Thomas Moscibroda, and Roger Wattenhofer. "What cannot be computed locally!" In: *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*.

Ed. by Soma Chaudhuri and Shay Kutten. ACM, 2004, pp. 300–309. DOI: 10.1145/1011767.1011811.

[LM18] François Le Gall and Frédéric Magniez. "Sublinear-Time Quantum Computation of the Diameter in CONGEST Networks". In: *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*. ACM, 2018, pp. 337–346. DOI: 10.1145/3212734.3212744.

[LNR19] François Le Gall, Harumichi Nishimura, and Ansis Rosmanis. "Quantum Advantage for the LOCAL Model in Distributed Computing". In: *36th International Symposium on Theoretical Aspects of Computer Science, STACS 2019, March 13-16, 2019, Berlin, Germany*. Ed. by Rolf Niedermeier and Christophe Paul. Vol. 126. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 49:1–49:14. DOI: 10.4230/LIPICS.STACS.2019.49.

[Lin87] Nathan Linial. "Distributive Graph Algorithms-Global Solutions from Local Data". In: *28th Annual Symposium on Foundations of Computer Science, Los Angeles, California, USA, 27-29 October 1987*. IEEE Computer Society, 1987, pp. 331–335. DOI: 10.1109/SFCS.1987.20.

[Lin92] Nathan Linial. "Locality in Distributed Graph Algorithms". In: *SIAM Journal on Computing* 21.1 (1992), pp. 193–201. DOI: 10.1137/0221015.

[MN22] Frédéric Magniez and Ashwin Nayak. "Quantum Distributed Complexity of Set Disjointness on a Line". In: *ACM Transactions on Computation Theory* 14.1 (2022), 5:1–5:22. DOI: 10.1145/3512751.

[MS02] Bojan Mohar and Paul D. Seymour. "Coloring Locally Bipartite Graphs on Surfaces". In: *J. Comb. Theory B* 84.2 (2002), pp. 301–310. DOI: 10.1006/JCTB.2001.2086.

[MST13] Bojan Mohar, Gábor Simonyi, and Gábor Tardos. "Local chromatic number of quadrangulations of surfaces". In: *Combinatorica* 33.4 (2013), pp. 467–495. DOI: 10.1007/S00493-013-2771-Y.

[Nao91] Moni Naor. "A Lower Bound on Probabilistic Algorithms for Distributive Ring Coloring". In: *SIAM Journal on Discrete Mathematics* 4.3 (1991), pp. 409–412. DOI: 10.1137/0404036.

[NS93] Moni Naor and Larry J. Stockmeyer. "What can be computed locally?" In: *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*. Ed. by S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal. ACM, 1993, pp. 184–193. DOI: 10.1145/167088.167149.

[NS95] Moni Naor and Larry J. Stockmeyer. "What Can be Computed Locally?" In: *SIAM Journal on Computing* 24.6 (1995), pp. 1259–1277. DOI: 10.1137/S0097539793254571.

[Oli19] Dennis Olivetti. *Round Eliminator: a tool for automatic speedup simulation*. 2019. URL: https://github.com/olidennis/round-eliminator.

[WWY21] Changsheng Wang, Xudong Wu, and Penghui Yao. "Complexity of Eccentricities and All-Pairs Shortest Paths in the Quantum CONGEST Model". In: *SPIN* 11.03 (2021), p. 2140007. DOI: 10.1142/S2010324721400075. eprint: https://doi.org/10.1142/S2010324721400075.

[WY22] Xudong Wu and Penghui Yao. "Quantum Complexity of Weighted Diameter and Radius in CONGEST Networks". In: *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*. Ed. by Alessia Milani and Philipp Woelfel. ACM, 2022, pp. 120–130. DOI: 10.1145/3519270.3538441.

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 145

Range Avoidance and the Complexity of Explicit Constructions

Oliver Korten
Department of Computer Science
Columbia University, New York City
oliver.korten@columbia.edu

Abstract

A recent line of work has investigated the complexity of explicit construction problems through the study of a search problem known as *Range Avoidance*: given as input a boolean circuit $C : \{0,1\}^n \to \{0,1\}^{n+1}$, find an element $y \in \{0,1\}^{n+1}$ outside of its range. Analysis of this search problem and its variants has lead to several exciting new results in derandomization and circuit complexity. In this survey we give an overview of this nascent research direction and its connections to some old and fundamental questions in complexity theory.

1 Introduction

1.1 Motivation: Explicit Constructions

Throughout combinatorics and computer science, a fundamental tool used to demonstrate the existence of interesting combinatorial objects is the *probabilistic method*. Rather than constructing an explicit example of the desired object, a random distribution of objects is chosen, and the probability of the desired object arising from the distribution is shown to be strictly positive. The first appearance of this argument was in a classical paper of Erdős who used it to establish the existence of so-called *Ramsey graphs*: n vertex undirected graphs whose largest clique and largest independent set each have size at most $2 \log n$ [15]. A few years later, an essentially identical argument was used by Shannon [42] to demonstrate the existence of functions $f : \{0,1\}^n \to \{0,1\}$ requiring boolean circuits of size $\geq 2^n/n$. In each case, the probabilistic argument gives no indication as to a *particular example* of the object in question, and a signficant body of research over the past eight decades has been devoted to matching these nonconstructive bounds

with explicit arguments. Similar phenomenon have become ubiquitous throughout combinatorics and computer science, whereby a nonconstructive argument proving the existence of an interesting class of object has spawned a deep line of work dedicated to producing a concrete example of one such object. In some fortunate cases, such as Ramsey graphs, error correcting codes, and expanders, decades of hard work eventually lead to optimal or near-optimal solutions. In many other cases no significant progress has been made to date.

The pursuit of *explicit constructions* in place of nonconstructive arguments has several motivations, which may be divided into roughly two classes: *informal/philosophical* and *formal/computational*. The informal motivation can be summarized by a maxim asserting that we do not fully understand a concept until we can produce an example. When we have some combinatorial property, the mere existence of objects satisfying that property does not give us any deeper information about *why* a certain object achieves the property. If we succeed in the quest of finding an explicit object, and have a mathematical proof that this object *in particular* has the relevant property, this proof will usually contain within it a furtherance of our general understanding of the concept in question.

This informal motivation utilizes a rather vague definition of the term "explicit." In this framework, we would call an object, such as a Ramsey graph $G \subseteq \binom{[n]}{2}$, "explicit" provided we can present it by a comprehensible definition from which we can extract and compute various other properties of the object by direct mathematical argument. The class of *formal/computational* motivations for studying explicit constructions centers itself on a more concrete definition of the notion of "explicitness:" in this context we will say that an object is explicit if it can be produced by an efficient algorithm. In the example of a Ramsey graph $G \subseteq \binom{[n]}{2}$, we might say that it is explicit if there is a poly(n) time algorithm which outputs its adjacency matrix given n; embedded in this kind of definition is the necessity of reserving the notion of explicitness for *sequences of objects* rather than individuals.

The motivations for the study of *computationally explicit* constructions can be further subdivided into two categories, which turn out to be intimately connected: *derandomization* and *computational lower bounds*. Derandomization is a field dedicated to the study of when randomized algorithms can be replaced by efficient deterministic algorithms posessing similar behavior. At a high level, the connection to explicit constructions may be viewed as follows. Say we have a randomized algorithm $\mathcal{A}$ that flips n random coins, which we hope to simulate deterministically. We may think of $\mathcal{A}(r)$ as a deterministic algorithm which takes an n-bit string r, and has some desirable behavior with high probability when r is chosen uniformly. Peering into the proof of $\mathcal{A}$'s correctness, it is often possible to isolate a *particular pseudorandom property* of n-bit strings, call it $\Pi \subseteq \{0,1\}^n$, so that a random string r has the property Π with high probability, and whenever $r \in \Pi$, $\mathcal{A}(r)$ has the correct behavior. Hence, if we could somehow *deterministically* construct a fixed string

$r \in \{0,1\}^n$ which has the necessary property Π, then we could reap the benefits of our randomized algorithm while using this fixed string in place of true randomness. The problem of constructing strings with property Π is then the relevant explicit construction problem.

In the case of computational lower bounds, the connection to explicit constructions is most direct. A fundamental problem in complexity theory is to show separations of the form $\mathcal{A} \nsubseteq C$ where $\mathcal{A}$ is a uniform complexity class such as $\mathsf{NP}, \mathsf{PSPACE}, \mathsf{EXP}$, and C is a *nonuniform* class of circuits, such as $\mathsf{AC}^0, \mathsf{TC}^0, \mathsf{P/poly}$. By a counting argument, it is usually straightforward to show that there exists boolean functions $f : \{0,1\}^n \to \{0,1\}$ which are hard for the class C. We may then study the explicit construction problem of producing an example of such a function f; if the construction itself has bounded uniform complexity in some class $\mathcal{A}$, then this implies by definition that $\mathcal{A} \nsubseteq C$. While this connection may sound rather tautological, the study of computational lower bounds through the lens of explicit construction has proved quite useful, particularly in the case that the uniform classes $\mathcal{A}$ is at least as large as EXP.

1.2 The "Right" Complexity Class for Explicit Constructions

The main topic of this survey is the Range Avoidance problem: given a boolean circuit $C : \{0,1\}^n \to \{0,1\}^m$ with $m > n$, find a string $y \in \{0,1\}^{n+1}$ outside the range of C. The primary motivation for studying Range Avoidance stems from its ability to capture the computational complexity of the kinds of explicit construction problems discussed in the last section [29]. This connection is based on the simple (and well-known) observation that most probabilistic existence arguments can be strengthened to *encoding arguments*: to show that most objects have a desired property, we show how any "bad object" failing to satisfy the property can be coded by description of nontrivial length. In an information theoretic sense this claim has no content: obviously every set of size 2^k can have its elements coded by k-bit strings. However, we shall see that if an encoding can be defined for which the *decoder* has an efficient algorithm, this will yield a reduction to Range Avoidance.

It is easiest to work through an example, and perhaps the simplest is the case of Ramsey graphs mentioned above. We start with the standard probabilistic argument which shows the existence of a k-Ramsey graph on n vertices (no clique or independent set of size $\geq k$) whenever $k >> \log n$. Choose $G \sim \{0,1\}^{\binom{n}{2}}$ uniformly at random. For each set $V \subseteq [n]$ of size k, the probability that $G_{u,v} = 1$ for all $u, v \in V$ is $2^{-\binom{k}{2}}$; a symmetric argument gives the same bound in the case $G_{u,v} = 0$. Now, taking a union bound over all choices of V and the choice of whether V is a clique or an independent set, the probability any clique or independent set

exists is at most

$$2 \cdot \binom{n}{k} \cdot 2^{-\binom{k}{2}} \leq 2^{1+k\log n-\binom{k}{2}}$$

Hence setting $k \geq (2 + o(1)) \cdot \log n$, this probability is strictly below 1 for n sufficiently large.

Now, say that we had a graph G and a clique or independent set V of size k in G. Using the exact same analysis, we can form a concise encoding of G as follows:

1. 1 bit indicating whether V is a clique or independent set.
2. $k \log n$ bits describing the set V as a list of vertices.
3. $\binom{n}{2} - \binom{k}{2}$ bits describing G on all edges $\{u, v\}$ with $\{u, v\} \nsubseteq V$.

It is clear that from this description we could uniquely decode the graph G in polynomial time. The total number of bits is $1 + k \log n + \binom{n}{2} - \binom{k}{2}$, which is strictly less than $\binom{n}{2}$ provided $k \geq (2 + o(1)) \log n$. Hence if we define

$$C : \{0,1\}^{\binom{n}{2}+1+k\log n-\binom{k}{2}} \to \{0,1\}^{\binom{n}{2}}$$

as the function which takes such a concise graph description and outputs the graph G being described, we see that any graph which fails to be k-Ramsey will lie in the range of C; hence any $G \notin \text{range}(C)$ is a solution to our explicit construction problem. Since C is computable in $\text{poly}(n)$ time, we can efficiently construct a boolean circuit computing C in time $\text{poly}(n)$ given n. Since the number of input bits of C is strictly less then the number of outputs, C is a valid instance of the Avoid problem and we have succeeded in reducing our explicit construction problem to Range Avoidance.

1.3 Road Map

Since its introduction in [28] and its investigation in connection to explicit construction problems in [29], Range Avoidance has received considerable attention in the past few years [9–11, 18, 21, 23, 30, 31, 35, 41, 43]. At a very high level, some of the key takeaways from this line of work are:

1. Essentially all interesting and unsolved explicit construction problems found throughout theoretical computer science can be reduced in polynomial time to Avoid using a small toolkit of reduction techniques.
2. Several important explicit construction problems actually reduce to a special case of Avoid in which the input circuit $C : \{0,1\}^n \to \{0,1\}^m$ has a special

form. In other cases, nontrivial algorithms for special cases of Avoid have been found unconditionally. Unfortunately, the known results of this form do not yet "match up" to generate new explicit constructions.

3. There are nontrivial conditional and unconditional upper bounds for Avoid inside of the polynomial hierarchy, one of which has lead to a recent breakthrough circuit lower bound.

This survey will attempt to cover the current state of knowledge surrounding this intriguing search problem, and some of its broader implications in complexity theory. The content of this article is laid out as follows. In Section 2 we describe the basic formulation of explicit construction problems, the connection between explicit constructions and circuit lower bounds, and make some preliminary observations about the complexity of Avoid. In Section 3 we survey the main high level techniques used to reduce various explicit construction problems to Avoid. In Section 4 we discuss reductions between different variants of the Avoid problem. In Section 5 we discuss some more involved *upper bounds* for Range Avoidance, some of which have important implications in circuit complexity. In Section 6 we cover some results indicating the *hardness* of Avoid, both in the white-box model (cryptographic hardness) and in the black box model (oracle separations). In Section 7 we discuss work on an easier variant of Avoid lying in TFNP called *Lossy Code*. Finally in Section 8 we present some important open problems.

1.4 Background– Bounded Arithmetic

The investigations discussed in this survey are deeply connected to, and in some cases directly inspired by, important work in the field of *Bounded Arithmetic*. Bounded Arithmetic studies the strength of subtheories of Peano Arithmetic whose induction principle is restricted to formulas of bounded computational complexity; for formal definitions and a comprehensive introduction to the area see [32]. An important early question in the field was whether the theory $I\Delta_0$ could prove the existence of infinitely many primes; in complexity-theoretic terminology, we may think of $I\Delta_0$ as the fragment of Peano Arithmetic with induction for formulas decidable in the linear time hierarchy:

$$\mathsf{LH} = \bigcup_{i,c\in\mathbb{N}} \Sigma_i\text{-}\mathsf{Time}[c \cdot n]$$

To resolve this question, [39] took the following approach: first isolate an abstract combinatorial principle which suffices to prove the existence of infinitely many primes in $I\Delta_0$, then give an $I\Delta_0$ proof of this principle[1]. The combinatorial principle

[1] In fact they could only prove this principle in a mild extension of $I\Delta_0$ now known as T_2.

used in [39] was the *weak pigeonhole principle*, which states that no formula in the language of $I\Delta_0$ can define an injective map $2^{n+1} \mapsto 2^n$; in a theory as strong as $I\Delta_0$ this is equivalent to the statement that no formula defines a surjective map $2^n \mapsto 2^{n+1}$. This work was influential in at least two respects. First, the technique used to prove the weak pigeonhole principle in $I\Delta_0$ based on repeated composition of a supposed surjection $2^n \mapsto 2^{n+1}$ has been applied repeatedly in later work, as discussed in Section 5.1. Second, the work of [39] highlighted the importance of the weak pigeonhole principle as a key lemma from which highly nontrivial results in number theory and combinatorics could be derived via the use of counting arguments.

This latter direction was taken to its logical extreme in the highly influential thesis of Jeřábek [25]. In this work, Jeřábek showed that a theory of bounded arithmetic possessing induction for polynomial time computable predicates together with the *dual weak pigeonhole principle for polynomial time functions* could formalize a wide array of difficult complexity theoretic arguments that utilize randomness and probability. This variant of the pigeonhole principle essentially asserts that the problem Avoid is a *total search problem*: any instance $C : \{0,1\}^n \to \{0,1\}^{n+1}$ of Avoid defined by a boolean circuit must have a solution. In particular, Jeřábek was able to show this theory could define and analyze many key properties of *randomized algorithms*. Jeřábek also showed that the dual weak pigeonhole principle defining his theory could be replaced by a principle asserting the existence of boolean functions $f : \{0,1\}^n \to \{0,1\}$ of high circuit complexity; this result plays an important role in the study of Avoid, and was used in [29] to characterize the complexity of Avoid in terms of a computational hardness assumption; this is discussed further in Section 5.2.

2 Preliminaries

In this section we introduce basic definitions related to explicit construction problems and Avoid.

2.1 Formalizing Explicit Construction

We start with a formal definition of an explicit construction problem:

Definition 1. *An explicit construction problem is defined by a language* $\Pi \subseteq \{0,1\}^*$. *We use* Π_n *to denote* $\Pi \cap \{0,1\}^n$. *The explicit construction problem* EC_Π *associated to* Π *is: given* 1^n, *output* $x \in \Pi_n$ *(or determine that none exist).*

We say that Π *is total if* $\Pi_n \neq \emptyset$ *for all n; we say that it is dense if there is a fixed c so that* $|\Pi_n|/2^n \geq 1/n^c$ *for all n.*

We will sometimes abuse notation and refer to Π directly as both a language and its associated explicit construction problem. In all cases of interest, Π is total, since an explicit construction problem only presents itself once the question of existence has already been settled. The reason for the terminology of "totality" is that in such cases EC_Π is a *total search problem*: every instance has a solution. We define the input to the search problem as being encoded in unary (i.e. 1^n) so that the complexity of algorithms solving it is measured as a function of n rather than $\log n$: ideally, we seek algorithms with running time $\mathrm{poly}(n)$.

As mentioned in the introduction, essentially all of the major examples of unsolved explicit construction problems are dense. The density of solutions immediately implies the existence of a nontrivial randomized algorithm for the search problem:

Lemma 1. *If Π is dense, then EC_Π is in FZPP^Π; in other words there is a randomized algorithm using a Π-membership oracle which outputs a correct answer to EC_Π with probability 1, and has polynomial runtime in expectation.*

Proof. Sample a uniform string $x \sim \{0,1\}^n$. Use a Π membership oracle to test if $x \in \Pi_n$; if the test passes output x, else repeat. □

This basic upper bound leads us to the first *conditional derandomization* result for general dense explicit construction problems, based on a classical and powerful result in derandomization:

Theorem 1 ([24, 33, 37])**.** *Assume that there is a language L computable in time $2^{O(n)}$ with a Π oracle, such that L requires Π-oracle boolean circuits of size $2^{\Omega(n)}$. Then $\mathsf{FZPP}^\Pi \subseteq \mathsf{FP}^\Pi$. Consequently, under the same assumption there is an FP^Π algorithm for EC_Π whenever Π is dense.*

For this reason, if the property Π is testable in polynomial time, then EC_Π has a deterministic polynomial time algorithm under plausible assumptions; if it is testable with an NP-oracle, then EC_Π is solvable in $\mathsf{FP}^{\mathsf{NP}}$ under similar assumptions.

There is one additional result, not covered by the general hardness/randomness connection, that gives an unconditional and highly nontrivial upper bound for dense explicit construction problems recognizable in P:

Theorem 2 ([13])**.** *If $\Pi \in \mathsf{P}$, then EC_Π has a polynomial time pseudodeterministic algorithm that works infinitely often: there is a sequence $Y = (x_n \in \Pi_n)_{n\in\mathbb{N}}$ and a polynomial time randomized algorithm $\mathcal{A}$ with the following property: for infinitely many n, $\mathcal{A}(1^n)$ outputs x_n with probability $\frac{2}{3}$ over its internal randomness.*

Most of the important unsolved explicit construction problems are not recognizable in polynomial time, and hence not covered by the above result; as we shall see, the class of explicit construction problems primarily investigated in this work are only recognizable in the larger class coNP. The primary exception to this is the problem of constructing prime numbers: given 1^n, output a prime $p > 2^n$. Testing membership in the primes lies in P by [3], and the primes have density $\frac{1}{O(n)}$ in the range $[2^n, 2^{n+1}]$ by the prime number theorem; hence the above result gives a polynomial time pseudodeterministic algorithm for this important explicit construction problem.

2.2 Circuit Lower Bounds as Explicit Construction Problems

Although the ultimate goal of circuit complexity is to obtain superpolynomial lower bounds for functions in NP, it remains a fundamental and difficult open problem to accomplish this even for much larger complexity classes such as E and E^{NP}, which denote the sets of languages decidable by Turing machines running in time $2^{O(n)} = \text{poly}(2^n)$ (respectively, with an NP-oracle). This problem came to more prominence after the classical work of Nisan, Wigderson and Impagliazzo mentioned above, who showed that if E (resp. E^{NP}) contains a language requiring circuits of size $2^{\Omega(n)}$ for sufficiently large n, then BPP $\subseteq$ P (resp. BPP $\subseteq \mathsf{P}^{\mathsf{NP}}$).

An important observation is that for classes running in time $2^{O(n)}$, deciding a language L on one input of length n has essentially the same cost as deciding *all inputs of length n* and printing out the entire truth table $L \cap \{0,1\}^n$. In particular:

Observation 1. *The following are equivalent for a language $L \subseteq \{0,1\}^*$:*

1. *$L \in$ E (resp. E^{NP})*
2. *There is a polynomial time algorithm (resp. with an NP-oracle) which, given 1^{2^n}, outputs the truth table of $L \cap \{0,1\}^n$.*

As an immediate consequence we have:

Corollary 1. *The following are equivalent:*

1. *There is a language in E (resp. E^{NP}) which requires circuits of size $s(n)$ for all n.*
2. *There is a polynomial time algorithm (resp. with an NP-oracle) for the following explicit construction problem: given 1^{2^n}, output the truth table of a function $f : \{0,1\}^n \to \{0,1\}$ requiring circuits of size $\geq s(n)$.*

Hence the complexity of explicit construction problems involving the production of hard boolean functions precisely captures the truth of corresponding circuit lower bounds. In this terminology, the main theorem of [24] can be rephrased as follows:

Theorem 3 ([24])**.** *Let $\epsilon > 0$ be any fixed constant. Every language in* BPP *is polynomial time reducible to the following explicit construction problem: given 1^{2^n}, output the truth table of a function $f : \{0,1\}^n \to \{0,1\}$ with circuit complexity $\geq 2^{\epsilon n}$.*

In other words, the explicit construction problem of producing hard truth tables is at least as hard as the simulation of polynomial time randomized algorithms.

