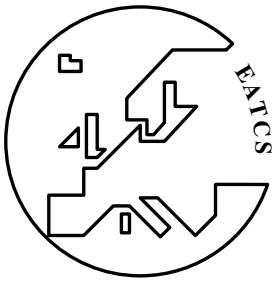
ISSN 0252–9742

Bulletin

of the

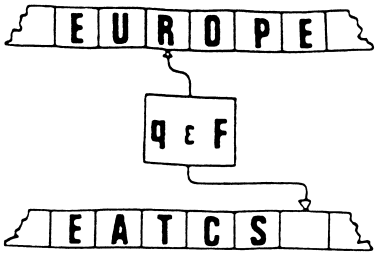
European Association for Theoretical Computer Science

EATCS



Number 147 October 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`.

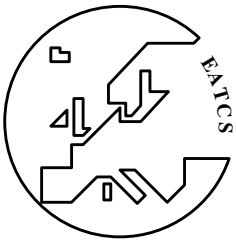
Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

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

EATCS Matters



Dear EATCS members,

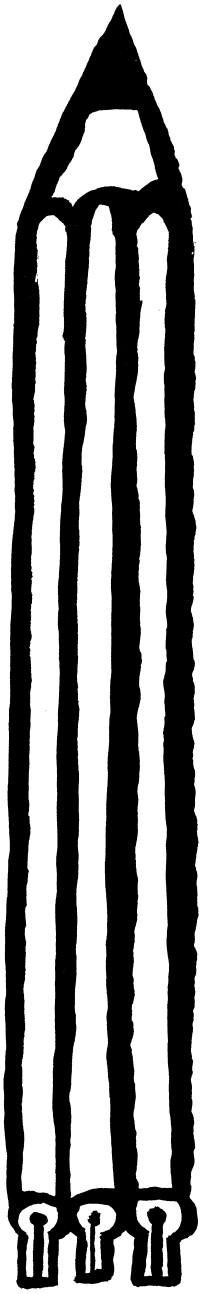
I hope you had a refreshing summer and are ready for the challenges and opportunities of the new academic year.

It was a pleasure to meet many of you at ICALP 2025 in Aarhus. The strong participation of young researchers and students was particularly encouraging, and I was glad to see such good attendance at the EATCS General Assembly. Thank you to everyone who took part.

The 52nd edition of ICALP in Aarhus was a success both scientifically and socially. The Program Committee put up an impressive scientific program, and the organizers worked hard to create a memorable event. I believe all participants felt welcome and engaged. On behalf of EATCS, I warmly thank the Program Committee Chairs (Keren Censor-Hillel, Fabrizio Grandoni and Joel Ouaknine), the Program Committee Members, the Organizing Committee Chairs (Ioannis Caragiannis and Kasper Green Larsen) and the local team for their excellent work.

Looking ahead, ICALP 2026 will take place in Royal Holloway, University of London, co-located with PODC and SPAA, on 6-10 July 2026. I encourage you to submit your best work and to join us in Royal Holloway for what promises to be another stimulating conference.

The EATCS General Assembly meetings in Aarhus were productive. While progress was made, much more could be achieved - especially in supporting young researchers - if additional resources become available. I strongly encourage you to join the EATCS



and to invite your students and colleagues to do the same. The membership fee is modest, and your participation directly supports our community and advances Theoretical Computer Science.

In a few months we will have calls for nominations for many awards, including the EATCS Award, the Presburger Award, the EATCS Distinguished Dissertation Award, the EATCS Fellows, and some other joint awards, such as the Gödel Prize, the Alonzo Church Award, and the Dijkstra Prize. Let me take the opportunity to thank all the Award Committee Members for their important service to our community. I encourage you to nominate excellent researchers and papers for these recognitions; all prizes will be presented at ICALP 2026.

Finally, please remember that your input is always welcome. You can write to me at president@eatcs.org with ideas on how to increase the impact of the EATCS in our community and how to make membership more appealing to young researchers - the future of our field.

With my best regards,

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

October 2025

Dear EATCS member!

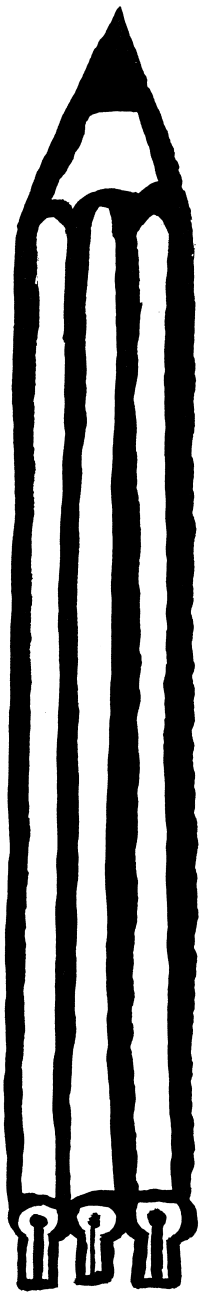
In front of you is Bulletin 147. An issue in which we welcome several new editors! And in which we also introduce a new column and a new initiative.

Pablo Barcelo and David Saulpic are our new editors for the new column on AI topics in TCS. In their first column, they present a nice and wide review on how different areas of TCS have contributed to recent advances in ML, or how it could contribute.

Anuj Dawar is the new editor of the Logic in Computer Science Column, stepping into the shoes of Yuri Gurevich. Welcome aboard! And many thanks Yuri for all your great contributions to the Bulletin over so many years - since 1988! In his first column, he explores the question about the smallest number of variables with which a formula can be equivalently written, an important measure related to notions of width arising in database theory, graph theory and permutation groups.

Furthermore, I am very happy to announce a new initiative, which aims to accompany the Bulletin with videos. The initiative is led by Sophie Huiberts and Ian Mertz. In his essay in the Perspective Column, Ian gives a motivation and overview of this initiative, and also advertises a call for videos for EATCS.

In the Computational Complexity Column, Ryan Williams provides a background exposition of a recent theorem in computational complexity related to how fast algorithms can be simulated with less memory.



In the TCS on the Web Column, Stefan Neumann talks to the current maintainers of the TCS Blog Aggregator: Nima Anari, Arnab Bhattacharyya and Gautam Kamath. The TCS Blog Aggregator is a community-maintained hub for researchers in complexity theory and algorithms.

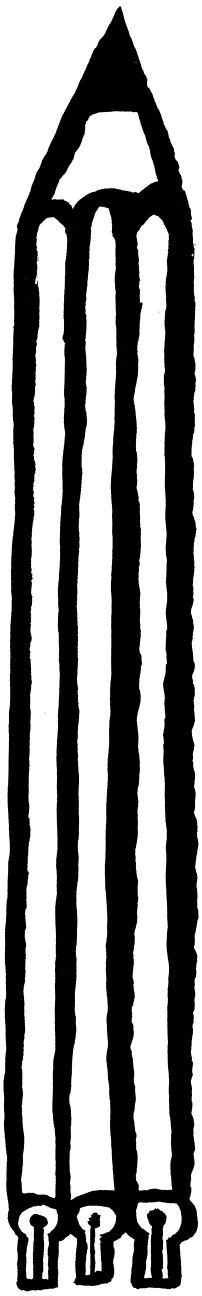
In the Formal language Column, Antonio Casares studies transition-based vs stated-based acceptance for automata over infinite words, and advocates using the former. He presents a collection of problems where the choice of formalism has a major impact and discuss the causes of these differences.

In the Distributed Computing Column, Amitabh Trehan provides insights into a most simple yet surprisingly deep algorithmic problem: flooding with no memory.

In the Education Column, R. Ramanujam explores whether concepts from formal logic can meaningfully contribute to mathematics education at the school level, reporting on an attempt to work with high school teachers of mathematics in India.

For the jubilee year of the DISC conference, the "DISC historians" Michel Raynal and Nicola Santoro tell us the story of WADS/DISC since 1985, a sibling to the then 4-years old PODC conference. They also share their thoughts of what makes a good conference in general, and share several nice historical photos.

The Bulletin also contains several nice conference reports: Matthias Bentert provides a spotlight of the ALGO conference, Florian Chudigiewitsch of STACS, and there is also a summary of the



British Colloquium for Theoretical Computer Science.

Last but not least, in case you are in Prague soon, you are welcome to check out the exhibition 'The Making of a New Science,' on the early history of theoretical computer science.

Enjoy the new Bulletin!

Stefan Schmid, Berlin
October 2025

Institutional Sponsors

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

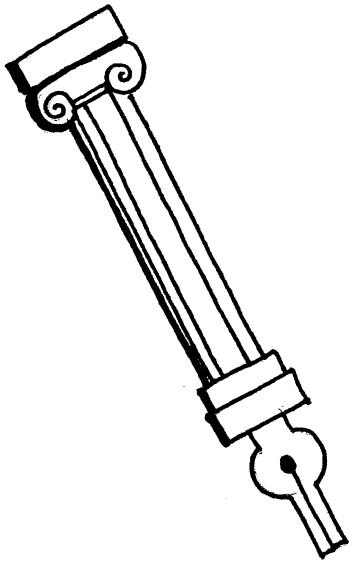
CWI, Centum Wiskunde & Informatica
Amsterdam, The Netherlands

MADALGO, Center for Massive Data Algorithmics
Aarhus, Denmark

Microsoft Research Cambridge
Cambridge, United Kingdom

Springer-Verlag
Heidelberg, Germany

EATCS
Columns



The Viewpoint Column

BY

Stefan Schmid

TU Berlin, Germany
stefan.schmid@tu-berlin.de

BEATCS no 147

We Should Be Making Great Video Talks

Ian Mertz

Abstract

Video talks, a staple of modern conferences and a great accessibility tool, are nonetheless maligned for being inferior versions of traditional live talks. We push back on this from two perspectives: first, that with clearer goals in mind video talks can be made to perform different functions than live talks; and second, that with basic editing video talks can be every bit as good as live talks, and can even reach creative heights unachievable in non-recorded settings. We make the case that creating such purposeful, engaging video talks is a benefit to the field, and should become more normalized therein.

1 The Current State of Video Talks

Since the onset of the COVID pandemic in early 2020, prerecorded talks have become a regular feature of TCS conferences everywhere. In a time where a large fraction of the community does not travel, whether for health reasons, home obligations, budget restrictions, environmental concerns, visa issues, or the shifting political climate of the hour, there is much to appreciate about this dual component to our live events. Even without these considerations, having talks which are rewatchable on demand, which can be slowed down or sped up at will, and which can reach anyone in the world (including those not on the conference circuit) is a great boon.

It is also widely agreed that these talks, frankly, suck. Trying to conjure up the energy of a live talk while going through slides in a silent, empty room is nearly impossible, and the effect is made worse by a disparate audience often watching in silent, empty rooms themselves. Every fumble hangs in the air, technical issues and poor mic placement quickly become grating, and the impersonality of it all can be more tangible than the content itself. If a traditional slide talk is meant to be polished, as a recording it often feels more mechanical than anything else.

This situation reinforces, and in turn is reinforced by, the unspoken yet assumed role of video talks, made especially clear since the overall return to in-person conferences: video talks are a useful yet inferior substitute for the live presentation, designed chiefly for those who cannot attend or for those scouting out which sessions they will attend. Only a handful of conferences, such as ITCS,

have given distinct roles to the in-person and video talks, where one serves explicitly as the condensation of, and advertisement for, the other, and even in these cases there is little emphasis put on the recorded portion on the conference website, during the submission process, etc. In short, recorded talks are treated with a utilitarian purpose, whether for accessibility at the conference or a quick memory refresh down the line, rather than something to be produced for, or considered on, its own merits.

In this paper we seek to push back on all of these assumptions and practices. We will argue two closely entwined points. First, video talk requirements should be designed with clearer purposes in mind, including ones orthogonal to, but no less important than, those of live talks. And second, in this capacity video talks can be not just good but spectacular, and can take advantage of the unique advantages of prerecording to even surpass live talks in many ways. We believe that making better and more purposeful video talks is a great potential asset for our field, and that, for reasons of equity, creativity, and outreach, we should make such content more respected, encouraged, and, ultimately, commonplace.

2 Why Make Video Talks?

We begin with the question of purpose. To start, we should ask: what is the purpose of a talk in general? While there are immediate ideas that come to mind—that they give the motivation and ideas of a paper at a higher level, for instance—the real answer is that it entirely depends on the format and context.

Let us take an example. Imagine a well-designed conference talk, timed and practiced to 30 minutes on the dot: 10 minutes of introducing and motivating the model, 5 minutes of prior work, 2 minutes of explaining the current results, 10 minutes of high-level proof ideas, 2 minutes of extensions and other implications, and one last minute for open questions. This talk is occurring at a top-level conference for the subfield, meaning much of the core language is shared but most people will come from outside the specialization, and thus over half of the talk is setup and the proof is restrained to the big ideas, yet leaving enough to add in a few hooks for the experts in the audience.

Now imagine the same speaker is asked to present the same work for a flash talk of only 5 minutes; is it correct to linearly scale everything down, with 3 minutes of setup, 2.5 minutes of proof, and thirty seconds of wrap-up? What about an invitation to give a two-hour technical seminar; is it correct to leave any mention of the proof until after the break, filling the first hour with more expansive details about the model and prior work?

Specifying the length of a talk is not just a matter of time constraints, but rather a signal as to the purpose of the talk itself, such as whether it is meant to be an

advertisement or a fully technical deep dive. The best talks use this knowledge, just as much as the target audience and the contours of the result and subfield, in a fundamental way.

Timing is just one technical component of a presentation, and this question directly applies, and is perhaps more widely understood, when considering different mediums. Just as adapting books to film requires understanding the differences between written and visual medium, a paper and a talk are fundamentally trying to fulfill different goals even when given comparable constraints on the amount of material to be covered. This further applies between variants within a given medium; within the realm of talks, a slide talk is a more polished and prepared exposition, while a board talk is typically more conversational and interactive; meanwhile plenty of written forms exist beyond papers, such as lecture notes and blog posts, both of which are more high-level but with the former typically discussing proof/technical ideas and the latter more context-driven.

Going back, then, to prerecorded talks, the question becomes what purpose we want to accomplish with a conference video requirement, taking into account what video itself is best suited for, and how to set the specifications around these goals. The most obvious factor of a video talk is its medium, namely an audio-visual presentation probably focused around slides and designed for a technical audience, but frequently we stop there and lump them together with slide talks without further ado. This also leads to time constraints being set in line with the corresponding live presentation, but unlike the conference version there are no obvious reasons to put such limits on a video (one byproduct of this is that people frequently overrun these time constraints, as they undoubtedly would, if they do not already, given the option in a live setting). Thus rather than deciding the length and type of talk as a result of the time constraints inherent to running a conference, recording and uploading videos automatically gives us the freedom to work backwards and set these restrictions based on the purpose we want these talks to fulfill.

Even more salient than the lack of hard time constraints is the relationship of presenter to audience. On one hand, the creator has unprecedented level of control over a video; on the other, it is the ultimate model of one-way information, akin to papers in lacking even basic feedback. We can consider a spectrum of styles, where casual, relaxed interactivity constitutes one pole while a tightly curated information channel forms the other, on which we can consider any given medium. For written works this may consist of forum/social media discussion on one end and journal publications on the other, with blog posts and lecture notes sitting in between; for presentations, discussion groups and videos sit at opposite extremes and with board and slide talks falling in the middle towards the discussion and video ends respectively. This does not prescribe videos to being the most purely informational form of talks, but rather encourages us to look to practices in good

paper-writing—specifically those practices which sit in opposition to more casual forms—to inform how we might use this spectrum to our advantage.

With these ideas in mind, just as we imagine conferences whose purposes are specialized and distinct from simply soliciting top theory papers writ large—again ITCS comes as an example—we can imagine a myriad of alternatives to recording 25 minute slide talks in an empty room. A conference can solicit 5-minute advertisements for the main talk, in which case we can look to actual advertising practices and hone in on a blitz of big takeaways with broad appeal, respecting the listener's time by giving them everything they need to decide whether or not to commit to the longer session. We can flip the script and have the live talk be the advertisement for a longer technical presentation on recording, which can serve as a lasting guide to deeply reading the paper. In service to the community, we can even ask for videos which focus only on preliminaries and surveying past results, becoming a go-to reference for understanding a subfield long after the conference is over.