2.3 Range Avoidance: Definition and Variants

We start by recalling the formal definition of Avoid:

Definition 2 ([28])**.** *Range avoidance, denoted formally as* Avoid*, is the following search problem: given a boolean circuit $C : \{0,1\}^n \to \{0,1\}^m$ with $m > n$, output a string $y \in \{0,1\}^m$ such that $y \notin \text{range}(C)$.*

This problem was first defined in [28] and investigated more comprehensively in [29]; in both of these works it went by the name Empty, however since the work of [41] the name "range avoidance" has become more standard. In [28, 29], the name "APEPP" is used to describe the class of search problems polynomial time reducible to Avoid; again this terminology has mostly fallen out of use in subsequent work on the topic and we will not use it here.

We can make the following basic observations about the complexity of this search problem:

Lemma 2.

1. Avoid *is a total search problem – every instance has a solution.*
2. *A candidate solution $y \in \{0,1\}^m$ can be verified for correctness in* coNP.
3. Avoid *is solvable in* $\mathsf{FZPP}^{\mathsf{NP}}$.

Proof.

1. This follows from the pigeonhole principle since $m < n$ implies $|\{0,1\}^n| < |\{0,1\}^n|$.
2. To test that $y \notin \text{range}(C)$ it suffices to verify that *for all* $x \in \{0,1\}^n$, we have $C(x) \neq y$; observe that the condition $C(x) \neq y$ is checkable in polynomial time since C is a boolean circuit.

3. Sample $x \sim \{0,1\}^m$ uniformly; use an NP-oracle to test if $y \in \text{range}(C)$. Repeat until a solution is found.

□

An important observation is that the above $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm does not favor any one solution over another: on a given instance C, it outputs each element of $\{0,1\}^m \setminus \text{range}(C)$ with equal probability. In Section 5.3 we will see that there is an alternative randomized NP-oracle algorithm for Avoid which has the much stronger property that the set of possible solutions output has size exactly one.

Conditions (1) and (2) place Avoid in the class $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ originally defined in [28]; indeed these two conditions may be taken as a definition of $\mathsf{TF}\Sigma_2^{\mathsf{P}}$. The class $\mathsf{TF}\Sigma_2^{\mathsf{P}}$ is rather new and largely unexplored; for a more in-depth coverage of the problems therein see [28, 31].

The last point in the above lemma, together with the general conditional derandomization result from Section 2.1, gives us the following conditional derandomization result for Avoid:

Corollary 2. *If there is a language in* E^{NP} *which requires* NP*-oracle circuits of size* $2^{\Omega(n)}$*, then* Avoid *is solvable in* $\mathsf{FP}^{\mathsf{NP}}$.

In Section 5 we will see that this theorem can be strengthened so that the circuits in question are oracle-free, and with this condition it becomes an equivalence. For now the important takeaway is that conventional complexity-theoretic wisdom says that the true upper bound for Avoid should be $\mathsf{FP}^{\mathsf{NP}}$; in Section 6 we will see some evidence that a better upper bound, for example Avoid $\in \mathsf{FP}$, is unlikely to hold.

2.3.1 Parameters

There are two main parameters through which we may restrict the problem Avoid – as we shall see later, the Avoid problem already becomes interesting in some highly restricted settings of these parameters. If our goal is to eventually obtain better Avoid algorithms in general, it is thus natural to study these weaker instances as a starting point.

Stretch: The *stretch* of an avoid instance refers to the relation between the number of output bits m and input bits n of the Avoid instance C. By definition we require $m > n$, however the problem becomes potentially easier when $m > 2n$ or $m > n^2$. In some cases it is relevant to study the ratio $\frac{m}{n}$, and in other cases the difference $m - n$. We will not fix one of these as the formal definition of a stretch parameter; instead we will simply say that an avoid instance has "stretch $n \mapsto m$" if it is of the form $C : \{0,1\}^n \to \{0,1\}^m$.

Circuit Complexity: For a circuit class C, e.g. $C \in \{\mathsf{AC}^0, \mathsf{NC}^1, \mathsf{P/poly}\}$, we define C-Avoid to be the special case of Avoid where the instance $C : \{0,1\}^n \to \{0,1\}^m$ has each output bit given by a circuit from the class C. By default Avoid = $\mathsf{P/poly}$-Avoid.

3 Reductions to Avoid

In this section we survey some of the techniques used to reduce problems to Avoid. We start with the most basic (and arguably most important) class of reductions, involving the production of hard boolean functions.

3.1 Hard Functions

The general scenario in the subject of non-uniform complexity lower bounds is the following: we have a finite set U, for example $U = \{0,1\}^n$, and a class of nonuniform "algorithms" $\mathcal{A}$ computing functions $U \to \{0,1\}$. $\mathcal{A}$ induces a set $\mathcal{F} \subseteq \{0,1\}^U$ of those functions which are computed by one of the algorithms $A \in \mathcal{A}$. We would like to exhibit an explicit function $f : U \to \{0,1\}$ which is not in $\mathcal{F}$, and is hence "hard" in the computational model $\mathcal{A}$. A crucial property of essentially all computational models of interest is that there are only a few "easy functions:" $|\mathcal{F}| << |\{0,1\}^U| = 2^{|U|}$. Hence the pigeonhole principle tells us that there exists $f \in \{0,1\}^U \setminus \mathcal{F}$, which must be a hard function.

Peering into the proof $|\mathcal{F}| << |\{0,1\}^U|$, the argument typically goes as follows: if $f \in \mathcal{F}$ then by definition there is some efficient computational device $A \in \mathcal{A}$ which computes it. If A computes f on every input, then from the description of A we may fully recover the function f. The fact that the algorithms in $\mathcal{A}$ are "simple" directly implies that the device $A \in \mathcal{A}$ may be explicitly described using some number of bits $m << |U|$. Hence we may define the function Eval : $\{0,1\}^m \to \{0,1\}^U$, which given the description of some $A \in \mathcal{A}$, outputs the function f which A computes. Then we have $\mathcal{F} \subseteq$ range(Eval), and hence any point outside the range of Eval is a hard function. This directly yields a reduction from finding a hard function to Range Avoidance by interpreting Eval as an Avoid instance; the only details to be checked are: (1) the function Eval can be computed efficiently, and (2) the encoding is sufficiently succinct so that $m < |U|$ while keeping the "sizes" of machines under consideration in $\mathcal{A}$ as large as possible (so as to prove the best quantitative lower bounds).

Essentially the strongest models[2] satisfying the first condition are boolean

[2]Large communication complexity classes such as $\mathsf{PSPACE}^{\mathsf{cc}}$ are (conjecturally) incomparable to $\mathsf{P/poly}$ and provide other examples of computational lower bounds reducible to Avoid which do not follow directly from Theorem 4, see [29].

circuits and Turing machines, where we obtain:

Theorem 4 ([10, 29])**.**

1. *Producing* $f : \{0,1\}^n \to \{0,1\}$ *requiring boolean circuits of size* $2^n/n$ *is reducible in* $\mathrm{poly}(2^n)$ *time to* Avoid.
2. *For any fixed polynomial p, producing* $x \in \{0,1\}^n$ *which cannot be printed by any Turing machine of length* $n-2$ *running in* $p(n)$ *steps is reducible in* $\mathrm{poly}(n)$ *time to* Avoid.

Note that every boolean function has circuits of size $(1 + o(1))2^n/n$, hence reductions to Avoid can achieve close to maximally hard boolean functions. Obtaining the exact bound $2^n/n$ is good example of the sometimes nontrivial work involved in choosing the right encoding of machines in our complexity class; an easy analysis in [29] yields the bound $2^n/2n$ by encoding circuits naively as DAGs; to obtain the bound $2^n/n$ in [10] requires a more careful encoding.

This reduction has some important consequences in light of the discussion in Section 2.2: since explicit constructions of hard truth tables correspond to circuit lower bounds for exponential time classes, we have:

Corollary 3 ([29])**.** *If* Avoid $\in \mathsf{FP}$ *(resp.* Avoid $\in \mathsf{FP}^{\mathsf{NP}}$*) then* E *(resp.* E^{NP}*) contains a language requiring circuits of size* $2^n/n$ *for all n.*

In Section 5 we will see that this can be strengthened to an "if and only if" in the case of $\mathsf{FP}^{\mathsf{NP}}, \mathsf{E}^{\mathsf{NP}}$. Combining this with the hardness/randomness connection of [24, 37] mentioned in Section 2.1, we can also observe:

Corollary 4 ([29])**.** *Every language in* BPP *is polynomial time reducible to* Avoid.

This corollary can be proven in a more direct fashion without appealing to the powerful results of [24, 37], by instead giving a direct reduction from the problem of constructing *pseudorandom generators* to Avoid. This argument is carried out in [29].

Fine-Grained Considerations: Recall that the two main parameters of interest for an Avoid instance $C : \{0,1\}^n \to \{0,1\}^m$ are the stretch $n \mapsto m$ and the circuit complexity of C. When reducing hard function construction to Avoid, there is a natural tradeoff between the quantitative strength of the lower bound and stretch of Avoid instance; when we consider "smaller" algorithms in a class $\mathcal{A}$ we typically get shorter descriptions. For example if we wish to produce $f : \{0,1\}^n \to \{0,1\}$ hard for circuits of size s, we can reduce to an Avoid instance with stretch $\mathrm{poly}(s(n)) \mapsto 2^n$. When it comes to the circuit complexity of the Avoid instances produced by

the reductions, the relevant question is the complexity of computing Eval, each of whose output bits is essentially an instance of the "Circuit Value Problem" for the class of algorithms $\mathcal{A}$. Roughly speaking, if $\mathcal{A}$ corresponds to a "typical" class of circuits such as $\mathsf{AC}^0, \mathsf{NC}^1$, etc., the circuit value problem for $\mathcal{A}$ can be computed by circuits in the same class $\mathcal{A}$; sometimes this is referred to as a circuit class having the "universal property," see [41] for a further discussion. As a consequence we have:

Theorem 5 ([41])**.** *Let C be a circuit class with the universal property, for example $C \in \{\mathsf{AC}^0, \mathsf{ACC}^0, \mathsf{TC}^0, \mathsf{NC}^1\}$. Then producing $f : \{0,1\}^n \to \{0,1\}$ requiring C-circuits of size $s(n)$ is polynomial time reducible to a C-*Avoid *instance with stretch* $\mathrm{poly}(s(n)) \mapsto 2^n$.

*As a corollary (recall Corollary 1), if C-*Avoid *is solvable in polynomial time (resp. with an* NP*-oracle) then* E *(resp.* E^{NP}*) requires C-circuits of size $2^{\Omega(n)}$.*

Note that, for circuit classes such as TC^0 and NC^1, it is an important open problem to show that E^{NP} does not have $\mathrm{poly}(n)$ size circuits from the class. By the above connection, this means that we would already achieve a breakthrough from, for example, a $2^{O(n)}$ time NP-oracle algorithm for TC^0-Avoid in the stretch regime $n^{\omega(1)} \mapsto 2^n$, i.e. when the number of outputs is almost exponential in the number of inputs.

3.2 Sparse String Encodings

Say we have a map $C : \{0,1\}^m \to \{0,1\}^n$, and an explicit construction problem $\Pi \subseteq \{0,1\}^n$ we want to solve which has the following property: if $x \notin \Pi$ then $\Delta(x, \mathrm{range}(C)) \leq d$, where $\Delta(x, A)$ denotes the Hamming distance of x to its closest point in a set $A \subseteq \{0,1\}^n$. In other words, while we have not quite completed a reduction from Π-construction to Avoid, we have exhibited an Avoid instance whose range has small hamming distance to all the "bad strings" failing to have property Π. This implies that Π is reducible to the following range avoidance variant: given C, output a point which has Hamming distance $> d$ from every element of range(C); this variant has been referred to as Remote Point in the literature [9].

In the case that m is sufficiently smaller than n, remote point can be reduced back to the standard variant of range avoidance. For this it suffices to show that sparse boolean strings can be given compressed representations decodable in polynomial time, which is standard:

Lemma 3. *For any $k \leq n$, there exists a polynomial time computable map $S : \{0,1\}^{\lceil \log \binom{n}{k} \rceil} \to \{0,1\}^n$ such that for every string $x \in \{0,1\}^n$ of hamming weight exactly k, $y \in \mathrm{range}(S)$.*

A simple proof of this result can be found in [19]. Clearly we may extend this scheme to encode a string of hamming weight at most k while incurring an additive cost of at most $\log k$ bits to specify the weight. This gives:

Corollary 5. *If $C : \{0,1\}^m \to \{0,1\}^n$ is given, and $\log \binom{n}{k} + \log k + m < n$, then finding a string of hamming distance $> k$ to* range(C) *is reducible in polynomial time to* Avoid.

This construction occurs repeatedly in reduction to range avoidance; many important explicit construction problems can be reduced most naturally to remote point, because their definition involves being far in hamming distance from a set of simple objects. Some examples where this has been applied include:

1. A boolean function $f : \{0,1\}^n \to \{0,1\}$ is hard on average for small circuits provided it is far in hamming distance from the set of low complexity functions (used in [29]).

2. A matrix is rigid provided it is far in hamming distance from the low-rank matrices (used in [29]).

3. A matrix generates a good linear error correcting code if each row is far in hamming distance from the span of the others (used in [21]).

4. A list of strings $(x^j \in \{0,1\}^n)_{j \in J}$ is a good pseudorandom generator for size-s circuits if the string $(x_i^j)_{j \in J} \in \{0,1\}^J$ is far in hamming distance from $(D(x_1^j, \ldots, x_{i-1}^j))_{j \in J}$ for all next bit predictors $D : \{0,1\}^{i-1} \to \{0,1\}$ of circuit size $O(s)$ and all $i < n$ (used in [29]).

In each case, combining these facts with an application of Corollary 5 yields the reduction to Avoid in a rather straightforward way.

In [21], it is shown that two of the above problems (construction of rigid matrices and strong error correcting codes) can be reduced to Avoid for restricted circuit classes, in particular NC^1-Avoid. It turns out that the main difficulty is in obtaining a sparse-string encoding lemma along the lines of Lemma 3, where moreover the decoder has restricted circuit complexity. The authors of [21] accomplish this using a construction from static data structure complexity:

Lemma 4 ([21, 38])**.** *For each $k \leq n$ there is a map $S_k^n : \{0,1\}^m \to \{0,1\}^n$ with the following properties:*

1. $m \leq \log \binom{n}{k} + O(\frac{n}{\log^2 n})$

2. *Every string $x \in \{0,1\}^n$ of hamming weight at most k lies in the range of S_k^n.*

3. *Each output bit of S^n_k is computable by a $O(\log n)$-depth decision tree over the inputs, and moreover there is a* $\mathrm{poly}(n)$*-time algorithm which outputs this list of decision trees.*

In particular, the last bullet means that the map S^n_k is computable in $\mathsf{AC}^0 \subseteq \mathsf{NC}^1$. Finally we mention an alternative approach used in [9] to reduce remote point to range avoidance in restricted circuit classes based on error correcting codes. Roughly, given a range avoidance instance C, we replace it with $\mathrm{Dec} \circ C$ where Dec, Enc are the decoder/encoder for a suitable error correcting code. If x is outside the range of $\mathrm{Dec} \circ C$ then $\mathrm{Enc}(x)$ will be far in hamming distance from $\mathrm{range}(C)$. This construction enjoys a superior stretch parameter at the cost of higher circuit complexity (note that Lemma 4 cannot obtain $n \mapsto n^2$ stretch in any sparsity regime because of the additive $n/\log n$ penalty).

3.3 Strongly-Explicit Extractors and Related Constructions

Here we describe a technique which allows us to reduce to Avoid the construction of circuits computing functions $f : \{0,1\}^n \to \{0,1\}^n$ with certain strong pseudorandom properties. The reductions here run in time $\mathrm{poly}(n)$, and produce a circuit for f of size $\mathrm{poly}(n)$; this is in contrast to the reductions in Section 3.1 which produce n-bit boolean functions in time $2^{O(n)}$ given as truth tables.

The reductions here hinge on a simple high-level construction involving k-wise independent generators. To explain the method in sufficient generality we require the following definition:

Definition 3. *Let $\mathcal{A} \subseteq \binom{\{0,1\}^n}{k}$, $\mathcal{B} \subseteq (\{0,1\}^n)^k$ be given. We say that $f : \{0,1\}^n \to \{0,1\}^n$ is a (A, B) extractor if whenever $(x_1, \ldots, x_k) \in \mathcal{A}$, $(f(x_1), \ldots, f(x_k)) \notin \mathcal{B}$.*

The idea behind the definition is as follows. We think of $\mathcal{A} \subseteq \{S \subseteq \{0,1\}^n, |S| = k\}$ as defining a class of *structured* subsets of k inputs, with $|\mathcal{A}| << \binom{2^n}{k}$. For any fixed set of k inputs $(x_1, \ldots, x_k)$ and a function $f : \{0,1\}^n \to \{0,1\}^n$, we may think of $(f(x_1), \ldots, f(x_k)) \in \mathcal{B}$ as some unlikely event, i.e. $|\mathcal{B}| << 2^{nk}$. We then have that f is an (A, B) extractor iff $f(S)$ avoids the event $f(S) \in \mathcal{B}$ whenever $S \in \mathcal{A}$. If $\Pr[(y_1, \ldots, y_k) \in \mathcal{B}]$ is very small for a random sequence $y_1, \ldots, y_k \sim \{0,1\}^n$, then for any fixed $S = \{x_1, \ldots, x_k\} \in \mathcal{A}$ of size k, if we choose f uniformly at random we are very likely to have $f(S) = \{f(x_1), \ldots, f(x_k)\} \notin \mathcal{B}$. If $|\mathcal{A}|$ is also sufficiently small then we can union bound over $S \in \mathcal{A}$ and argue that a random f will be an $(\mathcal{A}, \mathcal{B})$ extractor with high probability.

This existence argument can be improved in the following sense: say that instead of choosing the function $f \sim \{0,1\}^n \to \{0,1\}^n$ uniformly at random, we sample it from a k-wise independent distribution $\mathcal{F}$: for every fixed distinct $x_1, \ldots, x_k \in \{0,1\}^n$, $(f(x_1), \ldots, f(x_k))$ is distributed uniformly on $(\{0,1\}^n)^k$ when

$f \sim \mathcal{F}$. Then in this case we have $f(S) \notin \mathcal{B}$ with the same high probability over $f \sim \mathcal{F}$, and the rest of the argument goes through as before. Thus we can argue for the existence of an $(\mathcal{A}, \mathcal{B})$ extractor inside of any k-wise independent function family $\mathcal{F}$.

The following reduction formalizes this using the standard construction of k-wise independent families by univariate polynomials over a finite field. The reduction works provided that the "smallness" of $\mathcal{A}, \mathcal{B}$ can also be exhibitted by reductions to Range Avoidance, i.e. we can cover these sets by the range of some explicit length-expanding functions $A : \{0,1\}^\ell \to \binom{\{0,1\}^n}{k}$, $B : \{0,1\}^r \to \{0,1\}^{nk}$. The technique here is quite similar to an old result of Razborov [40].

Lemma 5. *Let $A : \{0,1\}^\ell \to \binom{\{0,1\}^n}{k}$, $B : \{0,1\}^r \to \{0,1\}^{nk}$ be given as circuits. Provided $\ell + r < nk$, the following problem is reducible in polynomial time to range avoidance: output the description of a polynomial size circuit $f : \{0,1\}^n \to \{0,1\}^n$ which is an* $(\mathrm{range}(A), \mathrm{range}(B))$ *extractor. Moreover:*

1. *The circuit f produced by the reduction lies in $\mathsf{AC}^0[2]$.*

2. *If $A, B \in \mathsf{NC}^2$ the Range Avoidance instance produced by the reduction also lies in NC^2.*

Proof. We construct an instance of range avoidance $C : \{0,1\}^{\ell+r} \to \{0,1\}^{nk}$. Elements of $\{0,1\}^{nk}$ are interpreted as vectors $\bar{\alpha} = (\alpha_1, \ldots, \alpha_k) \in \mathbb{F}_{2^n}^k$, and to each $\bar{\alpha}$ we associate the function $f_{\bar{\alpha}} : \{0,1\}^n \to \{0,1\}^n$ given by:

$$f_{\bar{\alpha}}(x) = \sum_{i=1}^{k} \alpha_i x^{i-1}$$

where $\{0,1\}^n$ and $\mathbb{F}_{2^n}$ are associated in some standard way. We will construct C so that if $f_{\bar{\alpha}}$ fails to be a $(\mathrm{range}(A), \mathrm{range}(B))$ extractor, then $\bar{\alpha} \in \mathrm{range}(C)$ which will give the theorem. The fact that $f_{\bar{\alpha}}$ lies in $\mathsf{AC}^0[2]$ for any choice of $\bar{\alpha}$ follows from results of [22] on the complexity of arithmetic over $\mathbb{F}_{2^n}$.

C is constructed as follows. We interpret the input as encoding (a, b) with $a \in \{0,1\}^\ell$ and $b \in \{0,1\}^r$. Let $(x_1, \ldots, x_k) = A(a)$ and let $(v_1, \ldots, v_k) = B(b)$. We then compute the unique degree $k-1$ polynomial $p \in \mathbb{F}_{2^n}[x]$ such that $p(x_i) = v_i$ for all i; this can be accomplished in polynomial time using Gaussian elimination on the corresponding Vandermonde matrix (by assumption $A(a)$ outputs a list of k distinct strings). Finally C outputs the coefficients $(\alpha_1, \ldots, \alpha_k)$ of p. Clearly we have $\bar{\alpha} \in \mathrm{range}(C)$ whenever $f_{\bar{\alpha}}$ fails to be a $(\mathrm{range}(A), \mathrm{range}(B))$ extractor. The complexity of C is upper bounded by the complexity of computing A, B, together with that of Gaussian elimination over $\mathbb{F}_{2^n}$ which lies in NC^2. □

The above lemma can be used to reduce the construction of various "structured seed extractors" with near-optimal parameters to Range Avoidance. An important case of a structured seed extractor is a 2-source extractor: in this case the domain of the function f is $\{0,1\}^n \times \{0,1\}^n$, and the family of simple sets $\mathcal{A}$ is the set of "combinatorial rectangles:" sets of the form $L \times R$ for some $L, R \subseteq \{0,1\}^n$, $|L| = |R| = k$. Such a combinatorial rectangle has size k^2 but can be encoded using $2kn$ bits, which yields the appropriate circuit A for the reduction. The "bad event" $\mathcal{B}$ is that the first bit of the function f is biased towards 1 or 0 over the rectangle; here we can use the sparse string encodings from Section 3.2 to form the circuit B. Details of this reduction can be found in [29]. Recent advances have yielded 2-source extractors with parameters very close to optimal, but still not meeting the bounds attainable here [8, 34]. Other kinds of structured-seed extractors studied in the literature include extractors for varieties and for affine spaces, see [7, 14].

4 Reducibilities Between Avoid Variants

In this section we cover some known reductions between variants of Avoid which are restricted in one of the two main parameters: stretch and circuit complexity. Most of the complexity classes C for which we typically study C-Avoid are restricted in terms of *circuit depth*; when we refer to the depth of an Avoid instance $C : \{0,1\}^n \to \{0,1\}^m$, we mean the depth of C as a boolean circuit.

4.1 Increasing the Stretch, and its Effect on Circuit Depth

The first natural question to ask is whether Avoid is robust to changes in the stretch parameter: can we reduce a general Avoid instance in polynomial time to an instance with stretch $n \mapsto n^{100}$? The following lemma is essentially the only known result along these lines, which says that such reductions are possible, provided we are afforded the use of an NP-oracle. The proof of this lemma is simple and parallels classical results in proof complexity and cryptography concerning reducibility amongst pigeonhole principles and amongst pseudorandom generators respectively [17, 39].

Before explaining the reduction, we mention an important parameter to keep track of, which we'll call the *depth complexity* of the reduction: the reduction will take a circuit $C : \{0,1\}^n \to \{0,1\}^{n_1}$ and produce a circuit $C' : \{0,1\}^n \to \{0,1\}^{n_2}$ which makes some oracle calls to C and otherwise performs some basic (NC^0) computations inbetween; we refer to the depth of the reduction as the depth of the longest chain of oracle calls C' makes to C. The relevance is that if C is given as a circuit of depth d_0, and we construct C' by a reduction with depth complexity d, then C' will be computed by an (explicitly given) circuit of depth $d_0 \cdot d$. By keeping

track of this depth complexity we can analyze how our stretching reductions affect the depth of the circuits defining our Avoid instances. To state the quantitative condition in the tightest way we need the following function:

Definition 4. *For $n' > n$ and $d \geq 1$, define $\Delta_d(n' \mid n)$ by induction on d as follows:*

1. $\Delta_1(n' \mid n) = n'$
2. $\Delta_{d+1}(n' \mid n) = \lfloor \frac{\Delta_d(n'|n)}{n} \rfloor \cdot (n' - n) + \Delta_d(n' \mid n)$

The function $\Delta_d(n' \mid n)$ is a kind of integral approximation to $(n'/n)^d n$. An operational interpretation is as follows: we start with n items in a basket, and get to perform the following update for d steps: choose any grouping of the current items into bundles, and replace each bundle of size exactly n by a larger bundle of size n'. The maximum number of items obtainable in this way after d steps equals $\Delta_d(n' \mid n)$.

The relevant fact used in the following lemma is that, if $f = g \circ h$, $y \notin \text{range}(f)$, then either $y \notin \text{range}(g)$, or else for any preimage $g(z) = y$ we must have $z \notin \text{range}(h)$.

Lemma 6. *Let $n_2 \geq n_1 > n$ be given. If $\Delta_d(n_1 \mid n) \geq n_2$, then* Avoid *with stretch $n \mapsto n_1$ can be reduced to stretch $n \mapsto n_2$ in* $\text{poly}(n_2)$ *time and depth complexity d with an* NP*-oracle.*

Proof. We prove the lemma by induction on d. Note that Avoid can always be reduced downward in stretch by ignoring some output bits. In the case $d = 1$, $n_2 \leq n_1$, and so by this observation there is nothing to prove. If the lemma holds up to d, then we can construct an instance $C' : \{0,1\}^n \to \{0,1\}^m$ computable with depth d calls to C, so that $m \leq \Delta_d(n_1 \mid n)$, and given any solution to C' Avoid we can find one for C in polynomial time with an NP-oracle. We now write m as $m = k \cdot n + \ell$ for maximal k subject to $\ell \geq 0$; we can then define C'' which, given $x \in \{0,1\}^n$, first applies C' to get an m bit string z. We can write z as $z_1, \ldots, z_k, z_{k+1}$ where $|z_j| = n$ for $j \leq k$, $|z_{k+1}| = \ell$. Now C'' outputs $C(z_1), \ldots, C(z_k), z_{k+1}$ which has length $n_1 \cdot k + \ell \leq \Delta_{d+1}(n_1 \mid n)$. Cleary the depth complexity of C'' is $d + 1$. Given $y_1, \ldots, y_k, y_{k+1}$ outside the range of C'', we can use an NP-oracle to search for a preimage of each of y_j, $j \leq k$ under C; if one has no preimage we solve Avoid for C, otherwise we find $z_1, \ldots, z_k, z_{k+1}$ which must lie outside the range of C'; by induction we have an NP-oracle algorithm which then maps this to an Avoid solution for C. □

Some important parameter regimes are highlighted below:

Reducing stretch $n \mapsto n_1$ to $n \mapsto n_2$		
n_1	n_2	Depth Cost
$n+1$	$2n$	n
$2n$	n^2	$\log n$
$1.01n$	$100n$	$O(1)$
$n^{1.01}$	n^{100}	$O(1)$

As mentioned previously, the methods here are almost identical to those used for a related problem: given a candidate cryptographic pseudorandom generator (PRG) $G : \{0,1\}^n \to \{0,1\}^m$, how and when can we construct a generator G' with better stretch parameters whose security can be based on that of G? On the positive side we can achieve the same stretching reductions with identical depth complexities for PRGs as in Lemma 6 for Avoid. Despite these strong similarities, no results are currently known which connect these problems in some more formal framework.

4.2 Reducing the Circuit Complexity

With respect to the circuit complexity of the range avoidance instance, there is one very powerful result known which allows us to reduce C-Avoid to C'-Avoid for C' much smaller than C. The result is due to [41] and utilizes the method of randomized encodings from low-depth cryptography [1]. To state it we need the following definition:

Definition 5. *We say that a circuit $C : \{0,1\}^n \to \{0,1\}^m$ is in NC^0_k if each output of C depends on at most k inputs of C.*

We then have:

Theorem 6 ([41])**.** NC^1-Avoid *is polynomial time reducible to* NC^0_4-Avoid.

Since we have the circuit class inclusions $\mathsf{NC}^0_4 \subseteq \mathsf{AC}^0 \subseteq \mathsf{ACC}^0 \subseteq \mathsf{TC}^0 \subseteq \mathsf{NC}^1$, this result implies that for all circuit classes $C, C' \in \{\mathsf{NC}^0_4, \mathsf{AC}^0, \mathsf{ACC}^0, \mathsf{TC}^0, \mathsf{NC}^1\}$, the problems C-Avoid and C'-Avoid are polynomial time equivalent.

An important caveat is that this reduction completely destroys the stretch of the Avoid instance we start with: if the original instance of NC^1-Avoid has stretch $n \mapsto n^{100}$, the NC^0_4-Avoid instance we obtain from the reduction in Theorem 6 will have stretch roughly $m + n \mapsto m + n^{100}$, where m is the size of the formula defining the original NC^1-Avoid instance. In particular if m is a polynomial larger than n^{100}, this will be in the stretch regime $n \mapsto n + n^{1-\Omega(1)}$ (up to a reparameterization $n := m + n$) which cannot be boosted to a regime $n \mapsto (1 + \Omega(1))n$ without incurring an $n^{\Omega(1)}$ blowup in depth if we rely on Lemma 6. This distinction becomes crucial in

light of some later results we will see in Section 5, which show that for NC^0_k-Avoid and even ACC^0-Avoid, there are unconditional FP and $\mathsf{FP}^{\mathsf{NP}}$ algorithms when the stretch is sufficiently large.

5 Upper Bounds for Range Avoidance

In this section we explore some unconditional upper-bounds on the Range Avoidance problem which significantly improve those mentioned in Section 2.3. We have already seen that suitably strong derandomization assumptions imply that Avoid lies in $\mathsf{FP}^{\mathsf{NP}}$. However an unconditional proof of this would immediately imply $\mathsf{BPP} \subseteq \mathsf{P}^{\mathsf{NP}}$ and more generally that E^{NP} requires $2^{\Omega(n)}$ size circuits almost everywhere, and thus seems currently out of reach.

The results covered here will allow us to obtain the derandomization Avoid $\in \mathsf{FP}^{\mathsf{NP}}$ under weaker assumptions, as well as place it unconditionally in a complexity class smaller than $\mathsf{FZPP}^{\mathsf{NP}}$. For restricted variants of Avoid, we will discuss two other sets of results which give unconditional $\mathsf{FP}^{\mathsf{NP}}$ and FP algorithms.

5.1 The Tree Construction

The first two results we discuss in this section, in addition to the result discussed in Section 7.3, all hinge on a single construction dating back to [17, 39] and used again in [25], which also bears a strong resemblance to the stretching reduction from Section 4. The application of this construction to the reduction in Section 5.2 is essentially identical to its use in [25]. The application to the algorithm in Section 5.3 bears a much stronger resemblance to [39] (and to a lesser extent [17]), however it requires a more careful analysis.

In the following, we think of the circuit $C : \{0,1\}^n \to \{0,1\}^{2n}$ as an instance of Avoid which we aim to solve. As mentioned in Section 4 we can reduce the general case of Avoid to the case of doubling-stretch, so this is without loss of generality.

Definition 6. *Let $C : \{0,1\}^n \to \{0,1\}^{2n}$. C_0 (resp. C_1) is the function obtained by restricting C to the first (resp. last) n bits of output. For any binary string s, define $C_s : \{0,1\}^n \to \{0,1\}^n$ inductively as follows:*

1. *C_ϵ is the identity map where ϵ is the empty string.*
2. *$C_{bs}(x) = C_s(C_b(x))$ for any $b \in \{0,1\}$, $s \in \{0,1\}^*$.*

If $S \subseteq \{0,1\}^*$ is a family of strings it is perhaps most natural to think of $(C_s)_{s \in S}$ as a family of functions $\{0,1\}^n \to \{0,1\}^n$. However for our purposes it will be most natural to invert our perspective:

Definition 7. *Let C be as above. For any set of strings $S \subseteq \{0,1\}^*$, consider the following function*

$$C^S : \{0,1\}^n \to \{S \to \{0,1\}^n\}$$

given by $C^S(x) = \{s \mapsto C_s(x)\}$.

The function C^S maps elements of $\{0,1\}^n$ to *functions* $S \to \{0,1\}^n$. We will concern ourselves only with those $S \subseteq \{0,1\}^*$ which are prefix-free: no string in S is the prefix of another. In this case we may identify S with a binary tree whose leaves are S. In the important special case $S = \{0,1\}^d$, this yields a perfect binary tree of depth d.

In this terminology we may think of C^S itself a succinctly represented instance of Avoid: it takes strings of length n and outputs functions $S \to \{0,1\}^n$, which we may in turn interpret as strings of length $|S|n$. This succinct Avoid instance has two useful properties:

Lemma 7. *Let C be as above, $S \subseteq \{0,1\}^*$ prefix free. Then $C^S : \{0,1\}^n \to \{S \to \{0,1\}^n\}$ has the following properties:*

1. *Given C, x and $s \in S$ we may compute $C^S(x)(s)$ uniformly in time* $\mathrm{poly}(|C|, |s|)$.
2. *Say $s_0, s_1 \in S$ have a common parent r, i.e. $s_0 = r0$ and $s_1 = r1$. Let $S' = S \cup \{r\} \setminus \{s_0, s_1\}$. Let $f : S \to \{0,1\}^n$ be given, let $v_0 = f(s_0), v_1 = f(s_1)$, and suppose v is a preimage of (v_0, v_1) under C. Then the for the function f' given by*

$$f'(s) = \begin{cases} f'(s) = v \text{ if } s = r \\ f'(s) = f(s) \text{ otherwise} \end{cases}$$

we have that $f \notin \mathrm{range}(C^S) \to f' \notin \mathrm{range}(C^{S'})$.

Proof. For the first part, $C^S(x)(s)$ is equal to $C_s(x)$ in our original notation. To compute this value we expand $s = (b_1, \ldots, b_d)$ where $d = |s|$ and output $C_{b_d}(\cdots C_{b_1}(x)\cdots)$ which requires $|s|$ evaluations of the circuit C.

For the second part, say that $f' \in \mathrm{range}(C^{S'})$ and let $x \in \{0,1\}^n$ be its preimage. Then for all $s \in S \setminus \{s_0, s_1\}$, $C^S(x)(s) = C^{S'}(x)(s) = f'(s) = f(s)$. On the other hand for r we have $C^{S'}(x)(r) = v$, and hence $C^S(x)(s_b) = C_b(C^{S'}(v)) = v_b$ for both $b \in \{0,1\}$ by the assumption that v is the preimage of (v_0, v_1) under C. Hence overall we have $C^S(x)(s) = f(s)$ for all s, and hence $C^S(x) = f$ and so $f \in \mathrm{range}(C^S)$. □

Interpreting S as the leaves of a binary tree T, we may visualize the lemma as follows: each function $f : S \to \{0,1\}^n$ is a labeling of the leaves of T by elements of $\{0,1\}^n$. Now, C^S is a map whose domain is $\{0,1\}^n$ and whose range is the set of functions $f : S \to \{0,1\}^n$; each $f \in \mathrm{range}(C^S)$ is thus a certain "highly compressible" leaf-labeling of T, with the following properties:

1. Given the "compressed representation" of f, namely the n-bit string x such that $C^S(x) = f$, we may compute any label $f(\ell)$ of any leave $\ell \in T$ in time polynomial in the *depth* of the leave ℓ.

2. Given a tree T labeled by a "compressible" f as above, if we prune the tree T to T' by removing two leaves with a common parent, we can attempt to relabel T' by a compressible f' in the following way: take the labels v, v' assigned to these leaves, find some r such that $C(r) = (v, v')$, and label the new leaf in T' by r. If a preimage is found we succesfully generate f', otherwise we solve Avoid for C by finding a string (v, v') outside its range.

5.2 Avoid Reduces to Hard Truth Tables

The first important upper bound for Avoid is the following:

Theorem 7 ([25, 29])**.** *There is an* $\mathsf{FP}^{\mathsf{NP}}$ *reduction from* Avoid *to the following problem* ϵ-Hard*: given* 1^{2^n}*, output the truth table of a boolean function* $f : \{0,1\}^n \to \{0,1\}$ *which requires boolean circuits of size* $\geq 2^{\epsilon n}$.

This is a rather immediate consequence of Lemma 7:

Proof. Let $C : \{0,1\}^n \to \{0,1\}^{2n}$ be a given Avoid instance. Set $d = K \log |C|$ for an appropriate constant K to be determined later. Consider the tree $C^{\{0,1\}^d}$. This is a map of the form $C^{\{0,1\}^d} : \{0,1\}^n \to \{\{0,1\}^d \to \{0,1\}^n\}$ and can be interpreted an instance of Avoid with input length n and output length $2^d n = \text{poly}(|C|)$. By Lemma 7 each $g : \{0,1\}^d \to \{0,1\}^n$ in the range of $C^{\{0,1\}^d}$ may be computed by a circuit of size $O(|C|d)$. Setting K sufficiently large w.r.t. ϵ, if $g : \{0,1\}^d \to \{0,1\}$ requires boolean circuits of size $2^{\epsilon d}$ then the function $f : \{0,1\}^d \to \{0,1\}^n$ given by $f(x) = (g(x), \ldots, g(x))$ cannot lie in the range of $C^{\{0,1\}^d}$. Hence any solution to ϵ-Hard will supply us with some $f \notin \text{range}(C^{\{0,1\}^d})$.

It remains to show that we can use f to find $y \notin \text{range}(C)$. For this we use the second part of Lemma 7. Initializing $S = \{0,1\}^d$ we have a string $g \notin \text{range}(C^S)$. Choose two leaves s_0, s_1 of S with a common parent r and let $v_0 = g(s_0), v_1 = g(s_1)$. Use an NP-oracle to search for the lexicographically first $v \in \{0,1\}^n$ which is a preimage of (v_0, v_1); if none exist we have solved Avoid for C. If we find a preimage v then set:

$$S' = S \cup \{r\} \setminus \{s_0, s_1\}$$

$$g'(s) = \begin{cases} f'(s) = v \text{ if } s = r \\ f'(s) = f(s) \text{ otherwise} \end{cases}$$

By Lemma 7 we have $g' \notin \text{range}(C^{S'})$. We keep repeating this procedure, at each step generating a new set S' and some $g' \notin \text{range}(C^{S'})$, with the property that each

successive S' is a subtree of the last. If we have not found a solution by the time we reach the point $S = \{0, 1\}$ and $g = \{0 \mapsto v_0, 1 \mapsto v_1\}$ then we have $g \notin \text{range}(C^{\{0,1\}})$ which by definition means $(v_0, v_1) \notin \text{range}(C)$ and we are done. □

Recalling Corollary 1 and Theorem 4, we observe the following equivalence between circuit lower bounds for E^{NP} and $\mathsf{FP}^{\mathsf{NP}}$ Avoid algorithms:

Corollary 6 ([29])**.** *The following are equivalent:*

1. *There is a language in* E^{NP} *which requires circuits of size* $2^n/n$ *for sufficiently large n.*
2. *There is a language in* E^{NP} *which requires circuits of size* $2^{\Omega(n)}$ *for all n.*
3. Avoid $\in \mathsf{FP}^{\mathsf{NP}}$

Proof. (1)→(2) is immediate. Recall that by Corollary 1 the existence of a language in E^{NP} requiring $2^{\Omega(n)}$ sized circuits is equivalent to the existence of a constant $\epsilon > 0$ and an $\mathsf{FP}^{\mathsf{NP}}$ algorithm for the problem of constructing truth tables of hardness $2^{\epsilon n}$. Combining this with the above lemma yields (2)→(3). Finally, (3)→(1) is given by Theorem 4. □

This places the question Avoid $\in \mathsf{FP}^{\mathsf{NP}}$ in rare company as a derandomization question which is *exactly equivalent* to a computational hardness assumption; classical hardness/randomness connections, such as those yielding $\mathsf{BPP} = \mathsf{P}$ [24, 37], are only known to hold in one direction.

5.3 Pseudodeterministic Algorithms for Avoid

So far, the best unconditional upper bound we have seen for Avoid is $\mathsf{FZPP}^{\mathsf{NP}}$, the class of search problems solvable with zero error by a randomized NP-oracle algorithm running in expected polynomial time. We have also seen that under plausible assumptions, this can be derandomized to Avoid $\in \mathsf{FP}^{\mathsf{NP}}$, but proving this unconditionally would resolve several longstanding open problems in complexity theory and seems currently out of reach. In a pair of recent breakthroughs [10, 35], it was shown that Avoid can be placed unconditionally inside a complexity class lying between $\mathsf{FP}^{\mathsf{NP}}$ and $\mathsf{FZPP}^{\mathsf{NP}}$, which is called $\mathsf{psZPP}^{\mathsf{NP}}$:

Definition 8. *We say that a search problem* $\mathcal{S} \subseteq \{0, 1\}^* \times \{0, 1\}^*$ *is in* $\mathsf{psZPP}^{\mathsf{NP}}$ *if there is a choice function* $F : \{0, 1\}^* \to \{0, 1\}^*$ *assigning each instance x of* $\mathcal{S}$ *to a solution* $F(x)$ *with* $(x, F(x)) \in \mathcal{S}$*, such that F is computable in* $\mathsf{FZPP}^{\mathsf{NP}}$*.*

The prefix ps- stands for *pseudodeterministic*. Note that the computational model associated with $\mathsf{psFZPP}^{\mathsf{NP}}$ and $\mathsf{FZPP}^{\mathsf{NP}}$ are identical: both are randomized algorithms running in expected polynomial time with an NP-oracle, which are required to output a valid answer with probability 1. The distinction is that a general $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm is allowed to output different solutions to a given instance depending on its internal randomness, while a pseudodeterministic algorithm must output a fixed solution that depends only on the instance. As mentioned in Section 2.3, the obvious $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm for Avoid is not pseudodeterministic: its output is a uniformly random solution. The following was shown in [35], strengthening a slightly weaker result shown immediately prior by [10]:

Theorem 8 ([10, 35])**.** Avoid $\in \mathsf{psZPP}^{\mathsf{NP}}$

The most important corollary of this new upper bound is that it in turn implies a new *lower bound*, via the connection established in Section 2.2:

Corollary 7 ([10, 35])**.** *There is a language in* $\mathsf{ZPE}^{\mathsf{NP}}$ *which requires circuits of size* $2^n/n$ *for sufficiently large n.*

Proof assuming Theorem 8. This is the same as Corollary 1. Since Avoid has a pseudodeterministic $\mathsf{FZPP}^{\mathsf{NP}}$ algorithm, and hard-truth table construction reduces in deterministic polynomial time to Avoid (Theorem 4), there is a pseudodeterministic $2^{O(n)}$ time randomized NP-oracle algorithm which, given n, produces $f_n : \{0,1\}^n \to \{0,1\}$ requiring circuits of size $2^n/n$. Note that the pseudodeterministic guarantee ensures that for each n, there is a *unique* boolean function f_n produced by this algorithm. This implies that the language

$$L = \{x \mid f_{|x|}(x) = 1\}$$

requires $2^n/n$ size circuits, and it can be decided in $\mathsf{ZPE}^{\mathsf{NP}}$ by running the explicit construction algorithm to produce $f_{|x|}$ as a truth table and outputting $f_{|x|}(x)$. □

The main result of [10, 35] is actually a bit stronger, giving an inclusion of Avoid in a single-valued, functional variant of the class $\mathsf{S}_2^{\mathsf{P}}$ which we do not define formally here. The pseudodeterministic algorithm is then obtained by combining this with an older result of Cai [6] which shows how to simulate $\mathsf{S}_2^{\mathsf{P}}$ in $\mathsf{ZPP}^{\mathsf{NP}}$.