Besides a call to conference organizers to revisit the question of why and how to solicit video talks, as presenters we can think about how best to use video not just as an reflection of the live presentation but as a complementary product serving purposes the live presentation cannot meet. Beyond the clear accessibility benefits we all accept from video talks, there is a freedom to explore them as a completely different mode of communication, one which utilizes both time and technical constraints to tightly curate the viewing experience.

3 Can Video Talks Be Good?

We now move to the second question: can prerecorded talks ever be good enough to make these freedoms and benefits worthwhile? We argue that, alongside leaning into the aforementioned differences between prerecording and traditional talks, the ability to edit can make videos truly shine. This holds along a large spectrum of approaches: basic editing, i.e. cutting out dead space and bad takes, is doable with low additional effort and would immediately make many videos more enjoyable to watch and rewatch; while for the truly motivated and inspired, making a fully formed piece of video content can be both personally rewarding and of incalculable benefit to the field.

3.1 Basic Editing: Low Cost, Major Benefits

Let us start from the classic image of a recorded talk, a speaker presenting a slide show to an empty zoom call. A practiced talk generally has few flubs, but when speaking to the computer it seems that many people will inevitably misspeak a few

extra times, especially since such a mistake, hanging awkwardly in the air, tends to beget another. The more devoted may stop and restart after a take with too many errors—such is one of the more obvious advantages of recording—but even for such speakers, or in fact especially in cases where a speaker is willing to spend extra time and effort to produce a polished product, the speaker's engagement can visibly begin to wane, important lines or explanation get skipped without notice, and their voice, bereft of earlier sharpness, starting to trail off as the technical section goes on. And in this situation, one overly long pause, possibly coming minutes after the stated time limit has been passed without comment, is space enough for the audience, sitting at home with a playlist of videos to watch the week before the conference, to zone out or simply move on.

The absolute basic tenant of editing is the ability to cut a recording into arrangeable and malleable clips, from which the final video is assembled. From this alone we can immediately tackle many of the issues plaguing our previous speaker. A flub or pause is no reason to restart, as one can simply stop and restart the sentence, cutting out the bad take in post-production. If they feel that they are making too many errors, or that the video is starting to drag, the speaker can stop, go take a break, and come back some time later to continue shooting. In some important sections it may even be worth stopping line by line, in order to make sure that each one comes across clearly; this has the added benefit that the cuts create a slight discontinuity which can snag the viewer's attention before it starts to drift. Missed lines can be recorded and added even days later, pacing can be controlled through massaging the space between and during lines, and fitting timing constraints need not require timing the talk so precisely ahead of time, as less crucial lines can be removed at the end of the process.

From here it is a small step to add additional minor elements that break up the monotony of a longer talk. For those who bolstered their Zoom setup during the pandemic by buying a separate camera and microphone, patching audio and video together is just a few clicks on any editing software. Adding external sounds, such as a snippet of music in place of speaking between talk sections or as an opening grab, is equally trivial. And playing with the prominence of video elements can bring an entirely new dimension beyond slides, from looping small animations for key subroutines, mimicking a "board talk" through recording a live derivation on a tablet, running code live in the background to understand performance guarantees in situ, or simply having the presenter front and center with slides and other technical parts only appearing as necessary.

There is, of course, a cost to editing, namely time and effort. While these basic edits are routine for those experienced with editing, there is still a learning curve and the first few videos do undoubtedly take longer, but resources are available to get over the hump. Sophie Huiberts has designed a suite of tutorials in tandem with this piece such that those interested can quickly progress and start knocking

out edits as a matter of course. Universities often have their own resources, both in terms of equipment and software and with regards to tutorials, in-house video producers, or even funding to hire editors, a minuscule fraction of the total cost of attending the conference itself. And when calculating the overall effort, it is worth remembering that even early unpracticed editing may be faster than constantly restarting, and with the latter coming across much cleaner in the final release.

3.2 Video Production: Creativity Unbound

If basic editing is the floor when it comes to videos, then the ceiling is very high indeed. Far from being bound to a slide deck, modern video editing software and accompanying tools allow virtually limitless creative freedom, a freedom which, in turn, can be used to convey information in ways more tailored to the audience and content at hand.

We can look to the top three most viewed STOC video talks on the SIGACT YouTube channel, all of which utilize completely distinct styles borrowed from popular science videos on the same platform, for examples of this creative potential. On the topic of shortest even cycles we have no human face but a tightly edited audio set over beautiful visualization software, in this case the Manim package developed by the hugely popular 3Blue1Brown channel (in fact this software is already in use for many live popular talks as well). The smoothed complexity of the simplex method video, by contrast, uses a face-plus-overlay approach to counteract the "impersonality" of recordings by putting the presenter front and center and using an easygoing conversational presentation style; this appears in popular videos like those of Dr. Angela Collier, who was the stated inspiration for the video. And a variety of experimental essay-style videos, particularly those by political creators, have become one of the hallmarks of YouTube as a whole in the modern era, with a crossover attempt[1] on the topic of space complexity appearing for the STOC audience as well.

Experimentation, in this case with form and presentation, is itself the essence of good science, and therein carries on a tradition—perhaps now less common with papers but still alive and well with blogs, threads, and lectures—of finding new ways to engage with audiences and teach mathematics. This is in many ways a necessary experimentation, as each field, result, and presenter is unique and should be handled as such. This meta-form of scientific discovery, moreover, is egalitarian in a way that the cutting-edge science it teaches inherently struggles with, namely that it can come from novices with a creative vision or even by

[1]This video, made by the present author, is shot over four locations and features costume changes, music, a sped up tablet section, color correction, audio balancing, and more; judgments on the success of this endeavor, however, are left to the reader.

chance rather than demanding years and years of study with near surety. This holds doubly so for video, a medium where, if anything, younger researchers have a massive edge with regards to intuition and technical know-how. Not every video will succeed in the creative goals it sets out to achieve, but there is little lost in an unsuccessful experiment besides effort on the part of the presenter, and this inevitable cycle of failing and learning is as integral a piece of the research process to teach to junior researchers as the science itself.

4 Why Put Effort Into Video Talks?

Finally, even accepting that recorded talks can be made both purposeful and polished, it seems nevertheless reasonable to ask whether making purposeful, polished recorded talks is worth the corresponding time and effort. Such videos are an unambiguous good for the viewers themselves, akin to the clarity that comes from a well-written paper, but even from the presenter's perspective this work demonstrably pays important dividends.

To frame this discussion, we must acknowledge first and foremost that these videos do, in fact, perform well. Revisiting the three STOC videos discussed above, we note that, excluding live recordings of plenary talks, these are the three most watched videos on the SIGACT page, with 2000-2500 views each to date. By contrast, the first non-edited talk, again excluding plenary and live recordings, clocks in at 1200, and even the most popular recording of any best paper talk over the past four years sits at just over 1000. Besides having double the view count, 2000 viewers is *four times* the average live attendance at STOC itself over this time; in fact we could confirm that for at least two of these three edited videos, their view counts had surpassed their respective live attendances before those conferences had even started.

Thus from the perspective of the speaker, this positive viewer experience directly translates to more exposure of their ideas to the scientific public. Many first-day conference discussions revolve around which videos people have watched and are interested in attending as a result—such is the goal of prerecorded talks as live talk advertisements, after all—and so more eyes can mean greater attendance for the "real" presentation as well. When a talk, whether live or online, stands out from the crowd, it also leaves a greater impression on viewers that increases the chances of them revisiting the result later; in this, a strong recorded talk can perform double duty, as it will also be the version any viewer who wants to revisit a presentation invariably returns to.

Beyond this, however, it is interesting to consider the further reach of those ideas to the public writ large. As mentioned above, popular computer science and mathematics thrive on platforms like YouTube, with well-produced videos at all

levels of rigor and detail garnering massive view counts; to take one example, a video by the generalist producer Vsauce on the Banach-Tarski paradox has the second-highest view count on the channel, having been watched over *46 million times*. These videos are often further disseminated through a variety of hubs, such as video competitions and festivals, amateur forums for boosting and discussing such topics, reactions and other "follow-on" content made by viewers, and other casual channels generally not touched by the more secluded work at the forefront of the field. And they are monolithic in neither style nor purpose; these can run the gamut from popular science articles which give viewers a taste of high-level research to more thorough investigations of a particular question or field.

In choosing a length and level of technical depth, in choosing what are the necessary preliminaries and which gritty details to sweep under the rug, above all else these presenters are choosing a target audience from among the internet public, an unrepresentative yet extremely broad swath of the broader population, and building a talk to fit both the purpose and aesthetic tastes of that audience. While STOC or FOCS attendees, for example, should be expected to have more background and mathematical maturity than an average viewer on YouTube, this approach still fits squarely with the question of speaking to a general theory crowd, and so there is no reason a well-constructed video talk hosted on a public platform cannot act as both within the field as a paper overview and externally as a more-technical-than-average advertisement for the field. At the very least, the success of such "amateur" videos should, perhaps, prompt us to look at the power of making recorded talks with purpose and quality in mind.

5 The Future of Video Talks

The current essay began from a thought, namely that it seems strange that many of us have collectively decided that a talk which we can edit—a talk where we can cut out dead space, redo bad takes, or restructure after reviewing a full draft—is *inherently worse* than a traditional talk. It is one thing to assert that awkward pauses, vocal tics, and other interruptions are more halting in a video than live, but it is quite another to assert that a video without these interruptions is still worse than a live talk laden with them, let alone when the video allows itself the audiovisual freedoms of the medium. And while it may be intimidating or even paralyzing to enter into the world of editing, the costs of learning this new skill, once amortized over a handful of submissions, are more negligible than they may feel from the outside.

The prerecorded talks component of conferences are not, and for many reasons should not be, leaving anytime soon. While the field has generally made peace with this fact, we may have missed the opportunity to truly embrace video talks

as a novel frontier for presenting computer science and mathematics at the highest levels. They are not without disadvantages; even for the most engaging styles the lack of human connection stands in sharp contrast to the live talks they so often walk in the shadow of. However, in light of their advantages, such as clarity and comprehensive level of control, it seems more useful to think of them as an orthogonal, rather than inferior, medium. And with in-person conferences still featuring live talks front and center, we have no need to pit these orthogonal modes against one another: we can really have it all.

Insofar as we accept and embrace video as a mode of presentation, it seems only natural to put in the effort to make it work. As with both papers and live talks of all stripes, videos with a baseline level of production and editing are more enjoyable and should be lauded, while those that go above and beyond and succeed therein should be viewed as perennial classics. While there may be a chicken-and-egg problem with winning such acceptance, the existing forays into this realm, both inside and outside our community, are already viewed in this way; the question, then, is simply whether we choose to make them the exception or the norm.

This is not meant to be prescriptive of how such videos look and feel. There is no single correct way to make a video just as there is no correct way to write a paper or give a talk; it is the diversity of presentation styles, along with experimentation therein, which drive the field forward. However, if we have resigned ourselves to feeling that every way to make a video is wrong, a view that is both untenable and plain incorrect, then it is our contention that we all benefit from working towards a new paradigm where videos are more meaningful, impactful, and, above all else, enjoyable.

Afterword: A Call for Videos for EATCS

This piece accompanies the beginning of the new EATCS YouTube channel for pre-recorded talks, an initiative launched by Stefan Schmid and hosted by Sophie Huiberts and myself. We invite speakers to pitch potential videos at any technical level with an emphasis on clarity of presentation and quality of production rather than topic; the submission and editing process will go through the channel rather than the bulletin and will be released throughout the year. We also invite all contributors to the Bulletin to submit video accompaniments to their submitted article, or even video-only contributions, to be released alongside the written bulletin, which will go via the ordinary bulletin soliciting/editing procedure. Our primary goal is to help lead the way in videos occupying a prominent place in TCS by becoming a hub for the best video talks our field has to offer.

Link to the channel: https://www.youtube.com/@eatcs

Acknowledgments

We thank Sophie Huiberts for years of conversations (and proofs of concept) on the topic of producing video talks and Stefan Schmid for the opportunity to publish this work in tandem to the channel launch. We thank Sophie Huiberts, Fran McManus, and Kristie Petillo for feedback on earlier drafts of this work.

TCS ON THE WEB

BY

STEFAN NEUMANN

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

The TCS Blog Aggregator[1] is a community-maintained hub for researchers in complexity theory and algorithms. It collects posts from roughly 50 theory blogs and community sites, plus recent open-access papers from arXiv (cs.CC, cs.CG, cs.DS) and the ECCC, and presents them in a single chronological feed.

Today I am more than happy to talk to the current maintainers of the TCS Blog Aggregator, Nima Anari[2] (Stanford University), Arnab Bhattacharyya[3] (University of Warwick) and Gautam Kamath[4] (University of Waterloo).

[1] https://theory.report
[2] https://nimaanari.com
[3] https://www.dcs.warwick.ac.uk/~u2470130/
[4] http://www.gautamkamath.com

THE TCS BLOG AGGREGATOR

A Conversation with Arnab Bhattacharyya and Gautam Kamath

Arnab and Gautaum, thank you for taking time for this interview. Can you give our readers a short overview of the TCS Blog Aggregator and its history?

Arnab: The TCS blog aggregator was originally the brainchild of Arvind Narayanan[1] who created it back in 2007. At the time, the TCS blogosphere was rapidly expanding, and Arvind's simple setup at `feedworld.net/toc` became hugely popular. It periodically polled a set of TCS blogs and displayed the posts in reverse chronological order.

In November 2018, the original domain name expired, and Arvind could no longer maintain it in his own personal capacity. Suresh Venkatasubramanian[2] took the lead in reviving the aggregator, and I (Arnab) joined his effort. I set it up at a new domain (`cstheory-feed.org`), to mirror the location of the jobs website at `cstheory-jobs.org`[3]. We wanted to make the setup more transparent, and if someone wanted to add a new blog to the roll, they would contact one of us who would make the change. Soon after, Gautam also joined the team to help maintain the blog roll.

Things continued this way till 2022 when there was a period of time the site was down due to a technical issue. Over this period, Nima Anari[4] independently experimented with creating his own site using Github Actions, which obviated the need for having a private server to host the code for polling. His codebase was also more streamlined, modern, and easier to maintain. Nima replaced Suresh as a maintainer, and the aggregator now uses Nima's implementation hosted at `theory.report`[5].

How would you say the TCS blog landscape has changed over the last, say, ten years?

Gautam: My recollection only goes back to the early 2010's, when I started grad school. It felt like a vibrant ecosystem: there would be some news in the

[1] `https://www.cs.princeton.edu/~arvindn/`
[2] `https://vivo.brown.edu/display/suresh`
[3] `https://cstheory-jobs.org`
[4] `https://nimaanari.com/`
[5] `https://theory.report`

community (e.g., a major result is proven, someone wins a big award, etc.), and all the blogs would discuss it, adding their own perspectives and going back and forth with their opinions. Many grad students in my program followed the blog aggregator, and the latest news and gossip were the topic of many a hallway conversation.

I fear that some of that sense of community has since dissipated. Most of the content of the blog aggregator currently is the arxiv feed from cs.CC, cs.CG, and cs.DS. While a few of the longest running blogs are still going strong (e.g., Computational Complexity[6], Shtetl-Optimized[7]), several have become less frequent or gone dormant. It doesn't feel like people are so plugged into the "TCS blogosphere," instead focusing more on individual blogs and bloggers. I suspect that people now find community in other places, for example "microblogging" (e.g., Twitter/X, Bluesky, Mastodon). This has pros and cons: on the "pro" side, these sites enable more voices to enter the conversation, since they have a much lower barrier to entry.

Are there any blog posts you'd like to highlight from over the years?

Gautam: I really like posts that share more about the humans behind the research. Omer Reingold ran a series on "research-life stories,"[8] a great representative one is Bobby Kleinberg's[9]. Luca Trevisan had a series of posts from gay and lesbian computer scientists in commemoration of Turing's 100th birthday[10], you can read his own account here[11].

Arnab: Completely agreed with Gautam. I'd also like to highlight Gil Kalai's[12] valiant effort at "Combinatorics and More."[13] Gil's blog has been actively maintained since 2008! It's still often the first place to break news about advances in combinatorics.

When our readers start their own blogs, how can they get them added to the TCS Blog Aggregator?