The proof methods in [10, 35] also rely on the "tree construction" described in the previous section. The crucial distinction between these methods and those in the previous section is that [10, 35] consider the function C^S in the case where the prefix tree for S has depth linear in n, and hence $|S|$ can have *exponential* size. In this regime the tree cannot be written down explicitly by a polynomial time algorithm, however we can still perform certain operations on the tree implicitly. The first observation needed in [10, 35] is the following[3]:

[3] In [10, 35] they set $S = \{0,1\}^{2n}$ and use a slightly different construction. The variant described here is used in [31] and originates from [39].

Observation 2. *There is an explicit function* $f_{\text{diag}} : \{0,1\}^n \to \{0,1\}^n$*, computable uniformly in* poly($|C|$) *time on every input, so that* $f_{\text{diag}} \notin \text{range}(C^{\{0,1\}^n})$.

Proof. The proof is via the standard Cantor diagonalization. For an n-bit string v let $\neg v$ denote the string obtained by flipping all its bits. For $s \in S$ set $f_{\text{diag}}(s) = \neg C_s(s)$. Then for any $x \in \{0,1\}^n$, $C^{\{0,1\}^n}(x)$ differs from f_{diag} on input x; hence f_{diag} cannot lie in range($C^{\{0,1\}^n}$). □

Hence, unlike the case of the shallower Avoid-tree in Section 5.2, we can *explicitly describe* an element outside the range of $C^{\{0,1\}^n}$. Unfortunately $|\{0,1\}^n|$ is prohibitively large and so we cannot explicitly expand the tree and perform the reduction procedure described in the proof of Theorem 7. Nonetheless we can consider the function C^S, for certain *succinctly describable* S which are subtrees of $\{0,1\}^n$, and reason about the functions $f : S \to \{0,1\}^n$ inside and outside of range(C^S). This line of analysis is applied in an ingenious way in both of [10,35] to isolate certain *special solutions* to an Avoid instance which have unique certificates of correctness. The original method in [10] used a so called *win-win* analysis over a family of Avoid instances of various input lengths, and is shown to work correctly for *some instance* in this family. Li was able to modify (and significantly simplify) this original construction so that it works unconditionally on all Avoid instances.

5.4 Algorithmic Methods

As we saw in Section 3.1, algorithms for C-Avoid imply C-circuit lower bounds for exponential time classes, however for classes C below $\mathsf{P/poly}$ the reverse implication is not known. It is then natural to ask whether, for the circuit classes C which are unconditionally known not to contain large uniform classes such as $\mathsf{EXP}, \mathsf{NEXP}, \mathsf{EXP}^{\mathsf{NP}}$, we can strengthen these lower bounds to achieve range avoidance algorithms. Two important special cases are $C = \mathsf{AC}^0$, where lower bounds for very simple functions (parity) have been known for decades [2, 16], and $C = \mathsf{ACC}^0$, where lower bounds inside NEXP were more recently obtained by Williams [45].

Williams' breakthrough result was established by general method laid out in [45] and denoted "the Algorithmic Method." The main theorem of [45] says that for any "sufficiently nice" class of circuits C, any algorithm for testing the satisfiability of C-circuits which has sufficiently nontrivial advantage over brute-force search implies that NEXP does not have polynomial size C-circuits. Williams' lower bound for ACC^0 is then obtained by designing the necessary satisfiability algorithms for ACC^0. In a series of works [9,41], Williams' algorithmic method is adapted to the context of Avoid, where results of the following form are given: if a certain computational task associated with C-circuits has a slightly nontrivial algo-

rithm, then C-Avoid (with a suitable stretch parameter) collapses into polynomial time (perhaps with an NP-oracle).

The results of [9, 41] are quite involved and beyond the scope of this survey; the main technical ingredient in both papers is the design of highly specialized PCPs. We mention only a major result of the second paper, which achieves the goal of extending the best known lower bounds techniques for ACC^0 to achieve matching ACC^0-Avoid algorithms:

Theorem 9 ([9]). ACC^0-Avoid *with stretch* $n \mapsto 2^{\log^{\omega(1)} n}$ *is solvable in* $\mathsf{FP}^{\mathsf{NP}}$. *More specifically, for each* $d, m \in \mathbb{N}$ *there exists* $c \in \mathbb{N}$ *and an* $\mathsf{FP}^{\mathsf{NP}}$ *algorithm* $\mathcal{A}$ *which solves* Avoid *on instances* $C : \{0,1\}^n \to \{0,1\}^m$ *in which each output is computed by a depth* d $\mathrm{poly}(m)$ *size circuit over the basis* $\{\vee, \wedge, \mathrm{MOD}_m\}$, *provided* $m \geq 2^{\log^c n}$.

It should be noted that this theorem is "optimal" in terms of the stretch in the sense that it recovers the best known quantitative lower bounds against ACC^0: any Avoid algorithm of the above kind immediately implies that E^{NP} requires $2^{n^{\Omega(1)}}$ size ACC^0 circuits by Theorem 5, and obtaining an $\mathsf{FP}^{\mathsf{NP}}$ algorithm in a stronger stretch regime would imply a better lower bound then is currently known.

5.5 NC^0 Avoid With Large Stretch

We mention here briefly the only other case of Avoid for which nontrivial algorithms are known:

Theorem 10 ([18, 21])**.**

1. NC^0_2-Avoid *has a polynomial time algorithm.*
2. NC^0_k-Avoid *has a polynomial time algorithm in the stretch regime* $n \mapsto n^{k-1}/\log n$.

Unlike the previous upper bounds for Avoid, these algorithms are based on direct combinatorial properties of the special Avoid instances. The first is from [21] and is a rather direct application of known polynomial time algorithms for 2-Sat. The second result was first proven in a weaker form in [21], utilizing the theory of k-wise independent distributions. This original algorithm required a stretch regime $n \mapsto n^{k-1}$ and only worked with an NP-oracle; both aspects were improved in [18], which uses an iterative algorithm to maintain a prefix $(y_1, \ldots, y_i) \in \{0,1\}^i$, so that many of its extensions to a complete string $y \in \{0,1\}^m$ lie outside the range of the Avoid instance C.

6 Lower Bounds for Avoid

In this section we cover some results which indicate the hardness of solving Avoid in certain complexity classes. Of course any unconditional lower bound for polynomial time algorithms would imply $\mathsf{P} \neq \mathsf{NP}$, hence we can only hope for results here which are conditional or which hold against severely restricted models of computation.

We have seen in Section 3 that many difficult and well-studied explicit construction problems reduce to Avoid. Certainly this qualifies as evidence of the *mathematical difficulty* of discovering improved algorithms for Avoid, however it would only qualify as evidence of *computational hardness* if we had reasons to believe that some of these problems were truly outside of polynomial time. While the situation is certainly unclear, the prevailing wisdom is that most (if not all) of the explicit construction problems discussed in Section 3 ultimately lie in P. The intuition seems to be that, once a suitable mathematical understanding of the relevant pseudorandom concept (e.g. circuit complexity or matrix rigidity) is obtained, explicit constructions will follow; the only barrier is our (humanity's) mathematical immaturity. Hence to get "evidence" of the computational hardness of Avoid, it seems we need to look for reductions to Avoid from problems of a different flavor.

Even if the intuition in the previous paragraph is unconvincing, there is a second reason why reductions from explicit construction problems fail to resolve the hardness of Avoid: since explicit construction problems have unary input, they are all solvable in $\mathsf{FP/poly}$ by hardcoding a solution of each length as advice. For this reason, the results in Section 3 seem to have no bearing whatsoever on the question of whether Avoid is solvable by polynomial-size circuits.

6.1 Cryptographic Hardness

The first evidence of strong computational hardness for Avoid, namely Avoid $\notin \mathsf{FP/poly}$, was obtained in [23] using a strong cryptographic assumption. This assumption, known as *indistinguishability obfuscation* or $\mathcal{IO}$ for short, posits the existence of a universal compiler which can take any program (circuit) C and produce an equivalent one $\mathcal{IO}(C)$ which has the same functionality as the original, but which hides all implementation details: for any pair C, C' with the same functionality, $\mathcal{IO}(C)$ and $\mathcal{IO}(C')$ should look the same to a computationally-bounded observer. This concept was originally introduced in [5] (where a more formal definition can be found) and has received considerable attention in the years since. In [23], it is shown that assuming the existence of $\mathcal{IO}$ we can obtain conditional hardness for Avoid:

Theorem 11 ([23])**.** *Assume that there exists $\mathcal{IO}$ scheme secure against subexponential time nonuniform algorithms and that* $\mathsf{NP} \not\subseteq \mathsf{coNP/poly}$*. Then* Avoid $\notin \mathsf{FP/poly}$*.*

For a considerable span of time after the original definition of $\mathcal{IO}$ in [5] there was little public consensus on whether or not secure $\mathcal{IO}$ was a plausible assumption. This changed after a recent breakthrough of Jain, Lin and Sahai [27], who demonstrated how to construct $\mathcal{IO}$ from a few concrete cryptographic hardness assumptions which have each withstood years of attacks and are considered highly plausible. As a result, the existence of $\mathcal{IO}$ is now considered by many as "likely true" and hence the results of [23] give a rather convincing argument that Avoid is not solvable by polynomial size circuits.

A second line of work by [11] obtains a similar kind of cryptographic hardness for Avoid, which moreover applies to C-Avoid for very simple classes of circuits C. In particular, assuming the hardness of certain nonstandard variants of LWE and LPN, [11] obtain hardness for TC^0-Avoid, and hence for NC^0_4-Avoid by Theorem 6. The cryptographic assumptions used in [11] are introduced in the paper itself and have an unclear relationship with the more standard variants of LWE and LPN studied widely in the literature.

6.2 Hardness in the Black Box Model

A second kind of "hardness" result for Avoid is known in a restricted computational model, called the "black box" or "decision tree" model. Here it can be shown that no $\mathsf{FP}^{\mathsf{NP}}$ algorithm can solve Avoid given that it treats the instance $C : \{0,1\}^n \to \{0,1\}^m$ as a black box and ignores its implementation as a boolean circuit. By standard techniques, this can be used to build an oracle relative to which Avoid is not in $\mathsf{FP}^{\mathsf{NP}}$ (see [43] for an explanation). This lower bound was essentially proven by Wilson [44], who gave an oracle relative to which $\mathsf{E}^{\mathsf{NP}} \in \mathsf{P/poly}$; the interpretation of this result in terms of Avoid was first noted by [43]. Since the proof is quite simple and Wilson states it in a rather different terminology, we reproduce it here in the language of range avoidance. We first need the following definition of a black-box $\mathsf{FP}^{\mathsf{NP}}$ query algorithm:

Definition 9. *Let N, M be given, $M \geq 2N$ and both powers of two. Each assignmet $\alpha \in \{0,1\}^{N \log M}$ can be interpreted uniquely as defining a function $f_\alpha : [N] \to [M]$, i.e. for $x \in [N]$ and $j \in [\log M]$ we interpret $\alpha_{x,j}$ as the j^{th} bit of $f_\alpha(x)$.*

Let Q be a query algorithm which, given an assignment α, may choose at each step a DNF D over the variables $\{\alpha_i \mid i \in N \log M\}$, and receive its value. After all queries have been made, Q must output a value $y \in [M]$. We say that Q solves the Avoid *problem if for all assignments α, $Q(\alpha)$ outputs a value $y \notin \mathrm{range}(f_\alpha)$. The "width complexity" of Q is the maximum width of a DNF it queries, and the*

"query complexity" is the maximum number of queries it makes before outputting an assignment.

If we imagine $N = \{0,1\}^n$ and $M = \{0,1\}^m$ for some $m > n$, then each $f_\alpha : [N] \to [M]$ corresponds to a potential "oracle instance" of Avoid. Now, from the perspective of an $\mathsf{FP}^{\mathsf{NP}}$ algorithm, each NP query may make a nondeterministic guess, followed by a polynomial time verification. This verification can only read $\text{poly}(n) = \text{poly}\log(N)$ bits of the oracle assignment α. The NP query outputs 1 if at least one of these verifications pass. Hence the value of this query, as a function of the oracle assignment, corresponds to a DNF of width $\text{poly}\log(N)$ over the variables of α, and since the $\mathsf{FP}^{\mathsf{NP}}$ algorithm must run in $\text{poly}(n) = \text{poly}\log(N)$ time, it can only make $\text{poly}\log(N)$ many such queries to the oracle before terminating. Hence if we can prove that any query algorithm Q of the above form must have either its query complexity or width complexity exceeding $\text{poly}\log(N)$ then this will show that no such $\mathsf{FP}^{\mathsf{NP}}$ algorithm can work relative to every oracle.

Theorem 12 ([44])**.** *If Q is as above, has query complexity q and width complexity w and solves the $[N] \to [M]$* Avoid *problem, then $qw \geq N$. Hence* Avoid $\notin \mathsf{FP}^{\mathsf{NP}}$ *relative to some oracle.*

Proof. We will design an adversary which can answer enough DNF queries so that Q is forced to output an answer, but does not give enough information about the purported assignment α so as to fix any y to be outside the range of f_α. At each step we maintain a partial assignment $\rho : [N] \to [M]$, so that after t steps we have $|\text{domain}(\rho)| \leq tw$. When a DNF $D = \bigvee_j \tau_j$ is presented, we search for a term τ_j such that there exists a total assignment consistent with both τ_j and the current partial assignment ρ. If so, then since the width of τ_j is at most w, we may extend ρ to ρ' by defining it on at most w additional inputs so that already $\tau_j(\alpha) = 1$ for all total assignments α extending ρ'; the adversary then answers "true" for the DNF query D. If it cannot find such an extension the adversary answers "false;" in this case we know that for any extension of ρ we will always falsify all terms in D. Now, if $qw < N$ then the adversary may continue this process q times and reach a termination point of Q, after which Q must output some candidate solution y. Since at this point $|\rho| \leq qw < N$, there is some $x \in [N]$ so that $\rho(x)$ is undefined; the adversary then sets $\rho(x) = y$ and completes ρ everywhere else to a total assignment α. At this point all the query responses supplied by the adversary are true of α, however $y \in \text{range}(f_\alpha)$, hence $Q(\alpha)$ has the wrong behavior. □

In [31] a much stronger oracle separation is shown for a variant of the Avoid problem, called Strong-Avoid, in which we are given a boolean circuit of the form $C : \{0,1\}^n \setminus \{0^n\} \to \{0,1\}^n$, and must find an n-bit string outside its range

(in other words we are excluding 0^n from consideration in the domain). It is shown in [31] that (relative to an oracle), Strong-Avoid is not solvable in $\mathsf{FP}_{||}^{\Sigma_2^P}$, the class of search problems solvable in polynomial time with *non-adaptive* access to a Σ_2^P oracle. This implies as a special case that the (relativizing) upper bound Avoid $\in \mathsf{psZPP}^{\mathsf{NP}}$ mentioned in Section 5.3 cannot be applied to the more general problem Strong-Avoid. The proof of this separation involves techniques developed to analyze bounded-depth proof systems and AC^0 circuits, and in particular relies on a specialized variant of the switching lemma.

7 Lossy Code

So far we have focused exclusively on the general Avoid problem, which is a total search problem lying in the second level of the polynomial hierarchy. In this section we will explore a "younger cousin" of Avoid which we refer to as the Lossy Code Problem or Lossy for short. This problem lies one level down in the polynomial hierarchy and is a total search problem in the heavily-studied class TFNP. Similar to Avoid, obtaining better algorithms for Lossy can be viewed as a derandomization problem. However, the motivation to study it is admittedly weaker than in the case of Avoid: while obtaining derandomized algorithms for Avoid would have tremendous consequences for circuit complexity and derandomization, we will see only one unconditional interesting consequence that follows from a derandomized algorithm for Lossy. On the other hand, since Lossy is an inherently easier problem than Avoid, and indeed easier even than derandomizing BPP and ZPP[4], it is possible that we can obtain positive results for this problem which are too difficult to obtain in the case of Avoid.

7.1 Definition and Basic Inclusions

We start with the formal definition of the problem:

Definition 10 (Lossy code, [30])**.** *Lossy Code, denoted* Lossy*, is the following problem: given circuits $E : \{0,1\}^n \to \{0,1\}^{n-1}$, $D : \{0,1\}^{n-1} \to \{0,1\}^n$, find an input $x \in \{0,1\}^n$ such that $D(E(x)) \neq x$.*

It is useful to think of E as an encoding function which compresses an n-bit string down to a shorter string, and D as a decoding function which attempts to recover the original string from its encoding. Here the pigeonhole principle tells us that E, D cannot define a true lossless compression scheme: there must be some x

[4]Technically speaking this only holds if we define BPP, ZPP as promise classes, but we will not concern ourselves with this distinction here.

which is not recovered form its encoding. The basic upper bounds for this problem are as follows:

Observation 3. Lossy $\in$ TFNP $\cap$ FZPP.

Proof. Lossy is an NP search problem since the validity of a solution $D(E(x)) = x$ can be checked in polynomial time. It is total by the pigeonhole principle. A random candidate solution is correct with probability $\geq \frac{1}{2}$. □

As per Lemma 1, the inclusion in FZPP tells us the following:

Corollary 8. *Under standard derandomization assumptions, namely that* E *requires* $2^{\Omega(n)}$ *size circuits,* Lossy *is solvable in deterministic polynomial time.*

Hence, unlike Avoid where "conventional wisdom" at best gives an upper bound of $\mathsf{FP}^{\mathsf{NP}}$, for Lossy it tells us the problem should completely collapse into polynomial time. We can also easily observe that in a more direct sense, Avoid is at least as hard as Lossy:

Observation 4. *Lossy Code is polynomial time reducible to Avoid.*

Proof. Let (E, D) be a Lossy Code instance. Note that D by itself can be considered as an instance of Avoid. If $x \notin \text{range}(D)$ is an Avoid solution for D, then clearly $D(E(x)) \neq x$ and so it is also a solution for the Lossy Code instance. □

Perhaps more enlightening is the following more informal connection to Avoid. Say that $C : \{0,1\}^{n-1} \to \{0,1\}^n$ is an arbitrary Avoid instance, and consider the lexicographical-pseudoinverse $C^{-1} : \{0,1\}^n \to \{0,1\}^{n-1}$ defined as follows: $C^{-1}(x)$ outputs the lexicographically first preimage of x if one exists, else it outputs 0^n. Then any solution to the Lossy "instance" (C^{-1}, C) is a solution to Avoid. Clearly (C^{-1}, C) is not actually a valid instance of Lossy, since given C we cannot efficiently construct a boolean circuit computing C^{-1} (indeed assuming $\mathsf{NP} \not\subseteq \mathsf{P/poly}$ there are some cases in which no small circuit for C^{-1} exists). However, if we have access to an NP-oracle, we can effectively use it to evaluate C^{-1}; hence an $\mathsf{FP}^{\mathsf{NP}}$ algorithm can, in some cases, treat an Avoid instance as an implicitly defined Lossy instance. In this sense we may view Lossy as a kind of "polynomial-time model" of the more general Avoid problem, where certain techniques that worked for Avoid only in the presence of an NP-oracle are now available in polynomial time for Lossy. For example:

Lemma 8. *For any constant $c \in \mathbb{N}$, a general instance of* Lossy *is polynomial time reducible to an instance $E : \{0,1\}^{n^c} \to \{0,1\}^n, D : \{0,1\}^n \to \{0,1\}^{n^c}$.*

Proof. Identical to Lemma 6, replacing the NP-oracle with E. □

Similar to the direct reduction of Lossy to Avoid, there is also an immediate reduction of Lossy to the more standard pigeonhole problems in TFNP:

Definition 11 ([26])**.** WeakPigeon *is the following search problem: given a circuit* $E : \{0,1\}^n \to \{0,1\}^{n-1}$*, find a pair* $x \neq x' \in \{0,1\}^n$ *such that* $E(x) = E(x')$*.*

WeakPigeon is the complete problem for the TFNP class PWPP, which itself is a subclass of the more famous PPP. We can then observe:

Observation 5. Lossy *is polynomial time reducible to* WeakPigeon*.*

Proof. Let (E, D) be an instance of Lossy. Interpret E as an instance of WeakPigeon and let $x \neq x'$ be a solution. Then either x or x' must be a Lossy solution since for $y = E(x) = E(x')$ it cannot be that $D(y) = x$ and $D(y) = x'$. □

7.2 Problems Reducible to Lossy Code

The primary motivation to study Avoid stems from the plethora of important problems which reduce to it. In contrast, the set of known reductions to Lossy is rather sparse. The only well-studied problem for which an *unconditional* reduction to Lossy is known is the derandomization of *Catalytic Logspace*, a complexity class defined in [4]:

Definition 12 ([4])**.** *A language L is in Catalytic Logspace, denoted* CL*, if it is computable by a Turing machine of the following form. The machine has three tapes:*

1. *Input tape: this tape has length n and is read-only.*
2. *Work tape: this tape has length* $O(\log n)$ *and is read-write.*
3. *Catalytic tape: this tape has length* $\mathrm{poly}(n)$ *and is read-write.*

At the beginning of the computation, the input x is written on the input tape, the work tape is initialized to all-zeroes, and the catalytic tape is intialized to an arbitrary value z. The machine must have the following behavior on all such initial configurations (x, z)*:*

1. *At the end of the computation, the machine successfully decides whether or not* $x \in L$*.*
2. *At the end of the computation, the catalytic tape is returned to its original state z.*

It was shown in [4] that $\mathsf{CL} \subseteq \mathsf{ZPP}$; hence under suitable derandomization assumptions $\mathsf{CL} \subseteq \mathsf{P}$, however it has remained an open problem to prove this unconditionally. The following upper bound, improving the bound $\mathsf{CL} \subseteq \mathsf{ZPP}$, is established in [12]

Theorem 13 ([12])**.** *Any language $L \in \mathsf{CL}$ is (deterministic) polynomial-time reducible to* Lossy

Two more reductions to Lossy are shown in [30]; unlike the previous they only give a reduction to a nonstandard variant of Lossy

Definition 13. *Let $A, B \subseteq \{0,1\}^*$ be languages. The problem* $\text{Lossy}^{A,B}$ *is defined as follows: given an A-oracle circuit $E : \{0,1\}^n \to \{0,1\}^{n-1}$ and a B-oracle circuit $D : \{0,1\}^{n-1} \to \{0,1\}^n$, solve the* Lossy *problem for (E, D): find a string x so that $D(E(x)) \neq x$. We abbreviate* $\text{Lossy}^A := \text{Lossy}^{A,\emptyset}$.