Gautam: It's easy to submit a pull request to the Github repo at `https://github.com/nimaanari/theory.report`. If this is unfamiliar to you, just email the maintainers a link to your blog.

[6] `https://blog.computationalcomplexity.org`
[7] `https://scottaaronson.blog`
[8] `https://windowsontheory.org/tag/research-life-stories/`
[9] `https://windowsontheory.org/2013/03/21/research-life-stories-bobby-kleinberg/`
[10] `https://lucatrevisan.wordpress.com/tag/turing-centennial/`
[11] `https://lucatrevisan.wordpress.com/2012/07/02/turing-centennial-post-4-luca-trevisan/`
[12] `http://www.ma.huji.ac.il/~kalai/`
[13] `https://gilkalai.wordpress.com/`

Before we conclude the interview, could you please let us know what your current research is about?

Gautam: My research right now is on algorithmic statistics and machine learning, particularly with a focus on considerations such as data privacy and robustness. I find it enjoyable because even fairly basic problems are fun to think about, the topics can get theoretically deep, and it's possible to say something interesting about practical settings that people care about.

Arnab: My current research area is also algorithmic statistics, with a focus on learning and testing high-dimensional distributions.

Finally, is there anything else you want our readers to know?

Arnab: Thanks for this opportunity to look back at the history of the feed aggregator and reflect on how the TCS community has evolved over the past two decades. Another example of a website that was much more active in the 2010s is `cstheory.stackexchange.com`[14]. While it still hosts many interesting questions and answers, activity has slowed considerably (for example, the backlog of unanswered questions has grown). With the rise of AI chatbots, its role is likely to diminish further.

As a community, we need to think seriously about how we share news, celebrate achievements, coordinate initiatives, and, more broadly, how we communicate with one another. The current situation is far from ideal. The fragmented nature of today's microblogging platforms makes it difficult to distinguish genuine news (e.g., research breakthroughs or funding opportunities) from a constant stream of useless jibber jabber. Just as the TCS aggregator once brought coherence to a scattered blogosphere in the 2010s, we now need to develop new tools in 2025 that can restore clarity and strengthen how our research community connects and collaborates.

Thank you for this nice interview!

[14] `https://cstheory.stackexchange.com`

The Machine Learning Column

BY

Pablo Barceló

Pontificia Universidad Católica de Chile
Avda. Vicuña Mackenna 4860
Santiago, Chile
pbarcelo@uc.cl

AND

David Saulpic

Institut de Recherche en Informatique Fondamentale (IRIF)
CNRS & Université Paris Cité
8 Place Aurélie Nemour
75013 Paris, France
david.saulpic@irif.fr

THE LANDSCAPE OF TCS FOR ML

Abstract

We introduce here a new column, TCS for ML. The idealistic goal of this column will be to examine what special directions, what special insight Theoretical Computer Science can bring to Machine Learning. In this initial edition, we attempt to examine how our TCS fields currently contribute to ML, from all side. This introduction will serve as the basis for deeper investigation in next columns, where we will try to highlight new theoretical questions arising from ML and its massive use.

1 Introduction

The past 10 years have seen the rise of Machine Learning (ML), which is now a recognized and dominant field - in terms of funding, public exposure, research progress, and applications. As many aspects of ML are directly related to computation, it would be natural that Theoretical Computer Science (TCS) contributes to this success-story. However, this is a relatively new field, growing extremely fast: it is not completely structured, and it is not easy to enter the field, nor to follow the different contributions. This is perhaps why editors of the *Bulletin* asked us to write a column on "TCS for ML".

As both TCS and ML are vast, fast-moving fields, we do not have a definitive picture of what this means exactly. Hence, we have asked -— more or less successfully —- experts from various TCS subfields to provide their own hindsight, in order to sketch an initial big picture. In this initial column, we present those expert perspectives along with our own views, and in the coming months we will dive into more specific aspects of TCS and ML, providing the *Bulletin* with interviews of researchers discussing their work. Our far-reaching objective will be to provide a global picture of the research directions followed by the TCS community to contribute to ML ; and, ideally, to identify open and interesting directions.

As a disclaimer, while we aim to capture a broad and honest picture of the landscape, we are aware that our perspective is limited and shaped by our own interests. We therefore welcome any comments or criticisms that could help improve this review.

TCS and ML: definitions. We considered as TCS the various communities represented by EATCS and showcased at ICALP. This spans a wide spectrum of areas—sometimes overlapping, sometimes far apart—but all connected by SIGACT's definition of TCS as "the formal analysis of efficient computation and computational processes".[1] Guided by this perspective, we focus here on works that present strong theoretical results or apply theoretical tools to rigorously analyze the capabilities and limitations of ML models.

ML, as we know, is an expansive discipline at the intersection of computer science, statistics, and numerous applied fields, dedicated to designing algorithms and representations that can infer patterns or decision rules from data. Beyond its practical successes, ML is a source of theoretical challenges for TCS. First, it calls to revisit fundamental questions about computation, efficiency, and representation. Second, it offers opportunities for interaction with traditional TCS communities, such as algorithms, logic, formal verification, databases, and knowledge representation, where decades of research provide principled ways to encode knowledge, enforce constraints, and explain or verify model behavior.

In this column we have organized our review by traditional TCS subfields (Theory of Computation, Ethical ML, Algorithms, Logic/ Formal Languages/ Discrete Structures, Foundations of Data Management, Formal Verification, and Knowledge Representation), highlighting foundational works and current research directions relating to ML in each area. But before diving into those subfields, it is useful to step back and group the interactions between TCS and ML more conceptually, following the SIGACT definition above. We identify three complementary roles:

- Formal analysis of ML models and their capabilities—understanding what learning architectures (e.g., Transformers, GNNs) can and cannot compute, their sample complexity, and their expressiveness.
- Formal analysis of ML outcomes — using TCS tools to verify, explain, and guarantee properties such as fairness, privacy, or logical consistency of predictions.
- Support for efficient computation—drawing on TCS techniques in algorithms, data management, and KR to scale ML pipelines, optimize primitives, and integrate explicit knowledge with data-driven learning.

These perspectives are not exhaustive, but they capture recurring themes across the subfields we survey. They also remind us that while ML often pushes TCS to confront new challenges, TCS offers a deep well of methods for making ML more principled, reliable, and interpretable.

[1] `https://www.sigact.org/`

Understanding the power and limits of neural networks. One of the central themes of TCS is understanding computation—specifically, what can be computed and with which model of computation. These questions can be directly reformulated in the context of ML: what can be learned, and what can a specific model learn? From the beginning of the field, several research areas have emerged to provide different perspectives on computation. As we will see, they all offer viewpoints that can explain—or at least begin to explain—the power and limits of neural networks.

First, Learning Theory emerged alongside ML precisely to build a theory of learnability. Here, "learning" means that an algorithm is given some labeled data as input and must infer the labels of new data points. The labels are functions of the data, and the goal is to understand which functions are "easy" or "hard" to learn. Difficulty is measured primarily in terms of the amount of input data necessary for inference or the computational complexity required. This general theory therefore aims to broadly characterize which functions can be learned, defining properties of functions (or dimensions) that measure the difficulty of learning them. One specific learning task that has recently been investigated—shedding TCS light on a broader debate—is the task of learning languages, which we will cover in more detail later.

Logic and formal languages provide a different view: instead of studying functions broadly, their tools and theory allow us to focus on particular network architectures. A fixed architecture, or a family of architectures, restricts the computations possible for a network; these fields examine the properties that can be recognized by specific architectures. A similar perspective comes from complexity theory, which focuses on specific learning tasks and derives lower bounds on the size of a network required to compute those tasks. Recent examples highlight the difficulty for LLMs to "compose" functions—a task they are neither explicitly designed nor trained for, but which they can sometimes handle surprisingly well and, at other times, fail at quite spectacularly.

In both cases, substantial modeling effort is required before answering such questions: one must model which class of functions networks attempt to learn or provide a formal model for the network architecture.

Formal analysis of ML's results. One of the major critiques of ML systems concerns the uncertain nature of their outcomes: given a new data point—different from those observed during the learning process—how can we be sure that the model will remain satisfactory? Here, formal guarantees are required regarding the model's ability to generalize. This is where TCS steps in, as proving guarantees is our daily bread.

These guarantees can sometimes be automatically established via formal veri-

fication: given a model, one may wish to automatically verify that its output satisfies certain desirable properties. Hardness results — even for simple models and properties — make this task quite challenging, which is why several relaxations have been investigated.

Another approach is to prove such guarantees manually, by establishing theorems about specific properties of the models. Beyond the correctness of predictions—which is the focus of Learning Theory—several ethical properties are being actively studied. A growing field of ethical AI and ML formally defines these properties and designs learning algorithms that enforce them, so that the resulting model can be called "fair" or "private," for instance. Here again, modeling plays a central role: mathematically defining "fairness" is not an easy task. Many definitions exist, are not necessarily compatible with each other, and may not provide practical leverage for algorithm design. As we will see in the dedicated section, this modeling can be guided by TCS insights.

Design of Efficient ML Algorithms. The aspect of TCS perhaps closest to practical applications is the design of efficient algorithms and database techniques that enable fast and memory-efficient computation. Since ML is inherently tied to big data, it requires algorithms and data-management tools that can scale: our field provides such tools, backed by solid theory and formal guarantees.

The communication between TCS and ML goes both ways. Ideas from TCS have been incorporated into big-data pipelines — such as streaming and sketching techniques in algorithms or in-database learning frameworks and functional aggregate queries (FAQs) from database theory, which allow training models directly within relational systems without expensive data extraction. Conversely, the ML industry continues to raise new computational questions: identifying and optimizing the critical tools or subroutines used in practice. For example, efficient gradient computation over relational data, fast query-based feature generation, and probabilistic database techniques for uncertainty handling are all areas where database theory informs ML workflows.

Finding which subroutines are ripe for algorithmic improvement is itself an art—one that requires expertise in both ML and TCS. This ongoing exchange not only optimizes existing pipelines but also opens opportunities for theoretical insights to drive entirely new approaches to scalable ML.

We will now dive, subfield by subfield, into more details. Again, we are not expert in all of those: we have tried to ask better-placed persons their opinion, and have probably overlooked some directions and over-emphasized some others. Nonetheless, we still believe those partial thoughts are an interesting first step, and will warmly welcome any feedback.

2 Theory of Computation

The interplay between theory of computation and ML has deep roots, from VC theory in statistics to PAC learning in theoretical computer science. Today, rapid advances in ML far outpace rigorous theoretical understanding, creating both a challenge and an opportunity. Theory is needed to explain the successes and limitations of current methods and to distinguish fundamental phenomena from transient trends. While ML theory can be messy—often involving toy models or impractical algorithms, it is precisely this principled, analytical perspective that ensures AI's development is grounded in solid foundations.

- **Learning Theory:** Learning theory is a well-established field whose primary goal is to understand which functions are *learnable* and which are not. Several definitions of learnability co-exist, most prominently PAC learning [113], and efficiency may be measured in terms of the number of samples or computational time. In other words, this field studies the theory of supervised machine learning and is therefore at the heart of our topic.

  To analyze learnability, no assumptions are made about the structure of the learning or prediction algorithms, so the results apply very broadly. The trade-off is that there are limitations on the types of problems studied. For instance, PAC learning traditionally assumes that input examples follow a distribution and that the test set *also* follows this distribution. In addition, positive results often require strong assumptions on this distribution, e.g., that it is Gaussian or a product distribution. Distribution-free learning is generally hard (see, for instance, [71, 30, 72]), and some recent works attempt to lift results from specific distributions to general ones [28, 29]. To address limitations regarding distributions, a theory of robustness has emerged (see, e.g., [88, 50, 80, 93]) to characterize what sorts of noise or adversarial conditions can be handled.

  Beyond fundamentally understanding what is learnable, concepts from learning theory are also used to evaluate algorithms in practice. For example, [54, 36] show that diffusion models can learn mixtures of Gaussians, extending a recent line of work on these models, and [116] evaluate the ability of transformers to learn k-fold composition functions.

  Recent work has studied "hallucinations" in LLMs [66, 65]. These results show that even an ideal, perfectly calibrated language model cannot entirely avoid hallucinations: it must assign nonzero probability to facts not seen during training, leading inevitably to some false outputs. Conceptually, these findings echo *No Free Lunch* theorems, as they highlight fundamental trade-offs between generality and accuracy.

- **Language generation in the limit:** The success of Large Language Models (LLMs) has intensified the need to understand the principles underlying their effectiveness and to identify the mathematical components that support it. Recent research has begun addressing these questions by studying the design of generative algorithms for language [78]. This line of work builds on a rich set of theoretical tools, most notably the *Gold–Angluin framework* for language identification, originally developed to analyze the learnability of formal languages. More formally speaking, imagine an adversary who keeps listing strings from some unknown language L, which we only know belongs to a possibly infinite list of candidate languages. A computational agent tries to learn how to generate strings from L. We say the agent *generates from L in the limit* if, after seeing enough of the adversary's examples, it can start producing new strings that (1) always belong to L, and (2) have never been shown before. The main result in this area shows that there exists such an agent for any countable list of possible languages. Subsequent work has examined the following key question: is it possible to design language generation algorithms that are not only *valid*—producing well-formed strings in the target language L—but also *broad*—producing a diverse set of strings from L [68, 100, 77, 67]. LLMs inherently face this validity–breadth trade-off: they must balance the need for accuracy and coherence (avoiding *hallucination*) with the need for variety and coverage (avoiding *mode collapse*).

- **Complexity, and Lower bounds for ML architectures:** Transformers are the ML architecture that lies at the core of LLMs. Transformers use a mechanism known as *self-attention* to weigh the importance of different parts of an input sequence. They also use *chain-of-thought* (CoT) reasoning as a way to break down a complex problem into a series of intermediate steps which the model explicitly generates before arriving at a final answer. By providing these intermediate steps, CoT effectively gives the Transformer model a form of "scratchpad memory" or "workspace" to perform computations.

  A key question in understanding the computational capabilities of Transformers is how many CoT steps they require to carry out function composition, a core operation in both symbolic reasoning and natural language understanding. Recent theoretical work has tackled this from complementary perspectives, depending on whether one considers *hard* or *soft* attention mechanisms. Hard attention makes a discrete, non-differentiable choice, focusing on a small, selected subset of input elements. In contrast, soft attention computes a weighted average of all elements in the input sequence. The first approach uses communication complexity to prove that single-

layer Transformers with soft attention cannot perform function composition when the function domains are large, and extend this to show that iterated composition requires a number of CoT steps that grows with the domain size [101]. The second approach uses the notion of *Ehrenfeucht-Haussler rank* of a Boolean function and the minimum number of Chain of Thought (CoT) steps required by a single-layer Transformer with hard attention to compute it, demonstrating that l-fold function composition requires exactly l CoT steps [18]. The third approach provides the first unconditional lower bounds for multi-layer Transformers with soft attention. This work introduces a new *multi-party autoregressive* communication model and uses it to obtain strong lower bounds for the number of CoT steps required to compute the composition of L functions [35].

3 Ethical ML

Another direction in which a principled and analytical perspective is required is that of ethical ML, in which we include urgent issues of AI safety, ethics, and trustworthiness. For these, formal guarantees are required—and sometimes even enforced by law. Theoretical insights not only clarify what is possible in these directions but may also help in modeling and proving these formal guarantees. The wonderful book by Kearns and Roth [70] dives deep into the topic; we will only briefly cover some of its aspects related to privacy and fairness.

In this field, TCS provides technical tools and analyses, but a technical solution is not enough. While this is beyond our survey, we refer, for instance, to the works by the FAccT community (organized around the ACM conference on Fairness, Accountability and Transparency) or STS scholars (e.g., [92] and the journal Big Data and Society), which are key to studying ethical questions. We emphasize that, in our view, TCS only provides a helping hand, not a complete answer.

- **Privacy:** Certain ML applications rely on sensitive training data, which must be carefully protected during training. This protection is required by the European GDPR; privacy concerns are therefore brought into legal debates, in which TCS arguments are used to argue for the privacy of some algorithms. Over the past decade, differential privacy has emerged as a foundational tool to address this challenge. A major line of research focuses on designing differentially private algorithms for empirical risk minimization (ERM), particularly when the loss function is Lipschitz or strongly convex [23]. This work introduced several algorithms and established tight theoretical error bounds, providing optimal risk guarantees. Building on these ideas, differential privacy has been extended to deep learning. Differentially Private Stochastic Gradient Descent (DP-SGD) offers a principled

framework for algorithm design while carefully controlling privacy budgets [1], whereas Private Aggregation of Teacher Ensembles (PATE) leverages the teacher-student paradigm to achieve strong utility alongside differential privacy guarantees [98]. More recent research has focused on efficient algorithms for differential learning and on handling non-convex loss functions [21, 22]. Other work investigates how to add just enough noise to preserve privacy without compromising accuracy, introducing mechanisms that carefully balance privacy and utility [55]. On the other hand, what differential privacy actually means for practical privacy is somewhat unclear—in particular because of parameter choices [48]—and some works focus on attacks in order to better understand the limits of this model for privacy [14].

- **Fairness:** The dramatic analysis of the COMPAS system by the journalists of ProPublica [64] has shown that ML predictions are prone to massive bias—sometimes replicating bias from the training data, sometimes inherent to the task. In order to measure this bias, several notions of fairness coexist, with two main categories: individual fairness tries to measure how much predictions differ for individuals that are similar [46], while group fairness measures whether the predictions are fair between different population groups [20]. While the Statistics community is designing fair procedures—i.e., that have small bias according to one of these measures—for problems such as regression or classification, with the objective of bounding risk or learning rate under some statistical assumption, TCS naturally contributed to designing fair and efficient algorithms and to designing a toolbox for fair algorithms [38, 47]; but also to understanding the relations between different fairness notions and to formalizing impossibility results [79, 105, 52].

Here, our survey has some shortcomings: the story for Ethical AI is not limited to these two topics, but we did not manage to cover more. In particular, it seems there are exciting developments around the notion of trustworthy ML, that we would have liked to be able to discuss.

We note that, in all cases, the solutions proposed by computer scientists are not sufficient as standalone tools. Quantifying privacy and fairness is not easy, and perhaps not even possible; and the number of different fairness notions may blur the debate, as mentioned on Wikipedia: "the different and sometimes competing notions of fairness left little room for clarity on when one notion of fairness may be preferable to another" [117]. However, these tools may be helpful when used in combination with other sociotechnical approaches.

4 Algorithms

One of the goals of algorithm design is to invent algorithms that make better use of memory and time. Quite naturally, some of the algorithmic techniques developed in the past now find applications in ML and diffuse ideas into that domain; conversely, new problems arising in ML are extensively studied and optimized by the algorithmic community.

- **Algorithms for data-intensive tasks:** Prominent examples of past algorithms finding application today are related to the "big data" side of ML, with huge memory and efficiency constraints. To address this, the study of metric embeddings (e.g., embedding into structured trees [49]) and dimensionality reduction (so-called Johnson-Lindenstrauss [83] or Locality-Sensitive Hashing [8]) helps simplify the input space and reduce the impact of the curse of dimensionality. The streaming model, with its emphasis on memory, led to the invention of sketches that allow data compression while still enabling computation of certain statistics or information [6]. Other extensively studied problems include basic unsupervised learning algorithms, namely clustering and regression. These have been studied through different formulations (e.g., metric-based with k-means [11, 40] or graph-based with sparsest cut [10]) and algorithmic settings, with emphasis on time efficiency [45] or memory efficiency [41, 42].

- **Computation of ML primitives:** More recently, part of the community has focused on efficiently computing primitives often used in contemporary ML. Two examples are optimal transport and its variants, investigated through an approximation-algorithm perspective [27], computational geometry tools [3], or with an emphasis on linear-time complexity [13]; and kernel-density estimation, with memory-efficient [104] or fast algorithms [34, 33]. Finding other problems relevant to ML and data analysis where the algorithmic toolbox may yield substantial improvements is not easy, as it requires expertise in both domains. Nevertheless, it remains one of the major questions for the subfield of algorithms that focuses on improving practical algorithms.

5 Logic, Formal Languages, and Discrete Structures

Studying the capabilities and limitations of ML architectures involves examining the properties they can express and the structures they can distinguish. This line of research is deeply rooted in logic and formal languages, which provides a framework for analyzing the properties of inputs that an architecture can represent. It

also draws on discrete algorithms as a tool for understanding how these architectures differentiate between various inputs. Over the past decade, a surge of studies has focused on understanding the capabilities of two fundamental ML architectures: *Transformers*, which we have seen before, and *Graph Neural Networks* (GNNs), which learn by propagating information across the edges of a graph.

- **Transformers:** The ability of Transformers to process sequences can be formally analyzed by drawing on the rich traditions of logic and formal language theory. Since Transformer inputs can be viewed as strings over a finite alphabet, researchers can investigate which classes of languages these models are capable of recognizing. The resulting characterization, however, critically depends on the architectural features under consideration. The languages recognized by Transformers with hard attention have been shown to be closely tied to those expressible in First-Order Logic (FO) and its extensions with counting [17, 120]. In turn, while equipped with certain expressive features, soft attention Transformers can be shown to recognize all FO-definable languages, and even extensions [119, 121]. However, when restricting the model to more practical architectures used in real-world applications, their ability to recognize the entire class of FO-definable languages becomes limited [84].

  Interesting results have also been obtained regarding the expressive limitations of certain Transformer models. For instance, Transformers equipped with a restricted form of hard attention, known as *unique* hard attention, are limited to recognizing languages within the complexity class AC^0 [61]. This limitation has significant consequences. Specifically, models with unique hard attention are provably incapable of recognizing languages that fall outside this class. A prime example is the Parity language, which determines whether a binary string has an even or odd number of ones. Because the Parity language is not in AC^0, these restricted Transformers are unable to solve this seemingly simple task [60].

  Yet another line of work has analyzed the ability to recognize languages for Transformers extended with CoT reasoning. This capability has been shown to boost the model's expressive power; in fact, several models of Transformers with CoT and hard attention have been shown to be Turing-complete, meaning that they are capable of recognizing all decidable languages [102, 90, 51].

  An excellent survey on different aspects of this topic has recently been published [109].

- **Graph Neural Networks:** The foundational work in understanding the expressive power of Graph Neural Networks (GNNs) began with the semi-

nal result that their ability to distinguish graphs coincides with that of the Weisfeiler-Leman (WL) test [118, 96]. The WL test is a widely studied polynomial-time heuristic for checking graph isomorphism. This initial result opened a large avenue of research exploring the relationship between different flavors of GNNs and various higher-order versions of the WL test. The WL connection has also allowed developing a clear understanding of which substructures of graphs can be detected, or counted, with GNNs [37, 15, 82]. Finally, the WL test has also been related to the VC dimension of GNNs, thus providing a critical benchmark for graph representation learning [95].

A parallel line of work has focused on understanding which properties of graphs can be recognized or defined by GNNs. Early results in this area provided a logical characterization of the class of FO definable graph properties that can be captured by GNNs, which coincides with the expressive power of *graded modal logic* (GML) [16]. This characterization requires GNNs to be equipped with a specific class of piecewise linear activation functions, which are expressive enough to simulate counting up to fixed bounds. On the other hand, it has been shown that if the activation functions are restricted to be linear or polynomial, the expressive power of GNNs drops significantly, and they can no longer capture even basic GML-definable properties [73]. More recently, the full expressive power of GNNs—beyond FO-definable properties—has been characterized for a restricted but expressive class of activation functions (known as *eventually constant* functions) in terms of an extension of GML enriched with Presburger arithmetic constraints [26]. In contrast, when activation functions are not eventually constant, the landscape becomes less well understood. In this more general setting, only upper and lower bounds are known for the expressive power of GNNs in logical terms [57]. Recently, the expressive power of recurrent GNNs—that is, GNN architectures where the same message-passing layer is applied repeatedly over multiple rounds—has also begun to be systematically investigated [4, 103, 32].

Several of these topics are presented in depth in a recent survey [56].

6 Foundations of Data Management

Database theory offers rigorous frameworks for organizing, querying, and managing large-scale structured data, which forms the backbone of many ML applications. By integrating principles from database theory, ML systems can achieve more efficient data access, better feature extraction, and more interpretable models

grounded in logical formalisms. This synergy enhances the scalability and reliability of ML algorithms and opens new avenues for understanding model behavior through formal queries. Below, we highlight several key ideas and approaches proposed by the database theory community that contribute to advancing these goals.

- **In-database learning:** A framework that enables the training of a wide array of statistical models directly inside a relational database has been proposed [74]. This is achieved by representing features as sparse tensors and expressing gradient computations as *functional aggregate queries* [75], a powerful and general framework for expressing a wide range of database queries that involve aggregation. The proposed method leverages the inherent join structure and functional dependencies of the database to significantly reduce per-iteration costs to a sub-linear level. By eliminating the need to extract data, the system achieves substantial speedups compared to conventional ML workflows, highlighting the power of applying database-theoretic optimization to modern ML pipelines.

- **Logic-based feature generation:** This line of research investigates how data management techniques can be exploited to learn sophisticated ML features specified by logical queries. A concrete example is the use of the well-known class of *unions of conjunctive queries* (UCQs)—which coincides with the existential-positive fragment of first-order logic—to separate classification data via a linear model whose features are given by such UCQs [76]. Learning such query-defined features is particularly important in settings with relational or graph-structured data, where predictive signals often stem from patterns spanning multiple entities and relationships. Their formal semantics also make them interpretable and amenable to theoretical analysis of expressiveness and generalization. This setting has inspired renewed interest in the long-studied problem of learning database queries from examples, leading to recent advances in both theoretical frameworks and algorithmic techniques [111, 112].

- **Querying ML models:** One of the central concerns in the foundations of data management is how to efficiently and effectively extract answers to logical queries over a dataset. A notable development in recent years is the realization that ML models can themselves be viewed as datasets—albeit implicit and compact ones—and thus can be queried to extract meaningful information about their input-output behavior [9, 58]. For example, querying an ML model can help generate explanations for its predictions, or verify whether it satisfies certain desirable properties. The computational com-

plexity of extracting such explanations over different classes of ML models has become a recurring topic in recent literature [19, 44, 97].

- **Dealing with uncertain data:** Modern ML models are powered by vast amounts of noisy data, which means that the information extracted from them often comes with varying degrees of confidence. A natural question, then, is how to measure such confidence in a principled way. Database theory has developed a foundational framework for addressing this challenge through the notion of *probabilistic data* [110]. At its core, a probabilistic database assigns a probability to each tuple, representing the likelihood that the tuple is present. These probabilities can be subject to constraints or correlations, enabling the modeling of rich and complex uncertainty scenarios. A central computational task in this setting is to determine the probability that a given query is true across all possible "worlds" encoded by the probabilistic data—an answer that directly quantifies our confidence in the query's validity given the noisy dataset. This framework has led to major theoretical advances, most notably a landmark dichotomy theorem for unions of conjunctive queries (UCQs): some UCQs can be evaluated in polynomial time with respect to data complexity, while others are #P-hard, revealing a sharp boundary between tractable and intractable cases [43]. Query evaluation over probabilistic databases is closely linked to another central problem in AI: knowledge compilation, which studies how to efficiently perform reasoning tasks—such as satisfiability—over propositional knowledge bases. This field has produced an impressive body of results, particularly in the design and use of classes of tractable Boolean circuits that enable efficient solutions to problems arising in the management and analysis of probabilistic data (see [7] for a survey on this topic).

7 Formal Verification

The theory of verification—concerned with rigorously proving that systems behave as intended—offers powerful tools for increasing the reliability and trustworthiness of ML models. As ML systems are deployed in safety-critical domains such as healthcare, transportation, and finance, ensuring their correctness, robustness, and fairness becomes essential. Verification theory contributes formal methods to specify desired properties of models, systematically check them against all possible inputs, and identify counterexamples when guarantees fail. This integration not only strengthens the safety and accountability of ML but also deepens our theoretical understanding of its behavior, enabling the design of models that are both powerful and provably reliable. Next, we highlight several key

ideas that have emerged at the intersection of formal verification and ML over the past decade.

- **Classical verification:** The verification of ML models has mostly centered around (deep) neural networks [5]. Languages for specifying desirable properties of the input–output behavior of neural networks are often based on logical formalisms such as Hoare logic or linear real arithmetic, allowing one to express statements of the form: whenever an input x to a network N satisfies certain constraints, the corresponding output $y = N(x)$ also satisfies certain constraints. A closely related *reachability* problem asks whether there exists an input in a given set that leads to an output in another given set. From a complexity-theoretic standpoint, exact verification of even simple properties is computationally intractable in the worst case. For feed-forward ReLU networks, the reachability problem is NP-complete [69], and this hardness persists even for shallow architectures with a single hidden layer [108]. Going beyond reachability, robustness verification—deciding whether all points in a perturbation set around a given input yield outputs in a safe region—has also been shown NP-complete [115]. Other properties can reach higher complexity: for instance, deciding surjectivity of ReLU networks can be Σ_2^{P}-complete [53]. These results delineate precise tractability frontiers, showing that certain restrictions—such as monotonic activations, fixed topology, or bounded input dimension—are necessary for polynomial-time verification. Together, these hardness classifications form the theoretical foundation for why approximate or restricted verification methods are often unavoidable in practice [69, 62].

- **Advanced verification:** Recently, research has increasingly focused on more sophisticated verification models and tasks. One prominent line of work investigates the decidability of verifying neural networks with *smooth* activation functions, such as sigmoid or tanh. It has been shown that this problem is equivalent to the decidability of the FO theory of the reals extended with the exponential function—a long-standing open problem in model theory known as *Tarski's exponential function problem* [63]. Another area of study is #NN-Verification, the counting analogue of standard neural network verification. The objective here is to determine the number of inputs that violate a given safety property. This problem is #P-complete, motivating the design of exact algorithms and randomized approximation techniques [89]. Building on this, a generalized probabilistic verification framework has been proposed, aiming to compute bounds on the probability of property violations under arbitrary input distributions. This probabilistic variant is also #P-hard, and a branch-and-bound approach has been

developed to address it, with formal proofs guaranteeing both soundness and completeness [31].

8 Knowledge Representation

Theoretical research in knowledge representation (KR) is essential for the advancement of ML because it provides a complementary perspective that helps build AI systems that are both powerful and interpretable. KR has been part of AI since its inception, giving the community deep insights into the strengths and limitations of purely data-driven methods, as well as a rich set of tools for addressing key challenges in ML. While the limitations of ML—such as difficulty in reasoning over structured knowledge or capturing complex relational dependencies—are well-known, the field of KR offers concrete technical approaches to address them. Properly integrated, KR can enhance ML by enabling the encoding of explicit domain knowledge, enforcing constraints, explaining predictions in human-understandable terms, and supporting reasoning over learned models. The subsequent sections illustrate some of these technical solutions, including statistical relational learning, probabilistic and differentiable logic, and knowledge graph embeddings, which collectively demonstrate how KR can complement and extend standard ML techniques.

- **Statistical relational learning:** Over the past decade, statistical relational learning (SRL) has made major theoretical advances that sharpen our understanding of the trade-offs between expressivity and tractability, while strengthening the links between logic, probability, and ML. SRL is important for AI because many real-world domains are both relational (entities and their relationships matter) and uncertain (data is noisy, incomplete, or stochastic). SRL provides the formal machinery to jointly model these properties, enabling AI systems to learn from data while reasoning with structured, semantically rich knowledge. A key unifying development has been the view of many SRL formalisms through the lens of *probabilistic circuits* (PCs), where structural constraints guarantee polynomial-time inference and closedness under key transformations, thus reframing inference complexity in relational models in circuit-theoretic terms [39]. Complementing this, there has been work on *lifted inference*, which is the task of doing probabilistic inference over a model without grounding all the variables in it. Lifted inference can be rephrased as the *model counting* problem: Given a FO formula ϕ and an integer n, compute the number of models of ϕ of size n. A precise characterization of which FO formulas admit tractable model counting has been obtained [24]. In addition, several extensions of this tractable fragment have been found by applying combinatorial

techniques [81, 114, 86]. SRL has also expanded beyond purely discrete domains through the theory of *model integration* [25, 94].