As hinted above, we can reduce a general Avoid instance C to $\text{Lossy}^{\mathsf{NP}}$ by letting $D = C$ and $E = C^{-1}$ (the lexicographical pseudoinverse); since we can always ignore the encoder and solve Avoid for the decoder, we see that the problems Avoid and $\text{Lossy}^{\mathsf{NP}}$ are completely equivalent. Hence any of the important explicit construction problems covered in Section 3 can be reduced directly to $\text{Lossy}^{\mathsf{NP}}$ and this reduction tells us nothing new. In [30], two examples are given where an interesting explicit construction problem is reduced to Lossy^A for a language $A \in \mathsf{NP}$ which is not known to be NP-hard, or to $\text{Lossy}^{A,B}$ where A, B lie in $\mathsf{NP} \cap \mathsf{coNP}$. Such a reduction tells us more than a generic reduction to Avoid = $\text{Lossy}^{\mathsf{NP}}$, however it is still much weaker than a full reduction to Lossy. The most interesting of these reductions is the following:

Theorem 14 ([30])**.** *Consider the problem: given 1^n, produce a prime number $p > 2^n$. This problem is reducible to* $\text{Lossy}^{\mathsf{FAC},\mathsf{FAC}}$*, where* FAC *is the integer-factorization problem. In particular if* $\mathsf{FAC} \in \mathsf{P}$ *then prime construction is reducible to* Lossy.

This theorem follows from a careful analysis of the result of [39] in Bounded Arithmetic mentioned in the introduction, who showed that the existence of infinitely many primes can be proved in $I\Delta_0$ if we add as an axiom the weak pigeonhole schema for Δ_0 formulas. The reduction in Theorem 14 produces a prime in the range $[2^n, 2^{32n}]$.

7.3 Conditional and Unconditional Upper Bounds for Lossy Code

The main positive result in [30] is a conditional derandomization for Lossy under an unusual hardness assumption for uniform deterministic algorithms. This deran-

domization result does not quite apply to the generic search problem Lossy, but rather to *uniform instance sequences* of the search problem:

Definition 14. *We say that a sequence of strings $X = (x_n)_{n\in\mathbb{N}}$ is a "uniform sequence" if there is a fixed deterministic Turing machine M_X which prints x_n in* poly(n) *time given* 1^n.

Let $\mathcal{S} \subseteq \{0,1\}^* \times \{0,1\}^*$ be a search problem. We say that a search problem $\mathcal{S}$ is polynomial time solvable on uniform instances if for every uniform sequence $X = (x_n)_{n\in\mathbb{N}}$, there is a second uniform sequence $Y = (y_n)_{n\in\mathbb{N}}$ so that for all n, $(x_n, y_n) \in \mathcal{S}$, i.e. y_n is a solution to instance x_n of the search problem $\mathcal{S}$.

The motivation for this definition is as follows: for explicit construction problems EC_Π, there is only one relevant "instance" of each length that we care about solving. The instance is the string 1^n, and the goal is to produce a string $y_n \in \Pi_n$ given this "instance." This means that solving the explicit construction problem EC_Π in polynomial time is equivalent to solving it on uniform instance sequences. Thus for the search problems Lossy and Avoid, which we primarily care about because of the explicit construction problems that reduce to them, we would already be satisfied with novel algorithms which only worked on uniform input sequences.

The main result in [30] says that, assuming certain *uniform* time-space tradeoffs for Turing machines, we can solve Lossy on all uniform sequences in polynomial time. The specific tradeoff assumption needed is the following:

Hypothesis 1. *There is a constant $\epsilon > 0$ and a language L so that the following holds:*

1. *L is decidable by a Turing machine running in time 2^n.*
2. *Any Turing machine M running in time $2^{(1+\epsilon)n}$ and space $2^{\epsilon n}$ fails to decide L on some n-bit input, for all sufficiently large n.*

We then have:

Theorem 15. *[30] Assuming Hypothesis 1,* Lossy *can be solved in deterministic polynomial time on all uniform instances.*

Recall (Corollary 8) that standard derandomization assumptions already collapse Lossy into polynomial time, with no caveats about uniform input sequences. These standard derandomization assumptions posit the existence of languages of bounded uniform complexity which are hard for boolean circuits, i.e. *nonuniform algorithms*. The important distinction between these results and the above theorem is that Theorem 15 uses a hardness assumption against *uniform* algorithms, i.e. Turing machines with no advice.

The proof of Theorem 15 utilizes the tree-construction from Section 5.1. The idea is roughly the following: say there is a uniform instance of Lossy which is hard for all polynomial time algorithms. This defines a compression scheme, computable uniformly in polynomial time, whereby any n-bit string may be compressed to $n-1$ bits and then recovered later; while the compression must fail for some strings, no polynomial time observer will be able to construct an example of such a string. Using a modification of the tree construction, this scheme can be iterated to construct a "virtual RAM," whereby a large memory of size s can be compressed to s^ϵ bits, and each bit can be read/written at cost s^ϵ; any read or write operation causing the virtual RAM to fail will indicate a failure point of the compression scheme. We can then use this to simulate any T-time computation in T^ϵ space, with an overhead of T^ϵ time to simulate each step of the original computation. If there was any uniform machine where the low-space simulation failed, a poly(T) time algorithm could use it to to find a string on which the compression procedure failed, and hence solve the uniform instance of Lossy.

Finally, we mention one additional positive result which is relevant to solving uniform instances of Lossy. This stems from the powerful result of [13] mentioned in Section 2.1, which gives a polynomial time pseudodeterministic algorithm for EC_Π whenever Π is a dense property recognizable in P. Observe that if $(x_n)_{n\in\mathbb{N}}$ is a uniform instance of Lossy, the set of y which form a solution for the instance x_n is a dense, polynomial time recognizable set. As a direct corollary we get:

Corollary 9. *For every uniform sequence of* Lossy *instances $X = (x_n)_{n\in\mathbb{N}}$, there is a solution sequence $Y = (y_n)_{n\in\mathbb{N}}$ and a polynomial time randomized algorithm which outputs y_n with high probability for infinitely many n.*

8 Open Problems

In this section we highlight some important open problems related to the topics discussed in this survey.

8.1 Better Stretching Reductions?

The first problem we highlight is to determine whether the stretching reduction in Section 4 is optimal in terms of depth:

Problem 1. *Is there an $\mathsf{FP}^{\mathsf{NP}}$ reduction from stretch $n \mapsto \alpha n$ to $m \mapsto \beta m$ with depth complexity significantly better than $\log_\alpha \beta$?*

Along the same lines we may ask:

Problem 2. *Can we reduce C-Avoid to constructing C-hard truth tables for any class C smaller than* $\mathsf{P/poly}$*?*

These two problems seem closely related: the reduction of Avoid to hard truth tables is essentially an extension of the stretching reduction, and the reason that it fails for depth-restricted circuit classes is precisely because of the depth cost inherent in the repeated composition of the avoid instance.

Perhaps the most exciting possibility is that the answer to one or both of these questions is positive, and that it can be proven unconditionally by exhibiting the reduction. Recall that the reduction from Avoid to hard truth tables allows us to prove a kind a hardness amplification result for the circuit complexity of E^{NP}: if this class requires $2^{\Omega(n)}$ size circuits then it also requires $2^n/n$ sized circuits. It can be shown using the same argument that if $\mathsf{EXP}^{\mathsf{NP}}$ requires $2^{n^{\Omega(1)}}$ sized circuits, then it also requires $2^n/n$ sized-circuits. If such an amplification result could be shown for the case $C = \mathsf{AC}^0$ using a superior stretching reduction, and we could boost the known $2^{n^{\Omega(1)}}$ AC^0 lower bounds for parity to $2^{\Omega(n)}$ lower bounds for a function in $\mathsf{EXP}^{\mathsf{NP}}$, this would imply the breakthrough $\mathsf{EXP}^{\mathsf{NP}} \not\subseteq \mathsf{NC}^1$ (this observation was made in [41]).

However it is equally natural to conjecture that the answer is negative, in which case we cannot hope to refute it unconditionally without proving $\mathsf{P} \neq \mathsf{NP}$. In this case, it would be interesting to show a negative result in a restricted *black box* model of reducibility: note that the stretching reduction in Lemma 6 treats the Avoid instance in a completely black box way, and the *depth* of the reduction can be measured in a black box sense with no reference to boolean circuit depth. The analogous question for PRG stretching reductions has been asked before (in a suitable formalization of the black box model), and remains almost completely unsolved [36].

8.2 Stronger Hardness for Avoid?

The second problem we highlight is to better understand the computational hardness of Avoid. The results of [11, 23] highlighted in Section 6 tell us that Avoid is hard under very strong cryptographic assumptions. It is still unclear whether we can base the hardness of Avoid on something more standard. In particular the following remains completely open:

Problem 3. *Is* Avoid NP*-hard under polynomial time Turing reductions? Would such* NP*-hardness contradict any standard complexity theoretic assumptions? Is there an oracle relative to which* Avoid $\in \mathsf{FP}$ *but* $\mathsf{P} \neq \mathsf{NP}$*?*

Note that if severe restrictions are placed on the reduction, it is possible to show that Avoid cannot be NP-hard unless $\mathsf{NP} = \mathsf{BPP}$ [23]. In particular if there is

a reduction from SAT making a small number of queries to AVOID instances with stretch $n \mapsto n^2$, then we may replace the oracle response with a random string, and the behavior of the reduction will still be correct with high probability. However, if a reduction adaptively queries poly(n) many instances each with stretch $n \mapsto n + 1$ and is only guaranteed to be correct when supplied with correct answers to *all* queries, replacing the oracle with random bits appears useless.

8.3 Lossy Code: More Reductions and Better Upper Bounds?

Lastly we discuss some directions for future work on the less explored problem LOSSY. The first direction is to try to demonstrate the *power* of the problem LOSSY by exhibiting more reductions from explicit construction problems to it. As discussed at the end of Section 7.3, any explicit construction problem reducible to LOSSY must also be reducible to the explicit construction problem EC_Π for a dense language $\Pi \in \mathsf{P}$. Currently the only interesting example of such a problem we are aware of is the construction of prime numbers, where a reduction to LOSSY is only known to hold under the (unlikely) assumption that integer factoring is in P. It would add significant motivation to the study of LOSSY if this assumption could be removed:

Problem 4. *Can the problem of constructing a prime $p > 2^n$ be reduced in* poly(n) *time to* LOSSY *unconditionally? More generally, are there any other interesting unsolved explicit construction problems which can be reduced unconditionally to* LOSSY*?*

A primary motivation for studying LOSSY is that it is essentially the easiest *generic derandomization problem* we are aware of, and thus we may hope to get unconditional upper bounds for it before we are able to achieve more lofty goals such as AVOID $\in \mathsf{FP}^{\mathsf{NP}}$ or $\mathsf{BPP} = \mathsf{P}$.

Problem 5. *Does Lossy admit nontrivial upper bounds similar to those given for* AVOID *in [10, 35], other than the infinitely-often pseudodeterministic algorithm on uniform instances given by [13]?*

For this problem, we note an obstacle which is that LOSSY is a TFNP search problem, and it is known that essentially all nontrivial algorithmic subclasses of TFNP collapse to P in the black box model, while LOSSY does not. This presents an obstacle for a result similar to [35] (which relativizes) to apply to LOSSY; see the introduction of [31] for a discussion of this issue. However, the pseudodeterministic construction in [13] is also non-relativizing in this same sense (see [20]). For this reason it seems possible that something could be done along the lines of [13], which uses specific properties of uniform instances of LOSSY that do not hold for general dense P explicit construction problems. For example:

Conjecture 1. *Let* $\Pi \in \{0,1\}^*$ *be an explicit construction problem reducible to Lossy Code. Then there is an* NP *language* L *so that for infinitely many* n, $L_n \subseteq \Pi_n$ *and* $|L_n| = 1$.

Of course under strong enough derandomization assumptions we expect significantly stronger assumptions to hold; the question is whether a result of this form, which is not immediately implied by [13], could be established unconditionally using specific properties of the search problem Lossy.

References

[1] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Cryptography in nc0. *SIAM Journal on Computing*, 36(4):845–888, 2006.

[2] Miklós Ajtai. $\sum^1_1$-formulae on finite structures. *Ann. Pure Appl. Log.*, 24(1):1–48, 1983.

[3] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Primes is in P. *Ann. of Math*, 2:781–793, 2002.

[4] Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff, and Florian Speelman. Computing with a full memory: catalytic space. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC 2014, pages 857–866, 2014.

[5] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2), may 2012.

[6] Jin-Yi Cai. S_2^P is subset of ZPP^{NP}. *Journal of Computer and System Sciences*, 73(1):25–35, 2007.

[7] Eshan Chattopadhyay, Jesse Goodman, and Jyun-Jie Liao. Affine extractors for almost logarithmic entropy. In *Proceedings of the IEEE 62nd Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 622–633, 2022.

[8] Eshan Chattopadhyay. Guest column: A recipe for constructing two-source extractors. *SIGACT News*, 51(2):38–57, June 2020.

[9] Yeyuan Chen, Yizhi Huang, Jiatu Li, and Hanlin Ren. Range avoidance, remote point, and hard partial truth table via satisfying-pairs algorithms. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 1058–1066, 2023.

[10] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 1990–1999, 2024.

[11] Yilei Chen and Jiatu Li. Hardness of range avoidance and remote point for restricted circuits via cryptography. Cryptology ePrint Archive, Paper 2023/1894, 2023. https://eprint.iacr.org/2023/1894.

[12] James Cook, Jiatu Li, Ian Mertz, and Edward Pyne. The structure of catalytic space: Capturing randomness and time via compression. *Preprint*, 2024.

[13] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *Proceedings of the IEEE 64th Annual Symposium on Foundations of Computer Science, FOCS 2023*, pages 1261–1270, nov 2023.

[14] Zeev Dvir. Extractors for varieties. In *2009 24th Annual IEEE Conference on Computational Complexity*, CCC 2009, pages 102–113, 2009.

[15] Paul Erdös. Some remarks on the theory of graphs. *Bulletin of the American Mathematical Society*, 53:292–294, 1947.

[16] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Math. Syst. Theory*, 17(1):13–27, 1984.

[17] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, August 1986.

[18] Karthik Gajulapalli, Alexander Golovnev, Satyajeet Nagargoje, and Sidhant Saraogi. Range Avoidance for Constant Depth Circuits: Hardness and Algorithms. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2023)*, volume 275 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 65:1–65:18, 2023.

[19] Alexander Golovnev, Rahul Ilango, R. Impagliazzo, Valentine Kabanets, A. Kolokolova, and A. Tal. AC0[p] lower bounds against MCSP via the coin problem. *Electron. Colloquium Comput. Complex.*, 26:18, 2019.

[20] Shafi Goldwasser, Russell Impagliazzo, Toniann Pitassi, and Rahul Santhanam. On the pseudo-deterministic query complexity of NP search problems. In *Proceedings of the 36th Computational Complexity Conference*, CCC 2021, 2021.

[21] Venkatesan Guruswami, Xin Lyu, and Xiuhan Wang. Range Avoidance for Low-Depth Circuits and Connections to Pseudorandomness. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2022)*, volume 245 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 20:1–20:21, 2022.

[22] Alexander Healy and Emanuele Viola. Constant-depth circuits for arithmetic in finite fields of characteristic two. In *Proceedings of the 23rd Annual Symposium on Theoretical Aspects of Computer Science, STACS 2006*, pages 672–683, 2006.

[23] Rahul Ilango, Jiatu Li, and R. Ryan Williams. Indistinguishability obfuscation, range avoidance, and bounded arithmetic. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 1076–1089, 2023.

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

[25] Emil Jeřábek. Dual weak pigeonhole principle, boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129(1):1–37, 2004.

[26] Emil Jeřábek. Integer factoring and modular square roots. *Journal of Computer and System Sciences*, 82(2):380–394, 2016.

[27] Aayush Jain, Huijia Lin, and Amit Sahai. Indistinguishability obfuscation from well-founded assumptions. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, pages 60–73, 2021.

[28] Robert Kleinberg, Oliver Korten, Daniel Mitropolsky, and Christos Papadimitriou. Total Functions in the Polynomial Hierarchy. In *12th Innovations in Theoretical Computer Science Conference (ITCS 2021)*, volume 185 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 44:1–44:18, 2021.

[29] Oliver Korten. The hardest explicit construction. In *Proceedings of the 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021*, pages 433–444. IEEE, 2021.

[30] Oliver Korten. Derandomization from time-space tradeoffs. In *Proceedings of the 37th Computational Complexity Conference*, CCC 2022, 2022.

[31] Oliver Korten and Toniann Pitassi. Strong vs. weak range avoidance and the linear ordering principle. In *Proceedings of the 65th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2024*, pages 1388–1407, 2024.

[32] Jan Krajíček. *Bounded Arithmetic, Propositional Logic and Complexity Theory*. Encyclopedia of Mathematics and its Applications. 1995.

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

[34] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 1144–1156, 2017.

[35] Zeyong Li. Symmetric exponential time requires near-maximum circuit size: Simplified, truly uniform. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024*, pages 2000–2007, 2024.

[36] Eric Miles and Emanuele Viola. On the complexity of non-adaptively increasing the stretch of pseudorandom generators. In *Theory of Cryptography*, pages 522–539, 2011.

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

[38] Mihai Patrascu. Succincter. In *Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008*, pages 305–313, 2008.

[39] Jeff B. Paris, A. J. Wilkie, and Alan R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *J. Symb. Log.*, 53:1235–1244, 1988.

[40] Alexander Razborov. Bounded-depth formula over {AND,XOR} and some combinatorial problems (russian). *Problems of Cybernetics. Complexity Theory and Applied Mathematical Logic*, pages 149–166, 1988.

[41] Hanlin Ren, Rahul Santhanam, and Zhikun Wang. On the range avoidance problem for circuits. In *Proceedings of the IEEE 63rd Annual Symposium on Foundations of Computer Science, FOCS 2022*, pages 640–650, 2022.

[42] Claude E. Shannon. The synthesis of two-terminal switching circuits. *The Bell System Technical Journal*, 28(1):59–98, 1949.

[43] Nikhil Vyas and Ryan Williams. On Oracles and Algorithmic Methods for Proving Lower Bounds. In *14th Innovations in Theoretical Computer Science Conference (ITCS 2023)*, volume 251 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 99:1–99:26, 2023.

[44] Christopher B. Wilson. Relativized circuit complexity. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science, FOCS 1983*, pages 329–334, 1983.

[45] Ryan Williams. Improving exhaustive search implies superpolynomial lower bounds. In *Proceedings of the Forty-Second ACM Symposium on Theory of Computing*, STOC 2010, pages 231–240, 2010.

The Education Column

BY

Dennis Komm and Thomas Zeume

ETH Zurich, Switzerland and Ruhr-University Bochum, Germany

dennis.komm@inf.ethz.ch and thomas.zeume@rub.de

BEATCS no 145

Logic for Systems: A Gradual Introduction to Formal Methods

Shriram Krishnamurthi
Department of Computer Science, Brown University
shriram@brown.edu

Tim Nelson
Department of Computer Science, Brown University
timothy_nelson@brown.edu

Abstract

We present a proposal for increasing the accessibility of formal methods to the large number of students who could benefit from it but may not be well-served by traditional introductions. Several principles drive our design: a focus on computer systems as the target of study; immersing in applications before theory; building up from programming knowledge; and the breaking down of big transitions into smaller pieces. Concretely, we have created tools, educational materials, evaluation platforms, and more to implement these ideas. This paper describes our philosophy, learning objectives, and learning progression and provides pointers to our materials.

1 Systems as the Motivation for Formal Methods

When the means of production grow more accessible, the need for verification grows in importance. A person growing their food in their backyard has great control over ingredients, whereas one buying it at a supermarket depends on food safety measures. Likewise, the artisan hand-crafting their program is likely to have a much better understanding of it than a person copying code from StackOverflow, GitHub, or AI. Thus, as it has become easier and easier to obtain and now generate code, the value of formal methods (FM) has correspondingly grown.

Nevertheless, we believe that the *accessibility* of FM to the average computer science student has not increased tremendously over the decades. In saying this, we recognize that many of our readers are teachers of courses or authors of books who strongly believe their work does not meet our description or proves our basic

premises wrong. We are, of course, not talking about their courses and books, but those of someone else.

On a more serious note, our goal is not to suggest that our ideas are unique (see, e.g., [5, 10, 11, 15, 19, 20]). Rather, we hope to see more of a movement of people who are rethinking approaches to teaching FM. Inasmuch as we share a vision, we would love to hear from you and to work collaboratively! The more materials we generate, the better our examples, the more territory they cover, the better for everyone. We are already growing an ecosystem of free materials.

Crucially, we center our teaching of FM around *computer systems*. Our main course is called "Logic for Systems," and covers systems topics such as role-based access control, garbage collection, electronic locks, mutual exclusion, leader election, and so on. For their final projects, students tackle systems[1] they have encountered in other courses, such as: virtual memory and page tables; process and thread scheduling; cryptographic protocols; distributed hash tables; formal grammars; blockchain, authentication, onion routing; network traffic control; algorithms for max-flow, stable matching, and heuristic search; and ownership and borrowing in Rust.

We focus on systems for several important reasons:

- Systems is a rich source of problems that benefit from the application of FM.
- Systems easily illustrate interesting and difficult parts of FM, such as state exploration.
- Using a uniform set of tools to explore a wide variety of systems topics shows the strength of those tools.
- Inasmuch as many computer science students self-identify as "not-theory people," they often gravitate towards building systems.
- Finally, whatever their leanings in computing, a good percentage of our graduates will go on to work on computer systems. Thus, focusing on systems both speaks directly to their careers and gives them ideas for how to port what they learn to their future reality. Our goal is to equip them with basic training before they go on to build tomorrow's digital infrastructure. We want all students to learn how to apply formal approaches to their own areas of interest and expertise.

It is crucial to recognize what this framing is *not*. It is not a classical logic curriculum, which has little purchase with students who don't care much about the mortality of Socrates as an example of a syllogism. It is not traditional "logic

[1] Because we want students to pursue their own interests, we also allow projects that are not traditionally "systems," provided they are sufficiently complex. These have included topics such as cellular automata, group theory, graph theory, election theory (e.g., Arrow's theorem), particle decay, chemical reactions, and music theory.

in computer science," which can often be too "on paper," front-loading the "logic" and back-loading the "computer science." Nor is it logic in the narrow service of proofs about programming languages (even though that is where the first author received his training).

One of us had a revealing moment a few years ago. A group of students (who had studied the material below) was discussing their distributed systems course project in the atrium outside his office. After overhearing some of their discussion, he peeked at the code they were writing. To his surprise (and delight), they were not programming—they were building and analyzing formal models! This is a kind of "Turing test" for a "logic for systems" FM course: when an observer cannot tell whether it is systems or FM that students are discussing.

2 Guiding Principles

Our curricular design is inspired by guiding principles:

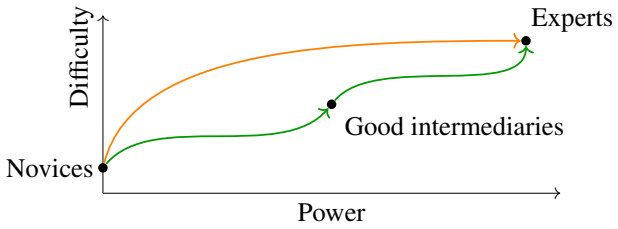
The other 90%. Many university courses are susceptible to the danger of teaching to students who are essentially "mini me"s: i.e., just like the professors. These students love the courses, gravitate to them, engage deeply, do great work, and gush about the offering afterwards, all of which are deeply rewarding to the instructor. Lost in this, however, is a large body of students who are disengaged or simply never take these courses at all.

Our view is that some small percentage—say 10%—fall in the former category, whereas our goal is to design courses to appeal to the other 90%. This is not to say that we want to exclude the 10%! But: (a) that 10% will still benefit from the course, provided it is intellectually honest, (b) they may see the material in a new light, (c) they are more likely to consult classical presentations of the material and learn it by themselves, and (d) many go to graduate school, where they will anyway see the classical version. Whereas *none* of these statements works if we design the course the other way around. Our focus, therefore, is to design FM education for the other 90%.

Play it Backwards. Many traditional "Logic in Computer Science" courses have a familiar progression: from propositional to predicate to first-order logic, and perhaps beyond; covering consistency, completeness, compactness, and other c's; and often ending with some demonstration of tools that implement these logics. These courses are excellent for the mathematically-minded—we were mini-me's who enjoyed these courses ourselves. But they fail to connect with the other 90%.

Our response is essentially to play this script "backwards." Why does logic matter in computer science? Because we can model systems and prove properties about them, and can do so at scale, using tools. We therefore *start* with tools and model systems. Students do so with a loose but sufficient understanding of the logics they are using (much as they do with the programming languages they use), which are arranged in a careful progression (Section 4). By doing this, we can quickly get to the reason these methods matter. Then, once students have accomplished enough with the tools, we introduce the formal aspects. Because, for instance, students have made extensive use of solvers with finite bounds on universes, they are now *motivated* to care about the theory: Why do those bounds exist? What happens in their absence?, and so on, turning those dry theoretical questions into ones that they can viscerally feel.

Finding "Mid"Points. It is common in FM to find a trade-off between power and difficulty: the more powerful a tool is, usually, the more difficult it is to use. Our job as FM educators is to take novices from points of low power and (relative to the instructors, though perhaps not for the novices themselves!) low difficulty to the position of experts, who have mastered the use of high-powered but difficult tools:



When confronted with these kinds of steep curves and long arcs, our instinct is to ask, "What points can we find in-between that break up this transition?" By finding meaningful "value adds" in-between, we can reveal the full complexity (and richness) of FM more gradually, and bring more students along on the ride. (Notably, our course is *optional*; students must choose to take it. This forces on us a useful discipline of accessibility.)

Combining these principles, we advocate the notion of *gradual* formal methods: a process to *meet students where they are*, with the goal of taking them to where we would like them to be (which, eventually, is where *we* are—and beyond). We are inspired by the Jackson and Wing notion of "lightweight" formal methods [8], but being "gradual" (inspired by the term "gradual typing" [17]) is more than that.

Lightweightness emphasis tractability and automation over completeness, but still sits squarely inside FM. Gradualism extends earlier and draws students into FM in a series of steps.

Although the remainder of this paper presents one possible instantiation of our principles, and the one we feel is best in our context, *we stress that these design principles are not tied to any specific tool*. A course using TLA+ [9], for instance, can also meet our criteria. Indeed, it is harmful educationally that we focus so much on languages and tools rather than on learning outcomes and progressions.

3 Learning Outcomes

After completing our curriculum, we want students to be able to:

- articulate and check correctness properties for their code;
- understand the syntax and semantics of propositional, first-order relational, and linear-temporal logics and identify where each is appropriate to use;
- use these logics to specify states and discrete-event transition systems;
- relate these abstract modeling ideas to real systems like physical locks, automated memory management, and mutual exclusion;
- design and build formal models of systems they have encountered outside the course;
- use model finders, SAT solvers, and SMT solvers to solve constraints;

and some other topics (like implementing a SAT solver and validating proof trees).

4 A Learning Progression

Learning outcomes provide a specification; the following learning progression is one implementation that we believe satisfies it. We evaluate these outcomes not only via performance on assignments, but also in live demos where students present and answer questions about their models, as well as through research and assessment projects.

4.1 From Tests to Properties

We begin by assuming no more background than programming and standard data structures and algorithms, corresponding roughly to CS1–2 courses. We do not assume that they have had a particular kind of programming (e.g., deeply-recursive functional programming), nor much depth in any particular programming language.

These assumptions are partly an artifact of the rest of our curriculum, but they are also useful in making our ideas more broadly applicable.

We assume that students recognize the value of *testing*. In our experience, this is one topic that requires little motivation: even students who refuse to acknowledge its value from classes do so quickly once they have had an industrial internship, and then pass the word along (e.g., as teaching assistants) to those younger than them. It is important that this is uncontroversial, because we build atop this foundation.

What is the correct role of testing with respect to FM? We all know the Dijkstra quote [3] that "Program testing can be used to show the presence of bugs, but never to show their absence." However, while as a method of proof it may be lacking, it is still a method of *specification*. We view tests as a form of specification: rather than conventional specifications, which are abstract (and hence define the system "from above"), tests are *point-wise* specifications that define the system "from below." That is, the weakness of tests from an FM perspective is not the tests themselves, but rather the conventional means of establishing them.

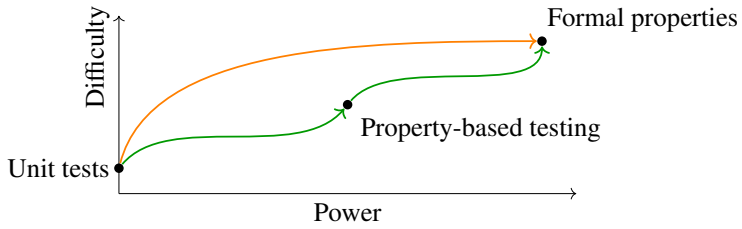
This perspective is important because, rather than being dismissive of testing, we embrace it as an aspect of FM that students are already aware of and comfortable doing. But now we set them up for failure!

Suppose one writes a test that `X must be Y` (where `X` is some program operation and `Y` is its expected output), and it fails. Why might it have failed? The most common reason is because the program is incorrect: `X` has a bug, so it does not produce `Y`. Much less frequently, `Y` is wrong.[2] But can it somehow be that neither is wrong, and yet the test fails?

An underlying assumption behind traditional unit testing is that the computation is a function: for a given input, it produces the same output. In situations where it is imperative, of course, the tester needs to do "setup" and "teardown" work to get the system into the right configuration (and then restore it). However, there is a more subtle case that lacks this input-output simplicity: when the "function" is actually a *relation*. In that case, one could have written a perfectly valid output (`Y`), but it happens to not correspond to what *this* implementation produces (*now*).

There are, in fact, many problems encountered regularly in computing that have this relational nature, which students can encounter relatively early in their education. For instance, classical graph algorithms—shortest paths, minimum spanning trees, etc.—can have multiple results, and which one is produced is clearly a function of internal implementation details. By running student tests against multiple correct implementations, each with slightly different internal strategies, we can expose the brittleness of unit tests.

[2]The reasons for this failure are important. Often, it is due to a typo or light misunderstanding. But sometimes, it can reflect a significant misunderstanding of the problem itself! Some pedagogic curricula are able to exploit this difference: [16] summarizes this line of work.



Clearly, then, a proper test needs to check for membership in a set of answers, or alternatively check for the "shape" of the answer. The former may be feasible when a given input only has a small number of outputs, but is clearly impractical in general. Instead, students learn to write recognizers of answers. In other words, they learn *property-based testing* (PBT, an old idea, but in its modern incarnation most closely associated with the QuickCheck project [2]), and through the very practical expedient of making robust unit tests, have learned to move from individual tests to *specifications*.[3]

We have written two detailed papers [14, 21] on teaching PBT, with several problem examples, as well as a detailed methodology of how to evaluate student work. An added side-effect of this work is that it sets students up to think relationally, which will become important later.

4.2 From Programs to Specifications

Students have now written specifications of the *properties* that should hold of their implemented systems, using a familiar programming language to do so. The next step is to specify the *systems* themselves formally.

Given our goals (Section 2), we used Alloy [7] because of its automation, and a syntax that mimics the object-oriented programming languages that most students have already seen. Moreover, Alloy's relational language is well-suited to modeling object relationships [6, 18] and concepts such as non-determinism, making it a great fit for modeling systems. Finally, Alloy is fundamentally grounded in model-*finding* (through satisfiability). This is crucial: in addition to supporting traditional verification against properties, it also enables property-*free* exploration of models, which addresses the oft-overlooked question of where properties come from in FM.

[3] We note an added benefit to this approach. Once students have been exposed to computational complexity—e.g., big-O—we can now discuss the complexity of a solution versus that of the recognizer. With an appropriate choice of examples—we have found subset-sum and Hamiltonian circuit especially useful—they recognize that there are problems where checking the solution is clearly in polynomial time, even though they cannot see any tractable way to solve it in less than at least exponential time. In short, this serves as an elegant and well-motivated introduction to NP-completness, leveraging their experience with PBT.

```
sig Counter {var value: one Int}
pred someTrace { Counter.value = 0 and
  always { Counter.value' = add[Counter.value, 1] } }
```

Figure 1: Modeling a counter in Temporal Forge. The syntax is similar to Alloy 6: each counter atom contains a value that varies over time. The system advances the counter by 1 with every transition.

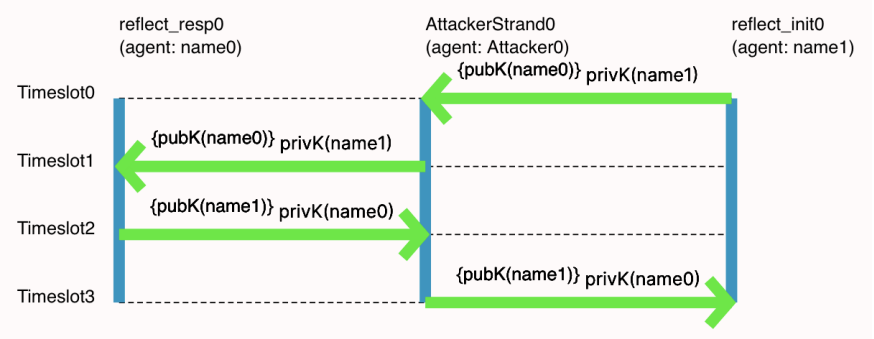


Figure 2: A domain-specific visualization of a cryptographic protocol execution.

By exploring system configurations, students can inductively identify undesirable states and turn these into formal properties to verify. This gives students a much richer and more accurate view of FM than just "verification": as all serious FM practitioners know, its value lies as much in formalization as in checking, but this is a lesson that can only be internalized from experience, not from lecture.

We now use Forge [13], a pedagogic variant of Alloy. Forge inherits all the qualities described above. In addition, it provides support for *domain-specific visualization*, *domain-specific input languages*, and a *nested series of specification languages* that gradually add power to what students can specify. Crucially, each language is self-contained, and not just a fragment of some larger language. Each has its own epistemic closure [4], coming equipped with its own (e.g.) error messages that limit themselves to concepts covered at that language level. Figures 1 and 2 show a basic example of Forge syntax and a domain-specific visualization for a more complex model, respectively.

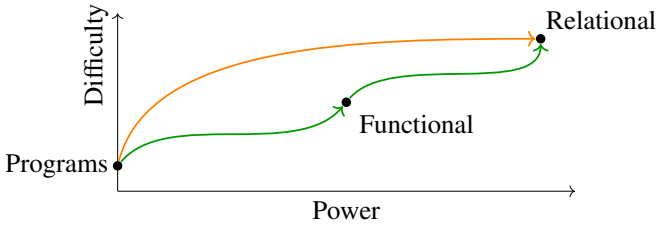
Forge is open-source and available for free at www.forge-fm.org. In addition to all the features described in this paper, the tool comes with a variety of supplemental materials and a draft textbook, making it ready for immediate use.

4.2.1 Starting with Functional Specifications

The language progression begins with *Froglet*, a subset of Alloy that allows specification with only first-order partial and total functions. For all its power, relationality comes at a cost: a larger vocabulary of operators, more subtle behavior (especially around joins), and so on increase the cognitive load of learning to model systems correctly and debug the solver output from a buggy specification. Even students who have taken a discrete math course and learned about relations in a mathematical context (which not all of our students will have done) will not necessarily be fully prepared for the visceral experience of actually using relations "for real."

Students are already used to the dotted field navigation syntax of objects and structures. Because Froglet uses the dot operator as function application (and not as general relational join, as Alloy does), students can describe constraints in a familiar way, and focus their learning on central concepts like quantification, using a pre-and-post-state pair in specification, and so on. Combining these concepts with the familiar dot-as-application notation can be surprisingly powerful. For example, if a student is modeling a leader election (perhaps as part of a protocol like Raft) they might allow a server to step down as leader if another server is more up-to-date: `some s2: Server | s2 != s and s2.currentTerm > s.currentTerm`. After some homeworks, students undertake a midterm project in which they must choose a system to model and then reason about it using techniques like invariant checking (with initiation and consecution, à la Manna and Pnueli [12]) or bounded model-checking [1].

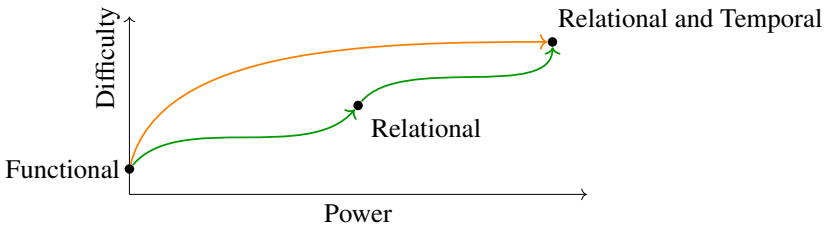
4.2.2 From Functional to Relational Specifications



Of course, not all object relationships are ideally specified as functions. Even the basic concept of a directed graph, while technically possible to model as a Boolean-valued function, can be worked with more conveniently as a relation. When students advance to *Relational Forge*, they gain access to a variety of relational operators, such as transitive closure and join. Students learn that the partial functions of Froglet are just constrained relations and (following Alloy) that the familiar dot operator is useful for far more than just field access.

For example, a student modeling basic network behavior can now represent the forwarding policy as a *relation* on the destination address, input host, and output host (`Address` × `Host` × `Host`), which is natural since a router may send a packet out multiple ports. The student can then use set comprehension and the product (`->`), transitive closure (`^`), and join (`.`) operators to build the host-reachability relation: `{s, d: Host | some a: Address | s->d in ^(a.policy)}` which would not be possible in Froglet. Students first use Relational Forge to specify and reason about automated memory management and mutual exclusion. Many choose to continue using it on their final projects to model graph algorithms, networks, the blockchain, etc.

4.2.3 From Relational to Temporal Specifications



Students have already done a great deal of temporal reasoning in both Froglet and Relational Forge. But this has largely involved safety and not liveness properties. Students could, in principle, encode liveness properties in Relational Forge, but it is subtle. Given a set of linearly-ordered timestamps, existential and universal quantification over that set are roughly analogous to the "finally" (`F`) and "globally" (`G`) modalities from Linear Temporal Logic (LTL). However, to make this work, students need to choose between encoding lasso traces (for infinite-trace semantics) or the special status of a final state (for finite-trace semantics).

For this reason, we transition to Temporal Forge roughly halfway through the course. Like Alloy 6, Temporal Forge enriches the relational language with future- and past-time LTL operators. This makes it easy to express many liveness properties without the added cognitive load of modeling traces themselves.

5 Other Formal Methods

We have presented a somewhat narrow view of FM through the lens of logic. In fact, many more "formal methods" are invaluable to the design, analysis, and implementation of computer systems, such as probabilistic methods and combinatorial optimization. We have excluded them above for the following pragmatic reasons.

First, of the lot, our own expertise lies most in logic. Second, there is enough to do in improving logic education itself, and in the space of one course, we would rather do one thing well than many things poorly. Third, there is already a rich tradition of education in probabilistic methods, which has grown rapidly with advancements in AI, so we feel we have little to contribute there.

In contrast, combinatorial optimization, with a computational focus and declarative specifications, does not appear to be as widespread (at least in the USA). It is another domain where one writes rich, concise specifications that are executed by powerful engines. Indeed, since Forge relies on solvers, there is a natural bridge from our solvers to traditional numeric ones. However, we are fortunate to work in a department with a whole class on the topic, so we simply point our students to take it next (and many do). In its absence, we would likely add some material about the broader space of solver-driven synthesis and analysis.

6 But What About Proofs?

A perhaps startling absence from our discussion above is any mention of formal deductive proof. This omission is not accidental.

There are many formal methods that have enjoyed widespread success in different spheres, from types to model-checking to even parsing. They all have in common a very high degree of automation. Even though they all come with some underlying proof, the process of arriving at that proof is hidden behind the scenes, and is even often buried in metatheory. The details are immensely valuable to the tool producers, but that is not most consumers.

It is worth asking to what extent students *should* care about proofs. Unfortunately, to make proofs tractable and fit within the time bounds of curricula, the examples they tend to see are either not from computing at all, or are so neutered as to be uninteresting. We are reminded of William Scherlis's comment, that it is better to prove "little theorems about big programs instead of big theorems about little programs," and automated methods allow us to focus on the former.

Of course, this can change! One can imagine providing large degrees of automation early on, and developing an entirely different learning progression that gradually tapers the amount of automation. Indeed, with collaborators at Brown, we have been investigating adding a proof-assistant mode that turns the bounded verification provided by Forge into a proper deductive proof. Innovations in AI-driven proof automation (e.g., using language models) can also significantly help (though arguably with very frustrating failure modes). In short, there is much room for growth—but we believe it will also take a while to approach the sophistication of domains that we can cover. (And we would be delighted to be proven wrong!)

Ultimately, we think an over-emphasis on proofs shuts the door on FM too

soon. FM should first prove itself accessible and relevant, through positive examples and not just through scare stories. Once a student has seen the value, an entire landscape of FM—including the many wonderful proof-assistants on offer—lies before them: all our work may merely be a midpoint itself. We should orient the community around gradually ushering the other 90% of students into that world through the power of persuasive examples.

Acknowledgments

We are grateful to the generations of students who have worked through iterations of this material. We appreciate our many co-authors who have built the tools and ideas that we describe here. We thank Kathi Fisler both for collaborating on this path over several years and for several comments on a draft, and Serdar Kadıoğlu and Rob Lewis for enlightening conversations. Matthias Felleisen exposited language levels as a discrete step function, which we have reformulated as a search for midpoints. Finally, we thank Dennis Komm and Thomas Zeume for editorial work.

References

[1] Armin Biere, Alessandro Cimatti, Edmund Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 1999.

[2] Koen Claessen and John Hughes. QuickCheck: A lightweight tool for random testing of Haskell programs. In *International Conference on Functional Programming (ICFP)*, 2000.

[3] E. W. Dijkstra. Structured programming. In *Software Engineering Techniques: Report of a conference sponsored by the NATO Science Committee*, 1970.

[4] Robert Bruce Findler, John Clements, Cormac Flanagan, Matthew Flatt, Shriram Krishnamurthi, Paul Steckler, and Matthias Felleisen. DrScheme: A programming environment for Scheme. *Journal of Functional Programming*, 12(2):159–182, 2002.

[5] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2nd edition, 2004.

[6] Daniel Jackson. Alloy: A lightweight object modelling notation. *ACM Trans. Softw. Eng. Methodol.*, 11(2):256–290, April 2002.

[7] Daniel Jackson. *Software Abstractions: Logic, Language, and Analysis*. MIT Press, 2nd edition, 2012.

[8] Daniel Jackson and Jeanette Wing. Lightweight formal methods. *IEEE Computer*, April 1996.