- **Probabilistic and differentiable logic:** Probabilistic and differentiable logic frameworks have become crucial for enabling reasoning under uncertainty while still supporting gradient-based optimization. This allows symbolic rules to be learned jointly with neural components. Over the last decade, several theoretical advances have shaped this area. For instance, differentiable logic systems such as DeepProbLog [87] have introduced end-to-end differentiable reasoning frameworks that extend classical logical semantics with gradient-based learning, together with analyses of complexity and expressivity [85]. This has been complemented by frameworks like *Logic Tensor Networks* (LTNs), which introduced a many-valued, end-to-end differentiable FO logic that unifies learning and reasoning in a single formalism [12]. Further advances include the development of *Logical Neural Networks* (LNNs)—a differentiable neuro-symbolic model where each neuron corresponds to a component of a logical formula, enabling resilience to logical contradictions and support for open-world semantics via real-valued truth bounds [107].

- **Knowledge graph embeddings:** Knowledge graph embeddings (KGEs) have become a central research topic at the intersection of knowledge representation and ML because they provide a scalable way to bridge symbolic relational knowledge with continuous vector-space learning. KGEs operate by mapping entities and relations into geometric spaces where reasoning tasks—such as link prediction, query answering, and ontology completion—can be carried out efficiently. Their design raises fundamental theoretical questions: what kinds of logical patterns, rules, and graphs can different embedding families represent, and what are their inherent limitations? In recent years, substantial theoretical progress has been made in addressing these questions. For example, a geometry-based framework was proposed demonstrating that widely used KGE models fail to capture certain existential rule patterns, and introducing embeddings based on convex regions that can faithfully encode a particular class of existential rules while ensuring both logical consistency and deductive closure [59]. Another important contribution is BoxE, a fully expressive embedding model in which entities are points and relations are boxes in latent space, supporting higher-arity relations and principled incorporation of logical rules [2]. Its temporal extension, BoxTE, preserves full expressivity in temporal knowledge graphs while exhibiting strong inductive generalization [91]. More recently, research has focused on enhancing both the expressivity and interpretability

of KGEs. For instance, ExpressivE represents entities as points and relations as hyper-parallelograms in a 2D Euclidean space [99]. This geometric design enables the model to capture a rich set of inference patterns, including composition and hierarchy, while providing an intuitive geometric interpretation of these patterns. An interesting survey on different aspects of lind prediction and query answering over incomplete knowledge graphs has recently been published [106].

9 Final Remarks

We thought of this initial column as a wide review on how different areas of TCS — both the questions they pose and the tools they develop — have contributed to recent advances in ML, or how they could contribute. And on the flip side, whether developments in ML are starting to shape or inspire work in those fields in any meaningful way.

Those questions appear ubiquitous in TCS: the future publications of the column will dive deeper into topics we glanced over. It will consist of interviews, reviews of specific topics or position papers: we would appreciate any suggestion, thoughts or feedback you might have!

Acknowledgements

We are extremely grateful to Cristobal Guzman, Javier Esparza, Wim Martens, Magdalena Ortiz, Santosh Vempala, and David Woodruff for their invaluable comments and ideas during the preparation of this article.

References

[1] Martín Abadi, Andy Chu, Ian J. Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *CCS*, pages 308–318, 2016.

[2] Ralph Abboud, Ismail Ceylan, Thomas Lukasiewicz, and Tommaso Salvatori. Boxe: A box embedding model for knowledge base completion. In *NeurIPS*, pages 9649–9661, 2020.

[3] Pankaj K. Agarwal, Hsien-Chih Chang, Sharath Raghvendra, and Allen Xiao. Deterministic, near-linear ϵ-approximation algorithm for geometric bipartite matching. In *STOC*, pages 1052–1065. ACM.

[4] Veeti Ahvonen, Damian Heiman, Antti Kuusisto, and Carsten Lutz. Logical characterizations of recurrent graph neural networks with reals and floats. In *NeurIPS*, 2024.

[5] Aws Albarghouthi. *Introduction to Neural Network Verification*. verifieddeeplearning.com, 2021. `http://verifieddeeplearning.com`.

[6] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29, 1996.

[7] Antoine Amarilli and Florent Capelli. Tractable circuits in database theory. *SIGMOD Rec.*, 53(2):6–20, 2024.

[8] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468. IEEE Computer Society, 2006.

[9] Marcelo Arenas, Daniel Báez, Pablo Barceló, Jorge Pérez, and Bernardo Subercaseaux. Foundations of symbolic languages for model interpretability. In *NeurIPS*, pages 11690–11701, 2021.

[10] Sanjeev Arora, Elad Hazan, and Satyen Kale. $o(\sqrt{(}\log(n))$ approximation to sparsest cut in $\tilde{o}(n^2)$ time. *SIAM J. Comput.*, 39(5):1748–1771, 2010.

[11] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 1027–1035. SIAM, 2007.

[12] Samy Badreddine, Artur S. d'Avila Garcez, Luciano Serafini, and Michael Spranger. Logic tensor networks. *Artif. Intell.*, 303:103649, 2022.

[13] Ainesh Bakshi, Piotr Indyk, Rajesh Jayaram, Sandeep Silwal, and Erik Waingarten. Near-linear time algorithm for the chamfer distance. In *NeurIPS*, 2023.

[14] Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. In *sp*, pages 1138–1156. IEEE, 2022.

[15] Pablo Barceló, Floris Geerts, Juan L. Reutter, and Maksimilian Ryschkov. Graph neural networks with local graph parameters. In *NeurIPS*, pages 25280–25293, 2021.

[16] Pablo Barceló, Egor V. Kostylev, Mikaël Monet, Jorge Pérez, Juan L. Reutter, and Juan Pablo Silva. The logical expressiveness of graph neural networks. In *ICLR*, 2020.

[17] Pablo Barceló, Alexander Kozachinskiy, Anthony Widjaja Lin, and Vladimir V. Podolskii. Logical languages accepted by transformer encoders with hard attention. In *ICLR*, 2024.

[18] Pablo Barceló, Alexander Kozachinskiy, and Tomasz Steifer. Ehrenfeucht-haussler rank and chain of thought. *ICML*, 2025.

[19] Pablo Barceló, Mikaël Monet, Jorge Pérez, and Bernardo Subercaseaux. Model interpretability through the lens of computational complexity. In *NeurIPS*, 2020.

[20] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness and machine learning. *Recommender systems handbook*, 1:453–459, 2020.

[21] Raef Bassily, Cristóbal Guzmán, and Michael Menart. Differentially private stochastic optimization: New results in convex and non-convex settings. In *NeurIPS*, pages 9317–9329, 2021.

[22] Raef Bassily, Cristóbal Guzmán, and Anupama Nandi. Non-euclidean differentially private stochastic convex optimization. In *COLT*, volume 134, pages 474–499, 2021.

[23] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, pages 464–473, 2014.

[24] Paul Beame, Guy Van den Broeck, Eric Gribkoff, and Dan Suciu. Symmetric weighted first-order model counting. In *PODS*, pages 313–328, 2015.

[25] Vaishak Belle, Andrea Passerini, and Guy Van den Broeck. Probabilistic inference in hybrid domains by weighted model integration. In *IJCAI*, pages 2770–2776. AAAI Press, 2015.

[26] Michael Benedikt, Chia-Hsuan Lu, Boris Motik, and Tony Tan. Decidability of graph neural networks via logical characterizations. *CoRR*, abs/2404.18151, 2024.

[27] Lorenzo Beretta and Aviad Rubinstein. Approximate earth mover's distance in truly-subquadratic time. In *STOC*, pages 47–58. ACM, 2024.

[28] Guy Blanc, Jane Lange, Ali Malik, and Li-Yang Tan. Lifting uniform learners via distributional decomposition. In *STOC*. ACM, 2023.

[29] Guy Blanc, Jane Lange, Carmen Strassle, and Li-Yang Tan. A distributional-lifting theorem for PAC learning. In *COLT*, volume 291, pages 375–379. PMLR, 2025.

[30] Avrim Blum and Ronald L. Rivest. Training a 3-node neural network is np-complete. *Neural Networks*, 5(1):117–127, 1992.

[31] David Boetius, Stefan Leue, and Tobias Sutter. Probabilistic verification of neural networks using branch and bound. *CoRR*, abs/2405.17556, 2024.

[32] Jeroen Bollen, Jan Van den Bussche, Stijn Vansummeren, and Jonni Virtema. Halting recurrent gnns and the graded μ-calculus. *CoRR*, abs/2505.11050, 2025.

[33] Moses Charikar, Michael Kapralov, Navid Nouri, and Paris Siminelakis. Kernel density estimation through density constrained near neighbor search. In *FOCS*, pages 172–183. IEEE, 2020.

[34] Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *FOCS*, pages 1032–1043, 2017.

[35] Lijie Chen, Binghui Peng, and Hongxun Wu. Theoretical limitations of multi-layer transformer. *FOCS*, 2025.

[36] Sitan Chen, Vasilis Kontonis, and Kulin Shah. Learning general gaussian mixtures with efficient score matching. In *COLT*, volume 291, pages 1029–1090. PMLR, 2025.

[37] Zhengdao Chen, Lei Chen, Soledad Villar, and Joan Bruna. Can graph neural networks count substructures? In *NeurIPS*, 2020.

[38] Flavio Chierichetti, Ravi Kumar, Silvio Lattanzi, and Sergei Vassilvitskii. Fair clustering through fairlets. *NeurIPS*, 30, 2017.

[39] Yitao Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic modeling. *arXiv:2006.10183*, 2020.

[40] Vincent Cohen-Addad, Fabrizio Grandoni, Euiwoong Lee, Chris Schwiegelshohn, and Ola Svensson. A $(2+\epsilon)$-approximation algorithm for metric k-median. In Michal Koucký and Nikhil Bansal, editors, *STOC*, pages 615–624. ACM, 2025.

[41] Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC*, pages 169–182. ACM, 2021.

[42] Vincent Cohen-Addad, David P. Woodruff, and Samson Zhou. Streaming euclidean k-median and k-means with o(log n) space. In *FOCS*, pages 883–908. IEEE, 2023.

[43] Nilesh N. Dalvi and Dan Suciu. The dichotomy of probabilistic inference for unions of conjunctive queries. *J. ACM*, 59(6):30:1–30:87, 2012.

[44] Guy Van den Broeck, Anton Lykov, Maximilian Schleich, and Dan Suciu. On the tractability of SHAP explanations. *J. Artif. Intell. Res.*, 74:851–886, 2022.

[45] Ilias Diakonikolas, Weihao Kong, and Alistair Stewart. Efficient algorithms and lower bounds for robust linear regression. In *SODA*, pages 2745–2754. SIAM, 2019.

[46] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *ITCS*, pages 214–226, 2012.

[47] Cynthia Dwork and Christina Ilvento. Individual fairness under composition. *FAccT*, 2018.

[48] Cynthia Dwork, Nitin Kohli, and Deirdre Mulligan. Differential privacy in practice: Expose your epsilons! *Journal of Privacy and Confidentiality*, 9(2), 2019.

[49] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

[50] Uriel Feige, Yishay Mansour, and Robert E. Schapire. Learning and inference in the presence of corrupted inputs. In *COLT*, volume 40 of *JMLR Workshop and Conference Proceedings*, pages 637–657, 2015.

[51] Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In *NeurIPS*, 2023.

[52] Sorelle A Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. The (im) possibility of fairness: Different value systems require different mechanisms for fair decision making. *Communications of the ACM*, 64(4):136–143, 2021.

[53] Vincent Froese, Moritz Grillo, and Martin Skutella. Complexity of injectivity and verification of relu neural networks, 2025.

[54] Khashayar Gatmiry, Jonathan A. Kelner, and Holden Lee. Learning mixtures of gaussians using diffusion models. In *COLT*, volume 291 of *Proceedings of Machine Learning Research*, pages 2403–2456. PMLR, 2025.

[55] Quan Geng, Wei Ding, Ruiqi Guo, and Sanjiv Kumar. Tight analysis of privacy and utility tradeoff in approximate differential privacy. In *AISTATS*, volume 108, pages 89–99. PMLR, 2020.

[56] Martin Grohe. The logic of graph neural networks. In *LICS*, pages 1–17, 2021.

[57] Martin Grohe. The descriptive complexity of graph neural networks. *TheoretiCS*, 3, 2024.

[58] Martin Grohe, Christoph Standke, Juno Steegmans, and Jan Van den Bussche. Query languages for neural networks. In *ICDT*, volume 328, pages 9:1–9:18, 2025.

[59] Víctor Gutiérrez-Basulto and Steven Schockaert. From knowledge graph embedding to ontology embedding? an analysis of the compatibility between vector space representations and rules. In *KR*, pages 379–388, 2018.

[60] Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Trans. Assoc. Comput. Linguistics*, 8:156–171, 2020.

[61] Yiding Hao, Dana Angluin, and Robert Frank. Formal language recognition by hard attention transformers: Perspectives from circuit complexity. *Trans. Assoc. Comput. Linguistics*, 10:800–810, 2022.

[62] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *CAV*, pages 3–29, 2017.

[63] Omri Isac, Yoni Zohar, Clark W. Barrett, and Guy Katz. DNN verification, reachability, and the exponential function problem. In *CONCUR*, pages 26:1–26:18, 2023.

[64] Lauren Kirchner Jeff Larson, Surya Mattu and Julia Angwin. How we analyzed the COMPAS recidivism algorithm. Technical report, ProPublica, 2016.

[65] Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why language models hallucinate. *CoRR*, abs/2509.04664, 2025.

[66] Adam Tauman Kalai and Santosh S. Vempala. Calibrated language models must hallucinate. In *STOC*, pages 160–171, 2024.

[67] Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. On the limits of language generation: Trade-offs between hallucination and mode-collapse. In *STOC*, pages 1732–1743, 2025.

[68] Amin Karbasi, Omar Montasser, John Sous, and Grigoris Velegkas. (im)possibility of automated hallucination detection in large language models. *CoRR*, abs/2504.17004, 2025.

[69] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *CAV*, pages 97–117, 2017.

[70] Michael Kearns and Aaron Roth. *The ethical algorithm: The science of socially aware algorithm design*. Oxford University Press, 2019.

[71] Michael J. Kearns, Ming Li, Leonard Pitt, and Leslie G. Valiant. On the learnability of boolean formulae. In *STOC*, pages 285–295. ACM, 1987.

[72] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[73] Sammy Khalife. Graph neural networks with polynomial activations have limited expressivity. *CoRR*, abs/2310.13139, 2023.

[74] Mahmoud Abo Khamis, Hung Q. Ngo, XuanLong Nguyen, Dan Olteanu, and Maximilian Schleich. In-database learning with sparse tensors. In *PODS*, pages 325–340. ACM, 2018.

[75] Mahmoud Abo Khamis, Hung Q. Ngo, and Atri Rudra. FAQ: questions asked frequently. In *PODS*, pages 13–28. ACM, 2016.

[76] Benny Kimelfeld and Christopher Ré. A relational framework for classifier engineering. In *PODS*, pages 5–20. ACM, 2017.

[77] Jon Kleinberg and Fan Wei. Density measures for language generation, 2025.

[78] Jon M. Kleinberg and Sendhil Mullainathan. Language generation in the limit. In *NeurIPS*, 2024.

[79] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. In Christos H. Papadimitriou, editor, *ITCS*, 2017.

[80] Adam R. Klivans, Pravesh K. Kothari, and Raghu Meka. Efficient algorithms for outlier-robust regression. In *COLT*, volume 75 of *Proceedings of Machine Learning Research*, pages 1420–1430. PMLR, 2018.

[81] Ondrej Kuzelka. Weighted first-order model counting in the two-variable fragment with counting quantifiers. *J. Artif. Intell. Res.*, 70:1281–1307, 2021.

[82] Matthias Lanzinger and Pablo Barceló. On the power of the weisfeiler-leman test for graph motif parameters. In *ICLR*, 2024.

[83] Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In Chris Umans, editor, *FOCS*, pages 633–638. IEEE Computer Society, 2017.

[84] Jiaoda Li and Ryan Cotterell. Characterizing the expressivity of transformer language models, 2025.

[85] Jaron Maene, Vincent Derkinderen, and Luc De Raedt. On the hardness of probabilistic neurosymbolic learning. In *ICML*. OpenReview.net, 2024.

[86] Sagar Malhotra, Davide Bizzaro, and Luciano Serafini. Lifted inference beyond first-order logic. *Artif. Intell.*, 342:104310, 2025.