[9] Leslie Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[10] Konstantin Läufer, Gunda Mertin, and George K. Thiruvathukal. Engaging more students in formal methods education: A practical approach using temporal logic of actions. *Computer*, 57(12):118–123, December 2024.

[11] Nuno Macedo, Alcino Cunha, José Pereira, Renato Carvalho, Ricardo Silva, Ana C. R. Paiva, Miguel Sozinho Ramalho, and Daniel Castro Silva. Experiences on teaching Alloy with an automated assessment platform. *Sci. Comput. Program.*, 211:102690, 2021.

[12] Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems*. Springer, 1995.

[13] Tim Nelson, Ben Greenman, Siddhartha Prasad, Tristan Dyer, Ethan Bove, Qianfan Chen, Charles Cutting, Thomas Del Vecchio, Sidney LeVine, Julianne Rudner, Ben Ryjikov, Alexander Varga, Andrew Wagner, Luke West, and Shriram Krishnamurthi. Forge: A tool and language for teaching formal methods. In *Object-Oriented Programming Systems, Languages, and Applications*, 2024.

[14] Tim Nelson, Elijah Rivera, Sam Soucie, Thomas Del Vecchio, John Wrenn, and Shriram Krishnamurthi. Automated, targeted testing of property-based testing predicates. In *The Art, Science, and Engineering of Programming*, 2022.

[15] Doron A. Peled. *Software Reliability Methods*. Springer, 2001.

[16] Brown PLT. The Examplar project: A summary. https://blog.brownplt.org/2024/01/01/examplar.html, 2024. Accessed: 2025-02-23.

[17] Jeremy G. Siek and Walid Taha. Gradual typing for functional languages. In *Scheme and Functional Programming*, pages 81–92, September 2006.

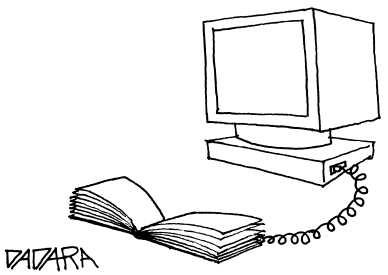
[18] J. Michael Spivey. *Z Notation - a reference manual (2. ed.)*. Prentice Hall International Series in Computer Science. Prentice Hall, 1992.

[19] Maurice ter Beek, Manfred Broy, and Brijesh Dongol. The role of formal methods in computer science education. *ACM Inroads*, 15(4):58–66, November 2024.

[20] Maurice H. ter Beek, Rod Chapman, Rance Cleaveland, Hubert Garavel, Rong Gu, Ivo ter Horst, Jeroen J. A. Keiren, Thierry Lecomte, Michael Leuschel, Kristin Yvonne Rozier, Augusto Sampaio, Cristina Seceleanu, Martyn Thomas, Tim A. C. Willemse, and Lijun Zhang. Formal methods in industry. *Form. Asp. Comput.*, 37(1), December 2024.

[21] John Wrenn, Tim Nelson, and Shriram Krishnamurthi. Using relational problems to teach property-based testing. In *The Art, Science, and Engineering of Programming*, 2021.

News and Conference Reports



CONFERENCE SPOTLIGHT: ITCS AND SODA

by Matthias Bentert (University of Bergen)

The 16th Annual Innovations in Theoretical Computer Science (ITCS) and the 36th ACM-SIAM Symposium on Discrete Algorithms (SODA) took place in January in New York City and New Orleans, respectively. I attended both and will share my experience here, highlighting some of the similarities and differences between them. I am a postdoc working on parameterized algorithms, but this spotlight is written with a broad TCS audience in mind.

ITCS (January 7-10). The 16th Annual Innovations in Theoretical Computer Science (ITCS) was held at Columbia University in New York City. This conference was much smaller, with only a single track of talks. The room where the sessions took place was quite unique, shaped like a quarter of a circle with beamers projecting onto two walls at a 90-degree angle. Figure 1 gives an illustration. A noteworthy tradition at ITCS are the *session chair rants*. These 10-minute summaries were delivered by the session chairs at the start of each session and provided an overview of all papers in the session. All session chairs did a great job, but Tal Malkin stood out with an impersonation of Ryan Williams. The talks at ITCS were quite short—only 7 minutes each—but each participant also had to record a longer video talk. These videos are publicly available and linked on the conference website.

I want to highlight a few speakers whose 7-minute talks were both educational and entertaining:

- Josh Alman: Sparsity Lower Bounds for Probabilistic Polynomials
- Georgi Li: Concentration of Submodular Functions under Negative Dependence
- Thorsten Götte: Distributed and Parallel Low-Diameter Decompositions for Arbitrary and Restricted Graphs

On a side note, New York was freezing cold at the time. I nevertheless took advantage of my trip to catch a performance of *Wicked* on Broadway. While not cheap, it was absolutely worth the price, and I highly recommend it to anyone visiting New York.

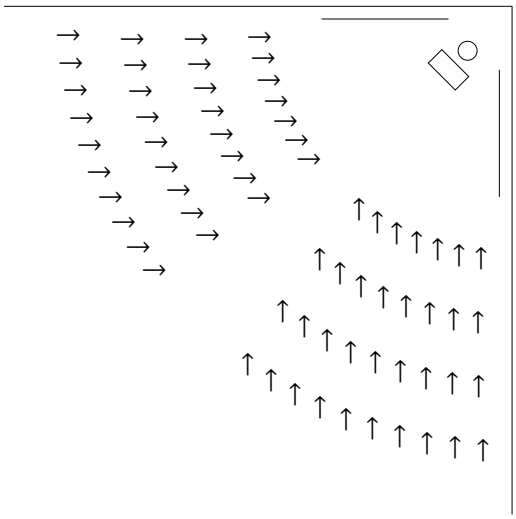


Figure 1: An illustration of the room layout at Columbia University.

SODA (January 12-14). The 36th ACM-SIAM Symposium on Discrete Algorithms (SODA) was held at the Astor Crowne Plaza at French Quarter in New Orleans. It was significantly larger than ITCS, with three parallel sessions and additional tracks for the co-located conferences ALENEX and SOSA. Given the size of the conference, I could only attend a fraction of the talks, but I still want to highlight a few that stood out to me as particularly well presented:

- Florent Becker: Strict Self-Assembly of Discrete Self-Similar Fractals in the Abstract Tile Assembly Model
- Michał Włodarczyk: Losing Treewidth In The Presence Of Weights
- Hantao Yu: Improving the Leading Constant of Matrix Multiplication

There was a joint lunch on Monday, but the event was unfortunately underwhelming and left many participants disappointed. Another curiosity was the conference's online program, which was not particularly user-friendly as it was tedious to compare talks in parallel sessions. A few participants took matters into their own hands, crawling the data from the webpage and reorganizing it. These efforts were quickly shared and greatly appreciated by the community. Another unique aspect of SODA was the business meeting, which was unlike any I had attended before. Instead of a bidding process to find a place to host the upcoming iterations of SODA, participants were asked to suggest and vote on potential places

and SIAM will try to find a suitable venue in one of the chosen cities. The result? Puerto Rico emerged as the top choice for 2027, followed by Boston and Philadelphia. It was also revealed that SODA 2026 will be held in Vancouver and this year's acceptance rate was 29%, with 192 out of 655 papers being accepted.

So far, I have mostly focused on the differences between the two conferences. To conclude, I want to highlight one similarity: both conferences celebrated gamification. At ITCS, there was a board game night where attendees played various games. I joined a few rounds of *Avion*, a social deduction game similar to Mafia or Among Us, but I also saw various other types of games being played at other tables. At SODA, Jane Street organized an *Estimathon*, where participants were tasked with estimating numbers they had no business knowing. The most memorable question for me? Estimating the smallest common multiple of three random numbers between 1 and 99 picked by the host the day prior. Both events were a fun way to unwind and connect with fellow researchers outside the formal sessions. They added a fun and engaging twist, making for an enjoyable experience beyond the academic discussions.

Report on POPL'2025

52nd ACM SIGPLAN Symposium on Principles of Programming Languages
Denver, Colorado, USA, Jan 19 — 25, 2025

Vikraman Choudhury, Università di Bologna, Bologna, Italy

POPL is one of the premier conferences in the field of programming languages. The 52nd edition of the conference, **POPL'2025**, along with its co-located events, was held in Denver, CO, at the Curtis Hotel Denver, from Jan 19 to 25, 2025. According to the General Chair's report, the conference was attended by 560+ participants. As usual, this was a huge event, with conference talks, posters, workshops, tutorials, from different areas of programming languages, as well as social events and activities.

POPL attracts researchers from different areas of computer science, connected by the common theme of programming languages. The range of topics covered in the conference was quite broad, from theoretical foundations of programming languages, to practical aspects of programming languages and software engineering. I would like to highlight a couple of talks that I attended and found particularly interesting, that would be most relevant for the theoretical computer science community.

Theory at Peek-A-Boo, Fri 24 Jan, Interaction Equivalence: Adrienne Lancelot presented a fascinating talk about the equivalence of (untyped) lambda terms. The standard notion of equivalence for lambda calculus is contextual equivalence, which means two terms are equivalent if they are observationally indistinguishable in all contexts. This is an *e*quational theory, in the sense that it is a congruence relation (equivalence relation closed under contexts), and contains β-equality. However, this is too coarse of an equivalence, since it doesn't distinguish terms by the number of steps of reduction (or cost), for example, β-equivalent terms can have different numbers of reduction steps. If equivalence is considered upto equal number of reduction steps, is there an appropriate equational theory? The paper solves this puzzle by first, decomposing the lambda calculus into a *c*heckers lambda calculus, which colors term formers black or white, and then, defining a notion of equivalence called *i*nteraction equivalence, which is an equational theory that captures cost equivalence. The key idea is that reductions can be seen as, either interactions between programs and contexts, or as internal steps within programs. The development uses various technical tools, such as Böhm separability, and intersection types, to achieve this result.

Semantic models at Peek-A-Boo, Wed 22 Jan: Abstract Operational Methods for Call-by-Push-Value: The program of Mathematical Operational Semantics

provides a rigorous foundation for giving operational semantics and contextual equivalence to programming languages, using a categorical framework of bialgebras. Recent work by the speaker and collaborators extends this framework to higher-order languages, that is, languages with binding, which is a significant improvement. The work presented in the talk extends this framework to call-by-push-value (CBPV) languages, allowing for languages with effects, and obtaining an abstract framework for proving theorems about applicative bisimulation and step-indexed logical relations.

While there were many other interesting talks and lots of exciting applications of programming languages to real-world problems, I noticed that most researchers base their work on the foundational ideas and techniques in the field, perhaps best summarized by a quote from Loris D'Antoni's keynote, "strong foundations transcend trends".

Obituary for Luca Trevisan

The Theoretical Computer Science Community is heartbroken by the untimely loss of Luca Trevisan, who passed away on June 19, 2024 in Milan at the age of 52. His husband, Junce Zhang, was at his side along with a few other dear friends and colleagues. Luca leaves behind a rich and diverse legacy of influential results and a profound impact on our community.

Life & Career Trajectory

An only child, Luca was born and raised in Rome, where as an adolescent he spent many long afternoons reading books about mathematics. He loved to cycle to Villa Ada or along the river Tiber, and was passionate about film—in particular Mel Brooks's *Young Frankenstein*, and films by Monty Python and Woody Allen. In school, Luca earned the highest marks, but he was particularly precocious in mathematics. Luca's lifelong friend, Flavio Marchetti, says "I've often spoken about Luca with our high school mathematics teacher Daniela Crosti...and we joke about Luca's questions, which frequently turned into extemporaneous original demonstrations in which he arrived at the answers by reasoning together with the class, without any advance preparation."

As an undergraduate at Sapienza University in Rome, he became the first student to graduate from the department of information science. An article about the event in the Roman newspaper *Il messaggero* from 1993 [31] sheds some light on

Luca's youthful personality. Luca's mother, Giuseppina, described him as calm, disorganized, and not very well equipped to deal with practical matters. Friends remarked that he drove like a madman and was a danger to public safety, but had a great sense of humor.

In his third year of university, he took a class on computational complexity taught by Daniel Pierre Bovet. That semester's announcement of the FGLSS breakthrough on hardness of approximation [17] caught Luca's imagination, and he began to do research on the topic with Pierluigi Crescenzi, continuing on to receive his doctorate from Sapienza in 1997. "Luca was the perfect PhD student," Pierluigi recalls. "Clearly, he was exceptionally intelligent. But he was also always so kind and thoughtful that he patiently and clearly explained his ideas, making you feel comfortable with them and never embarrassed for not immediately understanding them."

During his PhD, Luca did an internship with Madhu Sudan at IBM Research, who also hosted Luca for a postdoc at MIT, where Luca obtained his landmark result on randomness extractors [41]. Of meeting Luca, Madhu recounts "It was one of most striking first encounters. In fifteen minutes while I was driving on crowded streets of Manhattan, Luca explained elements of my own paper (with Bellare and Goldreich) to me, and explained why one idea in the paper led to a solution to what we thought was a completely different problem from a different part of the same paper. This led eventually to Luca's first FOCS(/STOC) paper with me and Sorkin and Williamson and helped launch a storied career. It was a privilege to have met Luca so early on!."

Subsequently, Luca did a second postdoc at DIMACS before joining the faculty of Columbia University. He then served on the faculty of Berkeley and Stanford before returning to Berkeley in 2014 to be the Senior Scientist at the Simons Institute for the Theory of Computing. "There is no question that Luca significantly elevated the intellectual climate in the theory group at Berkeley," remarks Simons Institute Founding Associate Director Alistair Sinclair, "and did so moreover with great humility and a disarming sense of humor. His ability to distill the essence of a huge range of mathematical ideas, and generously share his insights with others, through his blog and elsewhere, was a truly rare gift. It was a major coup for Berkeley to bring him back from Stanford as senior scientist at Simons. His advice and words of wisdom were invaluable during the endless rounds of decisions we faced in the early years of the Institute."

In 2019, Luca proudly returned to Italy to help launch the new Department of Computing Sciences at Bocconi University in Milan. His intellectual and personal charm were instrumental in attracting world-class scientists to Italy, catalyzing a period of remarkable and unexpected growth for the department. At Bocconi, he held the Invernizzi Foundation Chair in Computer Science and co-founded and directed the two-year Master of Science in Artificial Intelligence program, which

was launched in the fall of 2023 and is now undergoing significant expansion. Shortly after his return, Luca was awarded the prestigious Advanced Grant from the European Research Council, and in 2021 he became the first computer scientist to be elected to the Italian National Academy of Sciences. The Bocconi department continues to thrive on the momentum he created and remains committed to realizing the visionary legacy he shared during his final days.

Figure 1: Luca at a Bocconi University workshop, 22 May 2024

Research Contributions

In the course of his career, Luca made beautiful contributions across the theory of computing, spanning pseudorandomness, approximate optimization and PCPs, distributed computing, property testing, cryptography, and spectral algorithms. We survey just some of these below.

Approximability & PCPs. Luca's PhD research in the mid-1990s focused on approximability and included two very elegant non-approximability results: one showing that the doubly exponential dependence on the dimension of the approximation schemes for Euclidean TSP was inherent [40], for which he received the STOC 1997 Best Student Paper Award, and another that provided a systematic LP-based framework to discover gap-preserving gadget reductions and argue their optimality [49]. By the time he received his PhD, Luca had become an expert on the rapidly evolving PCP literature, crediting the thorough exposition of Bellare, Goldreich, and Sudan [11] as his entry point.

Among Luca's varied contributions to PCPs is a line of work that culminated with PCPs with the best bang for the buck for queries made, asymptotically yielding an almost factor-two gain in soundness for each additional bit queried [36, 34, 35]. The final paper in this series [35] brought tools from additive combinatorics like the Gowers norms [21, 22] into the purview of TCS. Luca had a real knack for identifying directions in pure mathematics with fruitful connections to complexity theory (and vice versa). Similarly, in [32, 50] he found deep connections between the Dense Model Theorem that plays a central role in the Green-Tao Theorem [23] on arbitrarily long arithmetic progressions in the prime numbers and fundamental concepts in average-case complexity and pseudorandomness, such as Impagliazzo's Hardcore Lemma [25].

Pseudorandomness. Luca made many important contributions to the theory of pseudorandomness, including the aforementioned Dense Model Theorem and related results. The standout of these was Luca's mind-blowing discovery [41], as a postdoc, that the Impagliazzo-Wigderson construction of pseudorandom generators from (conjectured) functions of high circuit complexity [26] is also an (unconditional) construction of randomness extractors. (See Figure 2.) Thus, Luca demonstrated that two major lines of work on randomness in computation were really studying the same problem in disguise. The connection established by Luca was very non-obvious and creative; it came as a surprise even to the leaders in the field who had been involved in both lines of work. This sudden shake-up gave many other young researchers around the world a perfect opportunity to enter what was previously a well-established and intimidating area.

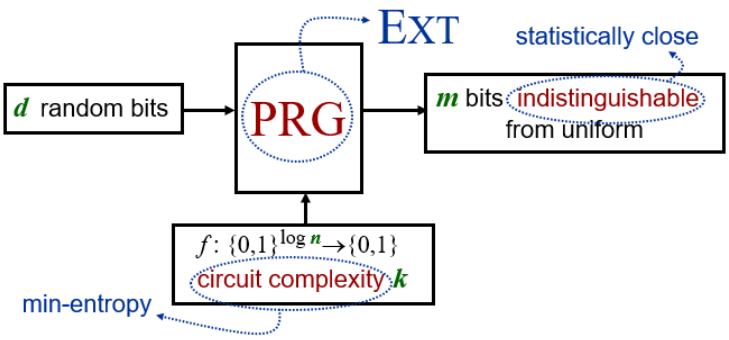


Figure 2: Luca's way of turning a pseudorandom generator (PRG) construction into a randomness extractor [41]. If instead of running the construction with a fixed function of high circuit complexity (as was always imagined), we feed it a random function drawn from a distribution of sufficient min-entropy, it turns out that output will not only be computationally indistinguishable from uniform but statistically close to uniform.

Spectral Graph Theory. In the latter half of his career, Luca's journey in approximation algorithms led him to work on algorithmic spectral graph theory. His several extensions of Cheeger's Inequality, relating the smallest eigenvalue of graph's Laplacian to nearly bipartite components [43], and relating the k'th eigenvalue to k-way graph partitioning [28] are already part of the canon of this area of research and also had striking consequences in pure mathematics [29, 30]. Reading their beautiful proofs, there is an inevitability about them that makes them hard to unsee.

Distributed Algorithms. Starting from 2012, Luca was attracted by the world of distributed computing and, in particular, by the self-organizing behaviour yielded by simple, fully-decentralized processes known as *dynamics* [7]. As part of a close-knit group of Roman collaborators (some shown in Figure 3), he proposed new techniques and models to study information-spreading and network-formation dynamics in evolving graphs that led to several significant advances in understanding the complex behaviour of such dynamical systems [9]. Luca achieved further important contributions in the study of majority and averaging dynamics and their ability to perform fundamental distributed tasks such as self-stabilizing consensus [8, 9] and community detection [6, 10].

Figure 3: Luca with friends and collaborators in Rome. From left to right: Stefano Leucci, Guido Proietti, Pierluigi Crescenzi, Luca Trevisan, Luca Becchetti, Riccardo Silvestri, and Luciano Gualà. Photo by Andrea Clementi.

Other Contributions. In cryptography, Luca had several influential papers on understanding the limitations of "black-box" constructions and reductions in constructing cryptographic primitives [19, 13, 33]. In coding theory, Luca gave the first lower bounds for locally decodable codes [27, 20], a research direction that has seen a resurgence of activity and progress two decades later. In property testing, he proved the first linear lower bound for testing a natural graph property (namely, 3-colorability) in the bounded-degree model [12].

Luca had an uncanny ability to quickly identify the significance and potential of a new direction when he saw one. He would comprehend the essence of the first paper on a topic at great depth, write a beautiful follow-up filled with insights that clarified and improved matters, and in turn inspire further follow-ups and progress. He did this repeatedly, on diverse topics, ranging from hardness amplification to Unique Games algorithms. This is best articulated in Luca's own words in his blog post "On being second" [42].

Human Impact

Beyond his research prowess, Luca was a leader in another (more subtle) way, which is no less important. As a teacher and expositor, Luca awakened curiosity in others. In his many survey articles [39] and lecture notes [38], he gave us the crispest expositions of many canonical results in theoretical computer science. His way of explaining was generous, making full use of his intellect but keeping it

in the background. He was passionate about making theoretical computer science approachable to newcomers, and his writings did so wonderfully. Many of us regularly benefited from Luca's notes for our teaching—he had a gift for distilling advanced material in an engaging style perfectly tuned and packaged for lectures; in fact, we sometimes picked topics to lecture on *because* Luca had notes on them!

On his blog, *in theory* [37], in addition to his lecture notes, Luca also shared with us what he was learning. This included topics such as zeta functions [46], manifolds [44], and other objects not so familiar to the TCS audience. It was an invitation to expand our imagination, and a reminder that it's okay to just learn something for its own sake. In person, Luca had a way of asking simple questions and then exploring them with clarity, devoid of any hint of competition or rush. Talking with him was a gentle invitation to wonder. He shared his questions and insights generously, and repeatedly went out of his way to collaborate with junior people, not even necessarily his students, including us. We doubt that our journey with Luca is unique. He represents what is best about our field: its openness and the generosity of its leaders. We hope that Luca's way will forever remain alive in our community.

As a faculty member for over 25 years, Luca was a beloved mentor to many students. He both collaborated with them on influential results and inspired them to grow as fearless researchers with their own ambitious agendas. His former students are now faculty or researchers at the University of Ottawa, NTT Research, TTI-Chicago, the University of Michigan, the Institute for Research in Fundamental Sciences, the University of Memphis, the University of Pennsylvania, and Google. "As graduate students we were awed and a bit intimidated by Luca's brilliance and prowess," recalls his first PhD student, Andrej Bogdanov. "His gentle personality and humorous manner quickly put us at ease. Luca had his share of fun solving hard problems, but what he truly enjoyed was talking, joking, or just sitting there over a cup of coffee. I signed up for an advisor. I ended up with a friend for life."