[87] Robin Manhaeve, Sebastijan Dumancic, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. Neural probabilistic logic programming in deepproblog. *Artif. Intell.*, 298:103504, 2021.

[88] Yishay Mansour, Aviad Rubinstein, and Moshe Tennenholtz. Robust probabilistic inference. In *SODA*, pages 449–460. SIAM, 2015.

[89] Luca Marzari, Davide Corsi, Ferdinando Cicalese, and Alessandro Farinelli. The #dnn-verification problem: Counting unsafe inputs for deep neural networks. In *IJCAI*, pages 217–224, 2023.

[90] William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. In *ICLR*, 2024.

[91] Johannes Messner, Ralph Abboud, and İsmail İlkan Ceylan. Temporal knowledge graph completion using box embeddings. In *AAAI*, pages 7779–7787, 2022.

[92] Brent Daniel Mittelstadt, Patrick Allo, Mariarosaria Taddeo, Sandra Wachter, and Luciano Floridi. The ethics of algorithms: Mapping the debate. *Big Data & Society*, 3(2):2053951716679679, 2016.

[93] Omar Montasser, Steve Hanneke, and Nathan Srebro. VC classes are adversarially robustly learnable, but only improperly. In *COLT*, volume 99, pages 2512–2530. PMLR, 2019.

[94] Paolo Morettin, Andrea Passerini, and Roberto Sebastiani. Efficient weighted model integration via smt-based predicate abstraction. In *IJCAI*, pages 720–728, 2017.

[95] Christopher Morris, Floris Geerts, Jan Tönshoff, and Martin Grohe. WL meet VC. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *ICML*, volume 202, 2023.

[96] Christopher Morris, Martin Ritzert, Matthias Fey, William L. Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *AAAI*, pages 4602–4609, 2019.

[97] Sebastian Ordyniak, Giacomo Paesani, Mateusz Rychlicki, and Stefan Szeider. Explaining decisions in ML models: A parameterized complexity analysis. In *KR*, 2024.

[98] Nicolas Papernot, Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. In *ICLR*, 2017.

[99] Aleksandar Pavlovic and Emanuel Sallinger. Expressive: A spatio-functional embedding for knowledge graph completion. In *ICLR*, 2023.

[100] Charlotte Peale, Vinod Raman, and Omer Reingold. Representative language generation. *CoRR*, abs/2505.21819, 2025.

[101] Binghui Peng, Srini Narayanan, and Christos H. Papadimitriou. On limitations of the transformer architecture. *CoRR*, abs/2402.08164, 2024.

[102] Jorge Pérez, Pablo Barceló, and Javier Marinkovic. Attention is turing-complete. *J. Mach. Learn. Res.*, 22:75:1–75:35, 2021.

[103] Maximilian Pflueger, David Tena Cucala, and Egor V. Kostylev. Recurrent graph neural networks and their connections to bisimulation and logic. In *AAAI*, pages 14608–14616, 2024.

[104] Jeff M. Phillips and Wai Ming Tai. Near-optimal coresets of kernel density estimates. *Discret. Comput. Geom.*, 63(4):867–887, 2020.

[105] Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. On fairness and calibration. *NeurIPS*, 30, 2017.

[106] Hongyu Ren, Mikhail Galkin, Zhaocheng Zhu, Jure Leskovec, and Michael Cochez. Neural graph reasoning: A survey on complex logical query answering. *Trans. Mach. Learn. Res.*, 2024, 2024.

[107] Ryan Riegel, Alexander G. Gray, Francois P. S. Luus, Naweed Khan, Ndivhuwo Makondo, Ismail Yunus Akhalwaya, Haifeng Qian, Ronald Fagin, Francisco Barahona, Udit Sharma, Shajith Ikbal, Hima Karanam, Sumit Neelam, Ankita Likhyani, and Santosh K. Srivastava. Logical neural networks. *CoRR*, abs/2006.13155, 2020.

[108] Marco Sälzer and Martin Lange. Reachability is np-complete even for the simplest neural networks. In *RP*, pages 149–164, 2021.

[109] Lena Strobl, William Merrill, Gail Weiss, David Chiang, and Dana Angluin. What formal languages can transformers express? A survey. *Trans. Assoc. Comput. Linguistics*, 12:543–561, 2024.

[110] Dan Suciu, Dan Olteanu, Christopher Ré, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[111] Balder ten Cate and Víctor Dalmau. The product homomorphism problem and applications. In *ICDT*, pages 161–176, 2015.

[112] Balder ten Cate, Victor Dalmau, Maurice Funk, and Carsten Lutz. Extremal fitting problems for conjunctive queries. In *PODS*, pages 89–98. ACM, 2023.

[113] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[114] Timothy van Bremen and Ondrej Kuzelka. Lifted inference with tree axioms. In *KR*, pages 599–608, 2021.

[115] Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In *NeurIPS*, pages 3839–3848, 2018.

[116] Zixuan Wang, Eshaan Nichani, Alberto Bietti, Alex Damian, Daniel Hsu, Jason D. Lee, and Denny Wu. Learning compositional functions with transformers from easy-to-hard data. In *COLT*, volume 291, pages 5632–5711. PMLR, 2025.

[117] Wikipedia contributors. Fairness (machine learning) — Wikipedia, the free encyclopedia, 2024. [Online; accessed 24-September-2024].

[118] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR*. OpenReview.net, 2019.

[119] Andy Yang and David Chiang. Counting like transformers: Compiling temporal counting logic into softmax transformers. *CoRR*, abs/2404.04393, 2024.

[120] Andy Yang, David Chiang, and Dana Angluin. Masked hard-attention transformers recognize exactly the star-free languages. In *NeurIPS*, 2024.

[121] Andy Yang, Lena Strobl, David Chiang, and Dana Angluin. Simulating hard attention using soft attention. *CoRR*, abs/2412.09925, 2024.

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 147

SIMULATING FAST ALGORITHMS WITH LESS MEMORY*

Ryan Williams
MIT and Institute for Advanced Study
rrw@mit.edu

Abstract

My goal is to provide a friendly background exposition of the recent theorem in computational complexity that $\mathsf{TIME}[t] \subseteq \mathsf{SPACE}[\sqrt{t \log t}]$. In English, the theorem says that for every problem that can be solved by a multitape Turing machine running in time t, there is another machine which solves the same problem using only $O(\sqrt{t \log t})$ space. The reader will determine whether or not this goal was met. ☺

1 Introduction

Many basic efficient algorithms consume an amount of memory that's proportional to their running time. For intricate algorithms where brand new information is constantly being computed from past results, it is natural to allocate a little bit of new space in every step or every other step, to store this new information for future use. For example, a dynamic programming solution, in which one builds a table with complex dependencies on previously-computed results, may require a table with its number of cells proportional to the total number of steps taken by the algorithm. In other words, any *particular* algorithm A with $t(n)$ running time may well require $\Theta(t(n))$ units of space. Is this level of space required by *all* algorithms computing the same function as A?

> **Question:** *Are there problems solvable in $t(n)$ time which require $\Omega(t(n))$ units of space to solve?*

*This material is based upon work supported by the National Science Foundation under grants DMS-2424441 (at IAS) and CCF-2420092 (at MIT).

To get a handle on the question, let's view it a little differently. Take any algorithm A that runs in time $t(n)$ (and which may use $\Theta(t(n))$ space). Does there always exist another algorithm B which is equivalent in functionality to A, but uses significantly less space than $O(t(n))$? Observe that a no-answer to this question is equivalent to a yes-answer to our original **Question**.

There does not seem to be any intuitive reason to believe that such an algorithm B always exists, for every algorithm A. It is easy to imagine that some algorithm A on n-bit input x can perform a sequence of operations that are so intricate and sophisticated, that along the way A produces some $t(n)$-bit string Y_x which is *pseudorandom*, or *unpredictable*, or *incompressible*, in some quantifiable sense. Why would (or should) there be an algorithm B which on input x can effectively produce all the bits of Y_x as needed, but uses drastically less memory than $\Omega(t(n))$ bits? It would only take one "weird" A capable of producing nastily incompressible strings on its inputs, in order for our **Question** to have a yes-answer.

A surprising answer. Amazingly, a sequence of three papers starting with Hopcroft, Paul, and Valiant in 1975 [8, 14, 7] showed that the answer to the **Question** is **NO** for essentially all reasonable models of computation.[1] Namely, these papers show that $t(n)$-time algorithms can always be simulated in only $O(t(n)/\log t(n))$ space![2] In complexity notation, we have the inclusion of classes

$$\mathsf{TIME}[t(n)] \subseteq \mathsf{SPACE}[t(n)/\log t(n)],$$

and this inclusion holds for a variety of underlying computational models.

However, there is a practical caveat to these simulation results: while the resulting algorithm B (simulating time-$t(n)$ algorithm A) runs in $O(t(n)/\log t(n))$ space as desired, B also takes running time $2^{O(t(n)^{\varepsilon})}$ for $\varepsilon > 0$ (but ε can be as small as desired). Therefore, we've exchanged a tiny-looking log-factor improvement in space usage, for an exponential blow-up in the running time. Hopcroft, Paul, and Valiant [8] also show that a time-space tradeoff is possible: for all reasonable functions $b(n)$, the (multitape Turing machine) algorithm B can be implemented in $O(t(n)/\log\log b(n))$ space and $O(b(n) \cdot t(n))$ time. So, for example, there is a B running in $O(t(n)/\log\log t(n))$ space and $t(n) \cdot 2^{(\log t(n))^{\varepsilon}}$ time, for every $\varepsilon > 0$.

Why should we care? What is the use of such a simulation, showing that time-t computations can be simulated in $o(t)$ space but using substantially more time?

[1] Note that these three papers were titled "On Time Versus Space", "On Time Versus Space II", and "On Time Versus Space III", respectively. It took a surprising level of self-control for me to resist naming my contribution "On Time Versus Space IV".

[2] We also note that the paper [8] cites an earlier work by Paterson-Valiant [16] which showed that every Boolean circuit of size $s(n) \geq n$ has an equivalent circuit of depth $O(s(n)/\log s(n))$, an equally surprising result in circuit complexity.

On the one hand, if one allows the simulation time to be exponential, such a simulation would become useless in practice. On the other hand, if one uses the polynomial-time version of the simulation instead, a log-log-factor improvement in space would apparently not be too noticeable. One hypothetical application would be to increase the computational ability of a device which has low memory and little access to the rest of us, such as an aging deep space probe. We imagine this probe has significantly less on-board memory than a modern computer, but has particular inputs (observations it can make) which are unavailable to computers on Earth. Instead of trying to send those inputs back to Earth, which could be intractable for various reasons, the probe could apply a $\mathsf{TIME}[t] \subseteq \mathsf{SPACE}[o(t)]$ type result to use a more memory-restricted version of a time-efficient algorithm, allowing strictly more computations to be performed in principle. Despite this colorful scenario, it is still not obvious (to me) that there is a concrete practical application of $\mathsf{TIME}[t] \subseteq \mathsf{SPACE}[o(t)]$ results when the time complexity of the $o(t)$-space simulation is large.

Nevertheless, there are at least two good non-practical (theoretical) answers to the question of why we should care.

1. Time and space are fundamental resources for computing, and understanding how the two relate to each other is one of the most basic problems that we could study. Any non-trivial result relating the TIME and SPACE complexity classes is worth knowing.

2. We can use such surprising simulations to prove impossibility results (a.k.a. lower bounds) for simulating space-bounded computations quickly. The famous P vs PSPACE question is equivalent to asking whether $\mathsf{SPACE}[n]$ is contained in $\mathsf{TIME}[n^k]$ for some constant $k \geq 1$. Using the simulations of [8, 14, 7], we can conclude that $\mathsf{SPACE}[n]$ is not contained in $\mathsf{TIME}[o(n \log n)]$. Otherwise, we would be able to derive that

$$\mathsf{SPACE}[n^2 \log n] \subseteq \mathsf{TIME}[o(n^2 \log^2 n)] \subseteq \mathsf{SPACE}[o(n^2 \log n)],$$

a contradiction to the Space Hierarchy Theorem [19]. Thus these simulations provide a very modest time lower bound against linear space.

Should we expect a better simulation? While simulations of TIME in smaller SPACE are indeed useful for proving lower bounds, the approach seems like overkill. To prove $\mathsf{SPACE}[n] \not\subseteq \mathsf{TIME}[T(n)]$ for *large* $T(n)$, we only have to find *some* hard problem in linear space that cannot be solved in $T(n)$ time. This seems to be much easier (and more likely to be true) than simulating *every* problem in $\mathsf{TIME}[T(n)]$ inside of $\mathsf{SPACE}[o(n)]$. In the words of Ben Brubaker of Quanta magazine [2], the approach "feels almost cartoonishly excessive, akin to proving

a suspected murderer guilty by establishing an ironclad alibi for everyone else on the planet."

Apparently it was believed that the $\mathsf{TIME}[t] \subseteq \mathsf{SPACE}[t/\log t]$ simulation could not be improved by much. First of all, there were tight lower bounds for the general approach: the various strategies of all prior work [8, 14, 7] can be captured by a simple pebble game on directed acyclic graphs, and extensive work [15, 10] showed that there are graphs on which all the time-space tradeoffs proved are optimal. Thus, it was well-known that if the simulation could be improved, it would have to be done using very different techniques from prior work.

At some point, researchers began exploring the consequences of assuming that Hopcroft-Paul-Valiant and its successors cannot be improved in a substantial way. In one of the first derandomization papers giving serious evidence that $\mathsf{P} = \mathsf{RP}$, Sipser [20] showed that assuming a certain expander conjecture (which was later proven to be true [21]) and assuming the conjecture that $\mathsf{TIME}[t] \not\subseteq \mathsf{SPACE}[t^{1-\varepsilon}]$ for all $\varepsilon > 0$, we can conclude $\mathsf{P} = \mathsf{RP}$. The later pioneering work by Nisan and Wigderson [11] showed that assuming $\mathsf{TIME}[t] \not\subseteq \mathsf{SPACE}[t^{1-\varepsilon}]$ for all $\varepsilon > 0$ also implies a derandomization of BPP. Crucially, their work also shows the assumption $\mathsf{TIME}[t] \not\subseteq \mathsf{SPACE}[t^{1-\varepsilon}]$ can be relaxed to the assumption that $\mathsf{TIME}[t]$ does not have Boolean circuits of size $t^{1-\varepsilon}$ (i.e., $\mathsf{TIME}[t] \not\subseteq \mathsf{SIZE}[t^{1-\varepsilon}]$). That is, Nisan and Wigderson showed we do not need *space lower bounds against time* to obtain derandomization; we only need *circuit size lower bounds against time*. Our recent work simulating time in smaller space contradicts the conjecture that $\mathsf{TIME}[t] \not\subseteq \mathsf{SPACE}[t^{1-\varepsilon}]$ for all $\varepsilon > 0$, but it says nothing about the more plausible conjecture that $\mathsf{TIME}[t] \not\subseteq \mathsf{SIZE}[t^{1-\varepsilon}]$ for all $\varepsilon > 0$.

The new surprise. If you read the abstract of this paper, there is unfortunately no real surprise. ☺ The simulation of $\mathsf{TIME}[t]$ inside of $\mathsf{SPACE}[t/\log t]$ can be radically improved for the multitape Turing machine model:

Theorem 1.1 ([22])**.** *Every multitape Turing machine running in time $t(n)$ can be simulated by another multitape Turing machine using space only $O(\sqrt{t(n)\log t(n)})$.*

To appreciate the scope of this result, we note the surprising power of multitape Turing machines. Recall that these are simply Turing machines endowed with a constant number of tape heads that may independently access any constant number of tapes. (In each step of such a machine, each tape head reads its current cell, writes to its current cell, and may move left or right on the tape.) Already two-tape Turing machines appear to be quite powerful: the P-complete Circuit Evaluation problem [13], sorting [17], fast matrix multiplications, fast Fourier transforms (and integer multiplication), and many other basic algorithms [18] can be implemented on such devices with running times that are comparable (up to

polylogarithmic factors) to the best known running times on arbitrary random-access models of computation. In fact, it is a major open question to find any decision problem that can be solved in $O(T(n))$ time on a random access model of computation (for some $T(n) \geq n$), which cannot be solved in $T(n) \cdot \text{poly}(\log T(n))$ time on a two-tape Turing machine. Despite their restricted sequential access to tapes, multitape Turing machines can perform surprisingly quick *amortized* computations if they are allowed (at least linear) time to shuffle data around.

2 How Does It Work?

The main idea behind proving this result is to read the correct pair of papers at the same time, and to believe very strongly that the ideas of these papers can be successfully combined. The two papers to read are:

1. Hopcroft, Paul, and Valiant's "On Time Versus Space" from FOCS 1975 [8], and

2. Cook and Mertz's "Tree Evaluation in $O(\log n \cdot \log\log n)$ Space" from STOC 2024 [4]

We'll go through both of these soon. Prior to discussing either of them, we will introduce a graph-theoretic notion for reasoning about a time-t computation. This notion makes sense for essentially any reasonable serial model of computation, and it models the "flow of information" in the computation.

Given a "machine" M of some type and an input x to run on M, we define a *computation graph* $G_{M,x}$ in the following way.

First, there are $t + 1$ nodes labeled $0, 1, \ldots, t$, representing the steps of the computation. Each node i will be tagged with some bits of information info(i), which informally is the information computed by M in the i-th step. (For now, we think of info(0) as encoding the initial state of the machine, which is passed to step 1.) For example, if M is a multitape Turing machine, info(i) may simply denote the transition computed in the i-th step: the states at the beginning and end of the i-th step, the symbols read, the symbols written, and the head positions moved.

Second, we put a directed edge (i, j) in $G_{M,x}$ if and only if *info*(i) *is needed to compute info*(j). For example, in basically every computational model, we have the edges $(i - 1, i)$ for all $i \in [t] := \{1, \ldots, t\}$: we always need the "program counter" or "state" at the end of step $i - 1$ in order to compute step i. We also need an edge (i, j) if there is a "register" or "cell" written to in step i which is read next in step j. For reasonable models, we only access $O(1)$ memory locations in any one step, so the indegree of each node of $G_{M,x}$ is $O(1)$. Observe that such a directed graph is always acyclic.

The graph $G_{N,x}$ is *defined* so that the following important property holds. For every $j \in [t]$, let in-nbrs(j) be the set of all i such that (i, j) is an edge. Then we have the principle:

> *Given info(i) for all $i \in$ in-nbrs(i), we can compute info(j) in $O(1)$ steps.*

(Indeed, the computation we are doing is effectively simulating M on x for one step.) Our goal is to determine info(t), which will tell us whether the computation accepts or rejects.

What we need from HPV. Motivated by the above principle, HPV define a *pebble game* to be played on DAGs such as $G_{M,x}$. The game has the following rules:

1. We can always put a pebble on a source node.
2. Given pebbles on all nodes in in-nbrs(i), we can put a pebble on node i.
3. We can remove pebbles at any time.

Here, the act of placing a pebble directly corresponds to simulating one step of M on x. Therefore, a very interesting question is: how many pebbles do we need to pebble the nodes of an n-node graph? HPV show that for directed acyclic graphs of indegree $O(1)$, there is a pebbling algorithm which uses only $O(n/\log n)$ pebbles. This directly corresponds to a memory-bounded algorithm: in principle, we only need to keep around $O(t/\log t)$ "intermediate results" of the computation, in order to determine the final outcome. This strongly suggests that we only need $O(t/\log t)$ space to simulate a time-t computation.

However, we don't know this graph $G_{M,x}$ in advance, and we can't naively store this $(t + 1)$-node DAG in only $O(t/\log t)$ space. For multitape Turing machines, HPV get around this issue by showing how to *compress* the computation graph, by partitioning the time and space of the computation into pieces. (Later work extending their simulation to random-access models [14, 7] works by performing clever space-saving simulations which effectively model good pebbling strategies, *without* storing a computation graph explicitly. This point will be discussed further in the last section.) This compressed graph notion will prove very useful for us.

Fix a multitape Turing machine M and input x. For a parameter $b \in [1, t]$ (which we will think of here as close to $\sqrt{t}$), each tape of M (on x) is partitioned into *blocks* of b contiguous cells each, and the time of M (on x) is partitioned into *intervals* of b consecutive steps. Our compressed computation graph $G'_{M,x}$ has only $n := O(t/b)$ nodes, one for each interval. For an interval i, we define info(i)

to be all the information computed by M (on x) during interval i. This includes the state and tape head positions at the end of the interval, along with the contents of tape blocks accessed during interval i, at the end of the interval. We define info(0) to simply be the *initial configuration* of M on x: the initial state of M and initial head positions, along with the input x. Similarly to the definition of the original graph $G_{M,x}$, we put an edge (i, j) exactly when we need some data from info(i) to compute info(j).[3]

As each tape block has b cells and intervals are b steps long, at most two tape blocks may be accessed in an interval. Since there are a constant number of tapes, the indegree of each node in $G'_{M,x}$ is still $O(1)$. Also observe that, when we are given info(i) for all in-neighbors of a node j, we can compute info(j) in $O(b)$ steps.

We can view the problem of computing info(n) in the $(n + 1)$-node graph $G'_{M,x}$ as a special Circuit Evaluation problem, where we have a circuit of n nodes with wires given by the edges of the graph, and where every gate of the circuit takes in $O(b)$ bits of information from a constant number of in-neighbors, runs for $O(b)$ steps on this information, and outputs $O(b)$ bits of information.

As before with the graph $G_{M,x}$, we cannot really know the graph $G'_{M,x}$ in advance; it needs to be computed somehow.[4] Since our graph $G'_{M,x}$ has only $O(t/b)$ nodes, which will be approximately $\sqrt{t}$, we could enumerate over all possible such graphs in $O(t/b \cdot \log(t/b))$ space (in fact, due to their structure coming from multitape Turing machines, these graphs can be encoded in $O(t/b)$ bits, by guessing the head movements across tape blocks; see the paper [22]). For each candidate graph G'', we may attempt to simulate M on x by pebbling G''. If G'' is not suitable, then it must "violate" the head movements of M on x in some way: such a violation will arise if we are missing an edge (i, j) but we needed info(i) in order to compute info(j). If we find that some edge is missing for us, then we move on to the next possible G''. We know that the $O(t/b)$-node $G'_{M,x}$ can be pebbled using only $O(t/b)/\log(t/b)$ pebbles, and we know that storing each pebble (which is just info(i) for various i) takes $O(b)$ space. Therefore for $b = t^{\varepsilon}$ for some $\varepsilon \in (0, 1)$, the overall space usage is $O(t/\log t)$.

What we need from Cook and Mertz. Cook and Mertz [4] study an apparently unrelated problem, introduced by (another) Cook et al. [5], called Tree Evaluation. For the purposes of this article, we will think of Tree Evaluation as the following problem. We are given:

[3] Hopcroft-Paul-Valiant also add other technical conditions, such as making the machine M "block-respecting", but they aren't really necessary.

[4] If M was an "oblivious" Turing machine with structure, then we could compute the edges of $G_{M,x}$, but making an arbitrary multitape Turing machine oblivious requires extra time overhead. See the paper [22] for details.

- a rooted tree T of height h, where each node of T has d children and $d \leq O(1)$,
- a function $f : (\{0,1\}^b)^d \to \{0,1\}^b$, presented as a table of $2^{d \cdot b}$ values, each of which are b-bits long, and
- for each leaf ℓ of T, we are given a value $v_\ell \in \{0,1\}^b$.

Note that the tree T could be very large, and have nearly d^{h+1} nodes in total. For each inner node u of T with children $u_1, \ldots, u_d$, we inductively define the value v_u to be $f(v_{u_1}, \ldots, v_{u_d})$ (we concatenate the values $v_{u_1}, \ldots, v_{u_d}$ and feed the db-bit result as input to f). Starting from the leaves, this yields a value v_u for every u in T. The goal of Tree Evaluation is to compute v_r, where r is the root of T.

We can apply the pebble game to any tree T, which we can think of as a directed acyclic graph where the children have arcs to their parents. A simple depth-first pebbling strategy shows that every Tree Evaluation instance (construed as a DAG) can be pebbled using $O(h)$ pebbles, storing at most two pebbles at each level of the tree. Since each node value takes b bits to describe, this corresponds to an evaluation algorithm using $O(h \cdot b)$ space. For pebbling, [5] show that we can't do better.

Incredibly, Cook and Mertz [4] show how to use only $O(h \log b + b)$ space, replacing the *product* of h and b with nearly the *sum* of h and b. In trying to describe Cook and Mertz's algorithm to others, Arthur C. Clarke's quote often comes to my mind: *Any sufficiently advanced technology is indistinguishable from magic* [3]. We are very accustomed to thinking about the space complexity of algorithms in particular rigid ways: viewing the space usage in terms of some stack, bounding the maximum height of the stack, bounding the "width" a.k.a. the number of bits needed on one layer of the stack, and so on. For Tree Evaluation, our stack height is h and our width is b. Cook and Mertz's algorithm provides a genuinely new way of reusing space in a recursive algorithm, inspired by work in *catalytic* computation which was initiated by Buhrman et al. [1]. They still have a stack of height h, but its width is only $O(\log b)$. Instead of having b bits at every layer of the stack, there is a common storage of $O(b)$ bits which is reused many times over, at every layer of the stack. Time and space prevent us from describing their algorithm in detail; beyond their paper, see [6, 22] for alternative expositions of their great result.

What does this have to do with computation graphs? We have a surprisingly space-efficient algorithm for evaluating *trees* in which each leaf holds a b-bit value, and each inner node of the tree computes a function from $O(b)$ bits to b bits, from children to parent. The problem of evaluating $G'_{M,x}$ amounts to evaluating a *circuit* or *directed acyclic graph* in which each source holds a b-bit value,

and each non-source node computes a function from $O(b)$ bits to b bits, taking in values from its in-neighbors.

A folklore result in Boolean circuit complexity states that every Boolean circuit C of depth h (where each gate has arbitrary fan-out) can be computed by an equivalent Boolean *formula* F of depth h, where each gate has fan-out 1. The idea is, starting from the output gate of C, to "unroll" C backwards: we perform a depth-first traversal of C by traversing arcs backwards, in effect enumerating over all paths from the output gate to some source node of C. Doing so, we obtain a *formula* where each gate of the formula can be given a *name* by specifying a path from some gate of C to the output gate of C. The formula F may contain many copies of gates from C, but its overall depth will be the same as that of C. Moreover, this transformation is space-efficient: given the encoding of C and a path P from a gate g to the output of C, the path P *names* a gate in the formula F. We can easily compute the names of the children of P in the formula F, using the circuit C. For an n-gate circuit C of depth d and $O(1)$ fan-in, it takes only $O(d)$ space to specify a gate in F (by walking backwards from the output of C), and to compute the names of its children.

We can think of Boolean circuits as a special case of a computation graph where the block size $b = 1$, and we can think of Boolean formula evaluation as a special case of Tree Evaluation where $b = 1$. The above transformation from circuits to formulas works equally well for circuits and formulas which pass along b-bit values on all their wires, where $b > 1$. Therefore we can apply the Cook-Mertz procedure for Tree Evaluation to a candidate computation graph G'', in order to simulate a multitape Turing machine M on an input x. Setting $b = \sqrt{t}$ to be the length of a time interval, the total number of intervals becomes $\Theta(t/b) = h = \Theta(\sqrt{t})$ and the simulation runs in $O(\sqrt{t} \cdot \log t)$ space overall. To minimize the space usage, we want to set

$$b = h \log(b).$$

Slightly adjusting b to be $\sqrt{t \log t}$, making the intervals a little longer, the number of intervals becomes $O(\sqrt{t/\log t})$ and we obtain the final $O(\sqrt{t \log t})$ space bound.

3 What's Next?

I believe that currently the most significant open problem is to extend the simulation of $\mathsf{TIME}[t]$ in $\mathsf{SPACE}[\sqrt{t \log t}]$ to more general random-access models of computation (RAMs), or to give compelling evidence that this cannot be done. A weaker but still very interesting result would be to show that time-$t(n)$ on RAMs can be simulated in space $O(t(n)^{1-\varepsilon})$ for *some* $\varepsilon > 0$. Let me outline some ideas for how this latter problem might be solved.

Concretely, let us suppose we have a Turing machine with access to a *tree storage* [14]: instead of a tape storage with cells arranged in a line, the Turing machine receives its input written on a complete binary tree of depth $O(\log n)$, where each node of the tree is now a cell. The tape head starts at the root of the tree. At each step, the Turing machine reads from the current cell, changes its state, overwrites the current cell, and moves its head to either the parent of the current cell, the left child of the current cell, or the right child of the current cell. For all reasonable random-access models that we know of, those that run in time $t(n)$ can be simulated on a tree storage in $t(n)\cdot\text{poly}(\log t(n))$ time. Thus it suffices to extend the space-efficient simulation to a tree storage. It does not seem that we can easily embed the computation graph for such a Turing machine into a nice Tree Evaluation instance that we can succinctly store and manipulate. But perhaps if we can go "beyond" Tree Evaluation—doing something similar to what Tree Evaluation does in spirit, without actually storing an explicit graph—then perhaps we can obtain a better simulation. This is precisely how the extensions of Hopcroft-Paul-Valiant to random-access models work [14, 7]: these extended simulations do *not* store a computation graph explicitly. Instead, they simply present particular recursive strategies for saving space, and argue *in their analysis* that their simulation corresponds to "pebbling" a t-node computation graph representing the RAM computation, where every node corresponds to a single step.

Another possible direction (towards a better simulation for random-access models) would be to improve the best-known $O(\sqrt{t(n)})$ space bound for simulating a one-tape Turing machine running in time $O(t(n))$ [9, 12]. It is not hard to show that every $O(t(n))$-time Turing machine with a tree storage can be simulated by a standard one-tape Turing machine in about $O(t(n)^2)$ time, by simply sweeping through the entire contents of the tree storage during every simulated step. Therefore if we could simulate time $T(n)$ on *one-tape* Turing machines in $O(T(n)^{1/2-\varepsilon})$ for some $\varepsilon > 0$ (a slight improvement over the multitape case), then we would have a new space-efficient simulation of $O(t(n))$-time RAMs as well.

References

[1] Harry Buhrman, Richard Cleve, Michal Koucký, Bruno Loff, and Florian Speelman. Computing with a full memory: catalytic space. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 857–866. ACM, 2014. doi:10.1145/2591796.2591874.

[2] Ben Brubaker. For algorithms, a little memory outweighs a lot of time. Quanta Magazine, May 2025. URL: https://www.quantamagazine.org/for-algorithms-a-little-memory-outweighs-...-20250521/.

[3] Arthur C. Clarke. *Profiles of the future: an inquiry into the limits of the possible*. Harper & Row, New York; London, rev. ed. edition, 1973.

[4] James Cook and Ian Mertz. Tree evaluation is in space O(log n · log log n). In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1268–1278. ACM, 2024. URL: https://doi.org/10.1145/3618260.3649664.

[5] Stephen A. Cook, Pierre McKenzie, Dustin Wehr, Mark Braverman, and Rahul Santhanam. Pebbles and branching programs for tree evaluation. *ACM Trans. Comput. Theory*, 3(2):4:1–4:43, 2012. URL: https://doi.org/10.1145/2077336.2077337.

[6] Oded Goldreich. On the Cook-Mertz Tree Evaluation procedure. *Electron. Colloquium Comput. Complex.*, TR24-109, 2024. URL: https://eccc.weizmann.ac.il/report/2024/109.

[7] Joseph Y. Halpern, Michael C. Loui, Albert R. Meyer, and Daniel Weise. On time versus space III. *Math. Syst. Theory*, 19(1):13–28, 1986. URL: https://doi.org/10.1007/BF01704903.

[8] John E. Hopcroft, Wolfgang J. Paul, and Leslie G. Valiant. On time versus space. *J. ACM*, 24(2):332–337, 1977. Conference version in FOCS'75. URL: https://doi.org/10.1145/322003.322015.

[9] John E. Hopcroft and Jeffrey D. Ullman. Relations between time and tape complexities. *J. ACM*, 15(3):414–427, 1968. URL: https://doi.org/10.1145/321466.321474.