On a personal level, Luca was one of the early out, gay theoretical computer scientists who have helped make this community a welcoming one for LGBTQIA+ researchers and students. He recounted his coming-out in 2000 in a characteristically humorous (and now legendary) blog post [47], which was part of Luca's effort to "put the gay back in the Turing Centennial." In addition to his own post, he hosted a series of 8 guest blog posts [45, 15], where LGBTQIA+ colleagues from TCS and Math wrote about their experiences in the field, an initiative that had a huge impact on the inclusiveness of our field. These kinds of contributions meant so much to Luca that, even while hospitalized in his final weeks, he heroically prepared an Inspirational Talk that he was invited to give at the TCS4All workshop at STOC, which was delivered posthumously on his behalf [48]. The community's love of and respect for Luca is reflected in the numer-

Figure 4: Luca with his first three PhD students. From left to right: Kenji Obata, Luca Trevisan, Andrej Bogdanov, and Hoeteck Wee.

ous tributes that have been paid to him since his passing, including well-attended tribute workshops or sessions at RANDOM 2024 [2], the Simons Institute [3], and Bocconi University [1]; blog posts by numerous colleagues [4, 5, 14, 18, 24]; an Open Problems Column in SIGACT News [16]; and the TCC Conference naming its Young Researcher Paper Award after him.

Conclusion

Luca Trevisan was an ACM Fellow and was the first computer scientist elected to the Italian National Academy of Science. Earlier in his career, he received the Oberwolfach Prize and a Sloan Research Fellowship, and was an Invited Speaker at the International Congress of Mathematicians. He served on the Turing Award Selection Committee, the Scientific Board of the Institute for Pure and Applied Math, and the editorial boards of JACM, the ACM Transactions on Computation Theory, the SIAM Journal on Computing, and the EATCS Bulletin. He was Program Chair for RANDOM 2001, RANDOM 2005, CCC 2005, FOCS 2010, and TAMC 2013.

Luca was a brilliant scientist and expositor, and unfailingly funny friend and colleague. His clarity and insight deepened our understanding, and advanced the frontiers of research in the theory of computing. Luca leaves behind an enduring

scientific legacy. He will be sorely missed.

Andrea Clementi, Venkatesan Guruswami, Kristin Kane, Alon Rosen, Nikhil Srivastava, Salil Vadhan, and Riccardo Zecchina.

The starting point for this piece was the blog post [24].

References

[1] In memory of Luca Trevisan (1971–2024). A Workshop at Bocconi University, Milan, Italy, 2 December 2024. https://andrejb.net/lucaworkshop.html.

[2] Luca Trevisan memorial session. The 28th International Workshop on Randomization and Computation (RANDOM 2024) and the 27th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2024), 29 August 2024. London, UK. https://approxrandom2024.site/.

[3] LucaFest@Simons. Workshop at the Simons Institute for the Theory of Computing, Berkeley, CA, 8–9 October 2024. https://simons.berkeley.edu/workshops/lucafestsimons.

[4] Scott Aaronson. Luca Trevisan (1971–2024). *Shtetl-Optimized* blog post, 19 June 2024. https://scottaaronson.blog/?p=8057.

[5] Boaz Barak. Luca Trevisan (1971–2024). *Windows On Theory* blog post, 19 June 2024. https://windowsontheory.org/2024/06/19/luca-trevisan-1971-2024/.

[6] Luca Becchetti, Andrea Clementi, Pasin Manurangsi, Emanuele Natale, Francesco Pasquale, Prasad Raghavendra, and Luca Trevisan. Average whenever you meet: opportunistic protocols for community detection. In *26th European Symposium on Algorithms*, volume 112 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 7, 13. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

[7] Luca Becchetti, Andrea Clementi, and Emanuele Natale. Consensus dynamics: an overview. *ACM SIGACT News*, 51(1):58–104, 2020.

[8] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Stabilizing consensus with many opinions. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 620–635. ACM, New York, 2016.

[9] Luca Becchetti, Andrea Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Finding a bounded-degree expander inside a dense one. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms*, pages 1320–1336. SIAM, Philadelphia, PA, 2020.

[10] Luca Becchetti, Andrea Clementi, Francesco Pasquale, Luca Trevisan, Robin Vacus, and Isabella Ziccardi. The minority dynamics and the power of synchronicity. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4155–4176. SIAM, Philadelphia, PA, 2024.

[11] Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability—towards tight results. *SIAM Journal on Computing*, 27(3):804–915, 1998.

[12] Andrej Bogdanov, Kenji Obata, and Luca Trevisan. A lower bound for testing 3-colorability in bounded-degree graphs. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 93–102. IEEE Computer Society, 2002.

[13] Andrej Bogdanov and Luca Trevisan. On worst-case to average-case reductions for NP problems. *SIAM Journal on Computing*, 36(4):1119–1159, 2006.

[14] Andrea Celauro. Luca Trevisan, a beautiful mind. *Bocconi University* blog post, 19 June 2024. `https://www.unibocconi.it/en/news/luca-trevisan-beautiful-mind`.

[15] Irit Dinur, Martin Farach-Colton, Rosario Gennaro, Oded Goldreich, Sampath Kannan, Robert MacPherson, Ashwin Nayak, Luca Trevisan, and Günter Ziegler. Turing centennial series. *in theory* blog posts, May–July 2012. `https://lucatrevisan.wordpress.com/tag/turing-centennial/`.

[16] William Gasarch (ed.), Lance Fortnow, Oded Goldreich, Johan Håstad, Salil Vadhan, and David P. Williamson. Open problems column. *SIGACT News*, 55(4):32–48, December 2024.

[17] Uriel Feige, Shafi Goldwasser, Laszlo Lovász, Shmuel Safra, and Mario Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996.

[18] Lance Fortnow. Luca Trevisan (1971–2024). *Computational Complexity* blog post, 20 June 2024. `https://blog.computationalcomplexity.org/2024/06/luca-trevisan-1971-2024.html`.

[19] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM Journal on Computing*, 35(1):217–246, 2005.

[20] Oded Goldreich, Howard Karloff, Leonard J. Schulman, and Luca Trevisan. Lower bounds for linear locally decodable codes and private information retrieval. *Computational Complexity*, 15(3):263–296, 2006.

[21] W. T. Gowers. A new proof of Szemerédi's theorem for arithmetic progressions of length four. *Geometric and Functional Analysis*, 8(3):529–551, 1998.

[22] W. T. Gowers. A new proof of Szemerédi's theorem. *Geometric and Functional Analysis*, 11(3):465–588, 2001.

[23] Ben Green and Terence Tao. The primes contain arbitrarily long arithmetic progressions. *Ann. of Math. (2)*, 167(2):481–547, 2008.

[24] Venkatesan Guruswami, Nikhil Srivastava, and Kristin Kane. Remembering Luca Trevisan (1971–2024). *Calvin Café: The Simons Institute Blog*, 18 July 2024. `https://blog.simons.berkeley.edu/2024/07/remembering-luca-trevisan-1971-2024/`.

[25] Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, USA, 23-25 October 1995*, pages 538–545. IEEE Computer Society, 1995.

[26] Russell Impagliazzo and Avi Wigderson. *P = BPP* if *E* requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 220–229. ACM, 1997.

[27] Jonathan Katz and Luca Trevisan. On the efficiency of local decoding procedures for error-correcting codes. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 80–86. ACM, New York, 2000.

[28] James R. Lee, Shayan Oveis Gharan, and Luca Trevisan. Multiway spectral partitioning and higher-order Cheeger inequalities. *Journal of the ACM*, 61(6):Art. 37, 30, 2014.

[29] Laurent Miclo. On eigenfunctions of Markov processes on trees. *Probability Theory and Related Fields*, 142(3-4):561–594, 2008.

[30] Laurent Miclo. On hyperboundedness and spectrum of Markov operators. *Invent. Math.*, 200(1):311–343, 2015.

[31] Cristina Pace. Festa per il primo laureato in informatica: Luca Trevisan, 22 anni, ha discusso la tesi alla presenza del rettore tecce. *Il messaggero*, 21 July 1993.

[32] Omer Reingold, Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. Dense subsets of pseudorandom sets. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 76–85. IEEE Computer Society, 2008.

[33] Omer Reingold, Luca Trevisan, and Salil Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of cryptography*, volume 2951 of *Lecture Notes in Comput. Sci.*, pages 1–20. Springer, Berlin, 2004.

[34] Alex Samorodnitsky and Luca Trevisan. A PCP characterization of NP with optimal amortized query complexity. In *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing*, pages 191–199. ACM, New York, 2000.

[35] Alex Samorodnitsky and Luca Trevisan. Gowers uniformity, influence of variables, and PCPs. *SIAM Journal on Computing*, 39(1):323–360, 2009.

[36] Madhu Sudan and Luca Trevisan. Probabilistically checkable proofs with low amortized query complexity. In *39th Annual Symposium on Foundations of Computer Science, FOCS '98, November 8-11, 1998, Palo Alto, California, USA*, pages 18–27. IEEE Computer Society, 1998.

[37] Luca Trevisan. *in theory* (blog). https://lucatrevisan.wordpress.com/.

[38] Luca Trevisan. Lecture notes. https://lucatrevisan.wordpress.com/lecture-notes/.

[39] Luca Trevisan. Survey papers. https://lucatrevisan.github.io/pubs/index.html#surveys.

[40] Luca Trevisan. When Hamming meets Euclid: the approximability of geometric TSP and Steiner tree. *SIAM Journal on Computing*, 30(2):475–485, 2000.

[41] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[42] Luca Trevisan. On being second. *in theory* blog post, 7 November 2006. https://lucatrevisan.wordpress.com/2006/11/07/on-being-second/.

[43] Luca Trevisan. Max cut and the smallest eigenvalue. *SIAM Journal on Computing*, 41(6):1769–1786, 2012.

[44] Luca Trevisan. The Cheeger inequality in manifolds. *in theory* blog post, 20 March 2013. https://lucatrevisan.wordpress.com/2013/03/20/the-cheeger-inequality-in-manifolds/.

[45] Luca Trevisan. In theory celebrates the Turing centennial. *in theory* blog post, 27 May 2014. https://lucatrevisan.wordpress.com/2012/05/27/in-theory-celebrates-the-turing-centennial/.

[46] Luca Trevisan. Riemann zeta functions and linear operators. *in theory* blog post, 22 September 2014. https://lucatrevisan.wordpress.com/2014/09/22/riemann-zeta-functions-and-linear-operators/.

[47] Luca Trevisan. Turing centennial post 4: Luca Trevisan. *in theory* blog post, 2 July 2014. https://lucatrevisan.wordpress.com/2012/07/02/turing-centennial-post-4-luca-trevisan/.

[48] Luca Trevisan. Spectral graph theory and three experiences as an outsider. 7th TCS For All Workshop, 24 June 2024. Delivered by Pravesh Kothari and Salil Vadhan. https://youtu.be/O2gpl5l2eQA.

[49] Luca Trevisan, Gregory B. Sorkin, Madhu Sudan, and David P. Williamson. Gadgets, approximation, and linear programming. *SIAM Journal on Computing*, 29(6):2074–2097, 2000.

[50] Luca Trevisan, Madhur Tulsiani, and Salil P. Vadhan. Regularity, boosting, and efficiently simulating every high-entropy distribution. In *Proceedings of the 24th Annual IEEE Conference on Computational Complexity, CCC 2009, Paris, France, 15-18 July 2009*, pages 126–136. IEEE Computer Society, 2009.

Tribute to Arto Salomaa

A Towering Figure in Theoretical Computer Science

Mathematician and theoretical computer scientist Arto Salomaa passed away on January 26, 2025. He played a key role in the establishment and early development of the *European Association for Theoretical Computer Science (EATCS)*, and served as its President from 1979 to 1985. The following is an excerpt from the preface to a Special Issue of *Theoretical Computer Science (track C)* published in Arto Salomaa's honor.

Arto Salomaa was a world-class Finnish mathematician and theoretical computer scientist, renowned for his foundational contributions to automata theory, formal languages, and numerous other areas within theoretical computer science. His work profoundly shaped the mathematical foundations of the field and inspired generations of researchers.

His scientific career was truly international. As a graduate student, he received a scholarship to Berkeley, where he attended J. Myhill's automata theory seminar, which deeply influenced him. Upon returning to his home institution, the University of Turku, he established a research group in automata theory that became internationally renowned and continues to be active today, more than half a century later.

In the late 1960s, Arto spent three years at what became his second academic home, the University of Western Ontario in Canada. During this time, his first book, *Theory of Automata*, was published, marking the beginning of his extraordinary career as an author of scientific books. Shortly thereafter, he published his iconic monograph *Formal Languages*, a highly influential work that cemented his status as a pioneering figure in the field. Over the years, Arto authored more than ten highly influential scientific monographs. Some were masterful explorations of their respective topics, while others introduced or promoted novel areas of study, including formal power series, L-systems, public-key cryptography, and DNA computing. In collaboration with Grzegorz Rozenberg, Arto co-authored a comprehensive treatment of formal languages. This monumental work, the *Handbook of Formal Languages*, spans three volumes and over 2,000 pages, serving as a definitive reference in the field.

Beyond being an exceptionally original and creative researcher who continually explored new directions, Arto was also a leading educator in theoretical computer science. The remarkable clarity of his mathematical writing introduced many scientists to formal languages and automata theory, and his texts became fundamental to the education of multiple generations of researchers. Also, many of the Ph.D. students he guided went on to become prominent scientists worldwide, further demonstrating his profound impact as an educator and mentor.

Arto served three five-year terms as an Academy Professor of the Academy of Finland, the highest academic position in the country. In 2001, the Academy of Finland awarded him the title of Academician, a prestigious honor granted to only 12 individuals in the sciences nationwide.

He was a highly decorated scientist, a testimony to his prestige within the international scientific community. Arto was the recipient of nine honorary degrees and a distinguished member of several esteemed institutions, including the Finnish Academy of Science and Letters, the Finnish Society of Science and Letters, Academia Europaea, and the Hungarian Academy of Science as foreign member.

His achievements were recognized with numerous prestigious awards, such as

the Prize of the Foundation for Finnish Culture, the Magnus Ehrnrooth Prize of the Finnish Society of Science and Letters, the European Association for Theoretical Computer Science Award, the Finnish Professor of the Year award, the Nokia Foundation Prize, and the title of Honorary Professor of the Al.I.Cuza University in Romania. In his honor, the *Developments in Language Theory Symposium* series has introduced the prestigious annual Arto Salomaa Prize.

Arto made significant contributions to the scientific community. For instance, he played a key role in the establishment and early development of the European Association for Theoretical Computer Science, where he served as president from 1979 to 1985. Also, he was an active member of numerous editorial boards for prestigious academic journals and book series, further shaping the field through his expertise and leadership.

We, the editors of the *Theoretical Computer Science (TCS)* journal Special Issue in his honor, express our deep sorrow at Arto's passing. Each of us had the privilege of a warm and special friendship with him. We will miss him immensely and remain grateful for his presence in our lives and the profound influence he had on us.

Here's to celebrating a remarkable life and the extraordinary scientific legacy that Arto leaves behind!

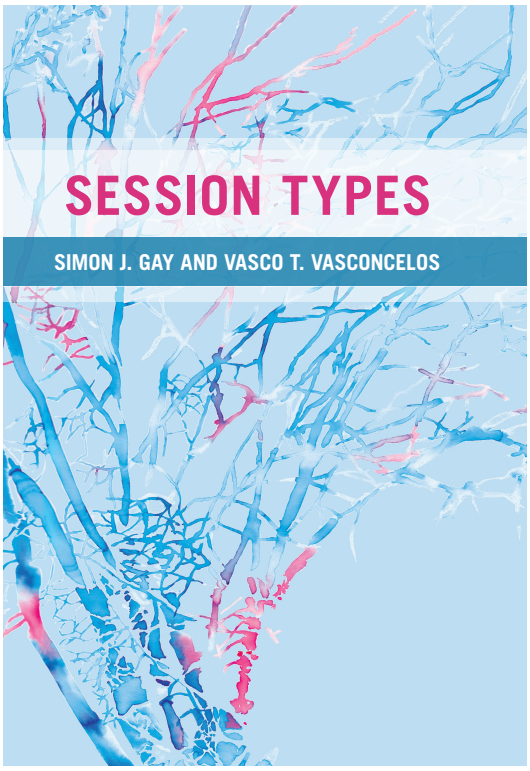
Juhani Karhumäki, Turku, Finland
Jarkko Kari, Turku, Finland
Lila Kari, Waterloo, Canada
Hermann Maurer, Graz, Austria
Ion Petre, Turku, Finland
Grzegorz Rozenberg, Leiden, The Netherlands and Boulder, Colorado, USA

February 2025

Miscellaneous

Book Announcement: Session Types

Simon J. Gay Vasco T. Vasconcelos



This book will be published by Cambridge University Press in April 2025.

Summary

Session types are type-theoretic specifications of communication protocols in concurrent or distributed systems. By codifying the structure of communication, they make software more reliable and easier to construct. Over recent decades, the topic has become a large and active research area within the field of programming language theory and implementation.

Written by leading researchers in the field, this is the first text to provide a comprehensive introduction to the key concepts of session types. The thorough theoretical treatment is complemented by examples and exercises, suitable for use in a lecture course or for self-study. It serves as an entry point to the topic for graduate students and researchers.

Contents

Preface

The topic of *session types* concerns the use of programming language type systems to specify and verify the communication behaviour of programs. Just as data types describe the structure of data and constrain the operations that can be performed on it, session types describe the structure of communication and constrain

the communication operations that can be performed. A session type is a formal description of a communication protocol, made accessible to programming language tools such as typecheckers, compilers, IDEs and runtime monitors.

The theory of session types was developed by Kohei Honda and his collaborators at Keio University during the 1990s. Honda's paper *Types for Dyadic Interaction* contains the key ideas, but is not always cited as the original paper. It was followed by *An Interaction-Based Language and Its Typing System*, co-authored with Kaku Takeuchi and Makoto Kubo, and *Language Primitives and Type Discipline for Structured Communication-Based Programming*, co-authored with Vasco Vasconcelos and Makoto Kubo. The latter paper included a range of examples showing that the type system could describe interesting protocols such as FTP. It was published at the ESOP conference, and is noteworthy for being 17 pages long at a time when ESOP papers were restricted to 15 pages. Kohei persuaded the programme committee chair that the paper was important enough to justify extra space.

None of the original papers used the phrase "session types", although they referred to both sessions and (of course) types. The phrase "session types" first appeared in the paper *Types and Subtypes for Client-Server Interactions* by Simon Gay and Malcolm Hole. Subsequently it became established as the name of the research topic, and at the time of writing, Google Scholar lists more than 500 papers with "session types" in the title. Most conferences on programming languages now include papers on session types every year, often with a whole session devoted to them.

The purpose of this book is to give a clear presentation of the core theory of session types. More precisely, we cover *binary* session types, which deal with interaction between pairs of agents. There is a large literature on *multiparty session types*, introduced by Kohei Honda, Nobuko Yoshida and Marco Carbone in the paper *Asynchronous Multiparty Session Types*, which extends the theory to collective protocols among multiple agents. In order to keep the book to a reasonable size with a clear focus, we have not included multiparty session types. They deserve a book of their own.

After an introductory overview in Chapter 1, Chapter 2 covers the basic theory of binary session types in pi calculus. In this chapter, all behaviour is finite and there is no sharing of channels.

Chapter 3 introduces infinite behaviour in both types and processes, detailing a coinductive foundation for a convenient finite syntax. Chapter 4 adds the possibility of shared channels. At this point we reach a language with similar expressiveness to the one presented in Vasco Vasconcelos' paper *Fundamentals of Session Types*, although the presentation is a little different.

Chapter 5 extends the theory to include subtyping, to support a more flexible approach to judging when two components can communicate safely. Chapter 6

adapts the declarative type system of Chapter 5 into an algorithmic system suitable for implementation as a type checker.

In Chapter 7 we present the ideas from our own paper *Linear Type Theory for Asynchronous Session Types*,integrating session types into a multithreaded functional programming language.

Chapter 8 presents work by Ornela Dardha, Elena Giachino and Davide Sangiorgi, showing how session types can be encoded in more primitive type-theoretic constructs. The encoding was originally published in *Session Types Revisited*.

Chapter 9 covers a Curry-Howard correspondence between session types and linear logic. This important direction in the field was opened up by Luís Caires and Frank Pfenning in the paper *Session Types as Intuitionistic Linear Propositions* and is still being explored. Our presentation follows Philip Wadler's reformulation of the correspondence in the paper *Propositions as Sessions*.

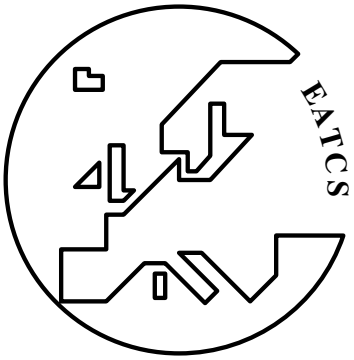
Endorsements

> "... a comprehensive account of the basic theory and [...] possible applications, beautifully leavened with insightful examples and exercises. It will become—and well deserves to become—the definitive reference textbook on session types." *Davide Sangiorgi, University of Bologna*

> "How should we structure distributed computation? One of the most promising approaches is session types. Here we have a superb introduction by two founders of the field. Gay and Vasconcelos provide a readable yet thorough introduction. It has long been needed and will help the field flourish." *Philip Wadler, University of Edinburgh*

> "This book offers a deep view on the core theory of binary session types. The reader is captivated by the appropriate and clarifying examples and stimulated by the challenging exercises. The authors are outstanding researchers as witnessed by the simplicity and rigour of their writing." *Mariangiola Dezani-Ciancaglini, University of Turin*

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
- Paris, France 2022
- Paderborn, Germany 2023
- Tallinn, Estonia 2024

(2) THE BULLETIN OF THE EATCS

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

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

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

(3) OTHER PUBLICATIONS

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

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

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

The editors of the 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. Giuseppe F. Italiano,
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:

Dmitry Chistikov
Computer Science
University of Warwick
Coventry
CV4 7AL
United Kingdom
Email: `secretary@eatcs.org`,