[10] Thomas Lengauer and Robert Endre Tarjan. Asymptotically tight bounds on time-space trade-offs in a pebble game. *J. ACM*, 29(4):1087–1130, 1982. doi:10.1145/322344.322354.

[11] Noam Nisan and Avi Wigderson. Hardness vs randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. URL: https://doi.org/10.1016/S0022-0000(05)80043-1.

[12] Mike Paterson. Tape bounds for time-bounded Turing machines. *J. Comput. Syst. Sci.*, 6(2):116–124, 1972. URL: https://doi.org/10.1016/S0022-0000(72)80017-5.

[13] Nicholas Pippenger. Fast simulation of combinational logic networks by machines without random-access storage. In *Proceedings of the Fifteenth Annual Allerton Conference on Communication, Control and Computing*, pages 25–33, 1977.

[14] Wolfgang J. Paul and Rüdiger Reischuk. On time versus space II. *J. Comput. Syst. Sci.*, 22(3):312–327, 1981. URL: https://doi.org/10.1016/0022-0000(81)90035-0.

[15] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. *Math. Syst. Theory*, 10:239–251, 1977. doi:10.1007/BF01683275.

[16] Mike Paterson and Leslie G. Valiant. Circuit size is nonlinear in depth. *Theor. Comput. Sci.*, 2(3):397–400, 1976. URL: https://doi.org/10.1016/0304-3975(76)90090-6.

[17] Claus-Peter Schnorr. Satisfiability is quasilinear complete in NQL. *J. ACM*, 25(1):136–145, 1978. doi:10.1145/322047.322060.

[18] Arnold Schönhage, Andreas F. W. Grotefeld, and Ekkehart Vetter. *Fast algorithms: a multitape Turing machine implementation*. Bibliographisches Institut, Mannheim, 1994.

[19] Richard Edwin Stearns, Juris Hartmanis, and Philip M. Lewis II. Hierarchies of memory limited computations. In *6th Annual Symposium on Switching Circuit Theory and Logical Design*, pages 179–190. IEEE Computer Society, 1965. URL: https://doi.org/10.1109/FOCS.1965.11.

[20] Michael Sipser. Expanders, randomness, or time versus space. *J. Comput. Syst. Sci.*, 36(3):379–383, 1988. URL: https://doi.org/10.1016/0022-0000(88)90035-9.

[21] Michael E. Saks, Aravind Srinivasan, and Shiyu Zhou. Explicit or-dispersers with polylogarithmic degree. *J. ACM*, 45(1):123–154, 1998. URL: https://doi.org/10.1145/273865.273915.

[22] R. Ryan Williams. Simulating time with square-root space. In Michal Koucký and Nikhil Bansal, editors, *Proceedings of the 57th Annual ACM Symposium on Theory of Computing, STOC 2025, Prague, Czechia, June 23-27, 2025*, pages 13–23. ACM, 2025. doi:10.1145/3717823.3718225.

THE DISTRIBUTED COMPUTING COLUMN

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

This month, in the Distributed Computing Column, Amitabh Trehan explores a seemingly simple problem: flooding with no memory. Here we have perhaps the simplest possible distributed algorithm, and yet he illustrates surprising depth. Along the way he raises the fundamental question of when is there only one algorithmic solution to a problem (and what does that mean)?

The Distributed Computing Column is particularly interested in contributions that propose interesting new directions and summarize important open problems in areas of interest. We are also interested in understanding the impact of distributed computing, inteprereted broadly. If you would like to write such a column, please contact me.

BEATCS no 147

AMNESIAC FLOODING AND THE CURIOUS CASE OF UNIQUE ALGORITHMS

Amitabh Trehan Durham University
amitabh.trehan@durham.ac.uk

Abstract

Amnesiac Flooding [6, 7, 9] is the stateless/historyless/amnesiac variant of probably the oldest and simplest of distributed algorithms: (classic) flooding. Beginning with a message at a set of initiator(s), the algorithm at every node is simply a single rule: if the message is received from some neighbour(s), immediately forward it to the rest of the neighbours. Note that unlike classic flooding, no copy of the message or history of the flooding is retained to ensure termination/quiescence of the messages. Yet, surprisingly, amnesiac flooding begun from any set of initiators (even in different rounds) terminates on all undirected graphs in the synchronous message passing model. Moreover, it terminates in optimal rounds in bipartite graphs and is at most twice as slow in non-bipartite graphs. A series of results have followed the first discovery, improving our understanding of the process and its variants.

Even more recently Austin, Gadouleau, Mertzios, and Trehan [3, 5] discovered *uniqueness* - under certain reasonable conditions including statelessness, amnesiac flooding is the only algorithm that solves terminating broadcast! As algorithm designers, the study of lower bounds and impossibility results (no algorithm exists) is well established, but we are not aware of results concerning uniqueness or, in general, even a sensible notion of countability of solutions to problems. This article presents some of the fundamental and interesting results around amnesiac flooding from its discovery to the present while not being a comprehensive review of the area that the initial result has spawned.

1 Introduction

Sometimes sloppiness can lead to happy accidents—at least, it did so, in this case. While giving a lecture which introduced the classic flooding algorithm, I neglected to add the standard condition that a node discards a message if it sees it again

(making it necessary for the node to keep copies of messages flooded in the past). The algorithm that remained (Definition 1.1), I named *Amnesiac Flooding* due to its in-built forgetfulness. The classic flooding algorithm is one of the earliest and simplest algorithms in the field of distributed computing. Informally, the classic flooding algorithm is: *if I have the message, I send it immediately to all my neighbours (or to all my neighbours who have not sent me the message in the preceding round); If I get the message again, I ignore the new receipt and do nothing,* Flooding is extremely useful because it is simple, it achieves *broadcast* (i.e. the message is received by every node in the network) and it terminates in optimal time (equivalent to the diameter of the network). Not surprisingly, it is used as a fundamental building block for many other algorithms. For example, in Peleg's time optimal Leader Election algorithm [11], a leader is elected by every node flooding the lowest ID it has seen so far to all its neighbours for diameter number of rounds. As James Aspnes summarises: "Flooding is about the simplest of all distributed algorithms. It is dumb and expensive but easy to implement" [2].

Amnesiac Flooding is thus so: *if I have the message, I send it immediately to all my neighbours who have not just sent me the message.* This variant obviously achieves broadcast, but does it terminate? The immediate answer of every expert I talked to initially when I posed the question was that the process should be non-terminating, i.e., messages would circulate ad-infinitum and all hell would break loose in Network land. Nobody, however, had a proof either way, and the subsequent stubborn search to prove the 'obvious' led to a fruitful path of discovery and even to the creation of a wikipedia page [1].

1.1 Amnesiac Flooding

The algorithm in the fault-free synchronous message passing model is defined as follows. The definition below is for all initiators starting simultaneously (for more general definitions, refer to [3–5]).

Definition 1.1. ***Amnesiac flooding algorithm (Synchronous).*** *(from [5] (Adapted from [9]) Let $G = (V, E)$ be an undirected graph, with vertices V and edges E (representing a network where the vertices represent the nodes of the network and edges represent the connections between the nodes). Computation proceeds in synchronous 'rounds' where each round consists of nodes receiving messages sent from their neighbours. A receiving node then sends messages to some neighbours in the next round. No messages are lost in transit. The algorithm is defined by the following rules:*

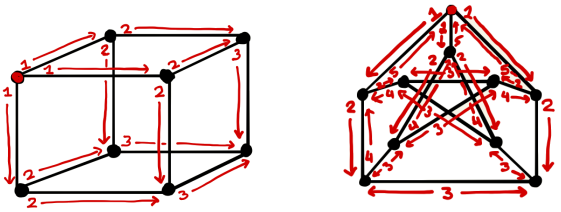
(i) All nodes from a subset of sources or initial nodes $I \subseteq V$ send a message M to all of their neighbours in round 1.

(ii) In subsequent rounds, every node that received M from a neighbour in the previous round, sends M to all, and only, those nodes from which it did not receive M. Flooding terminates *when M is no longer sent to any node in the network.*

In subsequent discussions, we shall restrict ourselves to a single initiator node (i.e. $|I| = 1$), unless otherwise stated. In the next section (Section 1.2), we try out some simple examples by hand (Figures1 and 2).

1.2 Some Illustrative Examples

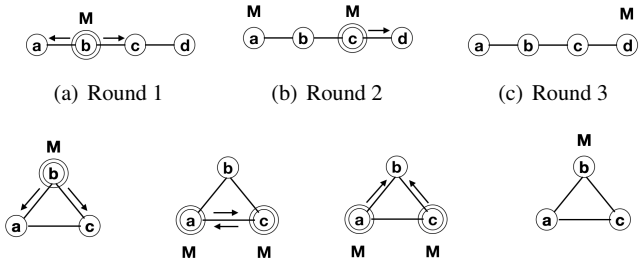
Consider two well known graphs in Figure 1; the hypercube (cube in 3 dimensions) graph and the Petersen graph. These graphs are symmetric so the starting point does not influence the execution of amnesiac flooding with respect to termination time. Consider *AF* on the hypercube first (Figure 1(a)) - it is easy to see that it stops in round 3 when the node diagonally opposite the origin gets *M* from all of its neighbours simultaneously. On the Petersen graph (Figure 1(b)), the process terminates in 5 rounds stopping at the origin itself. In terms of diameter, termination on the hypercube takes diameter time but the Petersen graph takes 2* diameter + 1 rounds.



(a) *AF* on the Hypercube (b) *AF* on the Petersen Graph

Figure 1: Two well known graph topologies (Hypercube and the Petersen Graph) and the execution of Amnesiac flooding (AF) (from the red coloured node) on them. Arrows point to direction of the transmission with the label giving the round number. Double headed arrows indicate the message crossing in both directions.

To get a better understanding, let's look at some simpler base cases. Figure 2 shows flooding over a line graph and a triangle graph. On the line on 4 nodes, the process beginning with the node *b* in the figure terminates at the ends of the graph taking only 2 rounds, which is equal to the *eccentricity*(the length of the shortest longest path from a vertex in a graph) of node *b* in the graph (whose diameter is 3). Note that a line is an example of a bipartite graph. In the triangle graph,



(a) Round 1 (b) Round 2 (c) Round 3

(d) Round 1 (e) Round 2 (f) Round 3 (g) Round 4

Figure 2: AF over line and triangle networks with circled nodes sending M in that round. On the above line, beginning with node b, AF terminates in 2 ($< diameter = 3$) rounds. When AF is executed over the above Triangle graph beginning with node b, both node a and c send M to each other in round 2 and to b in round 3. This is an odd (# nodes) cycle topology (also, a clique) with termination taking $2 * diameter + 1$ rounds.

termination from any initiator (the graph being symmetric) takes only 3 rounds, However, here the diameter is only 1. The triangle is also the smallest clique and the smallest non-trivial cycle with an odd number of nodes (an important subgraph, as it turns out). Potentially, a kind of pattern is emerging: On a symmetric bipartite graph (such as the hypercube) termination takes exactly diameter rounds, on a (non-symmetric) bipartite graph such as a line, it may even take less than diameter time but never more; On symmetric non-bipartite graphs such as a triangle and the Petersen graph, it is taking time that is exactly twice the diameter of the graph plus 1.

2 Termination

Ultimately, it turns out that this simple stateless process of Amnesiac flooding is terminating and termination time stated in terms of eccentricity of the initiator nodes has a strong co-relation with bipartiteness of the graph.

2.1 The First Proof

The first proof showing that amnesiac flood terminates on any graph appears in [6, 7, 9]. This is a proof by contradiction. It relies on a simple but useful observation:

Observation 2.1. *Any event that occurs infinitely often, observed on a discrete*

time scale, must have at least two occurrences separated by an even interval.

In fact, one can make a stronger claim - any event that occurs 3 or more times must have two occurrences separated by an even interval. Let these three occurrences be at time steps t_1, t_2, and t_3. If all these t's are even, then all the intervals are even (e.g. $t_2 - t_1$). If all these t's are odd, then again all the intervals are even since the difference of two odd numbers is even. For the other cases, either there are two even numbers and their difference is even, or there are two odd numbers, and their difference is again even. Using the observation above, if amnesiac flooding was non-terminating, there must be some node that receives the message infinitely often, in particular, at least 3 times, and therefore, at an even interval. We assume such a node and prove by contradiction that there is no such node showing that amnesiac flooding is terminating. The formal proof follows below (from [7]):

Definition 2.1. *Let G be a graph. The round-sets $R_0, R_1, \ldots$ are defined as:*

R_0 *is the singleton containing an initial node,*
R_i *is the set of nodes which receive a message at round i* $(i \geq 1)$.

Clearly, if $R_j = \emptyset$ for some $j \geq 0$, then $R_i = \emptyset$ for all $i \geq j$. We shall refer to rounds R_i, where $R_i \neq \emptyset$, as *active* rounds.

Theorem 2.2. *Any node $g \in G$ is contained in at most two distinct round-sets.*

Proof. Define $\mathcal{R}$ to be the set of finite sequences of consecutive round-sets of the form:

$$\underline{R} = R_s, \ldots, R_{s+d} \quad \textit{where } s \geq 0,\ d > 0, \textit{ and } R_s \cap R_{s+d} \neq \emptyset\ . \tag{1}$$

In (1), s is the *start-point* $s(\underline{R})$ and d is the *duration* $d(\underline{R})$ of $\underline{R}$. Note that, a node $g \in G$ belonging to R_s and R_{s+d} may also belong to other R_i in (1). If a node $g \in G$ occurs in three different round-sets R_{i_1}, R_{i_2} and R_{i_3}, then the duration between R_{i_1} and R_{i_2}, the duration between R_{i_2} and R_{i_3}, or the duration between R_{i_1} and R_{i_3} will be even. Consider the subset $\mathcal{R}^{EV}$ of $\mathcal{R}$ of sequences of the form (1) where d is even. To prove that no node is in three round-sets, it suffices to prove that $\mathcal{R}^{EV}$ is empty.

We assume that $\mathcal{R}^{EV}$ is non-empty and derive a contradiction.

Let $\mathcal{R}^{EV}_{\hat{d}}$ be the subset of $\mathcal{R}^{EV}$ comprising sequences of minimum (even) duration $\hat{d}$, i.e.

$$\mathcal{R}^{EV}_{\hat{d}} = \{\underline{R} \in \mathcal{R}^{EV} \ \mid\ \forall\ \underline{R}' \in \mathcal{R}^{EV}.\ d(\underline{R}') \geq d(\underline{R}) = \hat{d}\} \tag{2}$$

Clearly, if $\mathcal{R}^{EV}$ is non-empty then so is $\mathcal{R}^{EV}_{\hat{d}}$. Let $\underline{R}^* \in \mathcal{R}^{EV}_{\hat{d}}$ be the sequence with earliest start-point $\hat{s}$, i.e.

$$\underline{R}^* = R_{\hat{s}}, \ldots, R_{\hat{s}+\hat{d}} \tag{3}$$

where

$$\forall\ \underline{R'} \in \mathcal{R}^{EV}_{\hat{d}}\ .\ s(\underline{R'}) \geq s(\underline{R^*}) = \hat{s} \tag{4}$$

By (1), there exists $g \in R_{\hat{s}} \cap R_{\hat{s}+\hat{d}}$. Choose node g' which sends a message to g in round $\hat{s}+\hat{d}$. As g' is a neighbour of g, either g' sends a message to g in round $\hat{s}$ or g sends a message to g' in round $\hat{s}+1$. We show that each of these cases leads to a contradiction.

Case (i): g' sends a message to g in round $\hat{s}$

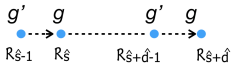


Figure 3: Node g' sends a message to node g in round $\hat{s}$: the first round of the minimum even length sequence (of length $\hat{d}$) in which g repeats

Refer to Figure 3. In this case, there must be a round $\hat{s}-1$ which is either round 0 and g' is the initial node, or g' received a message in round $\hat{s}-1$. Thus, the sequence

$$\underline{R}^{*'} = R_{\hat{s}-1}, R_{\hat{s}}, \ldots, R_{\hat{s}+\hat{d}-1} \quad \textit{where } g' \in R_{\hat{s}-1} \cap R_{\hat{s}+\hat{d}-1} \tag{5}$$

has $d(\underline{R}^{*'}) = (\hat{s}+\hat{d}-1)-(\hat{s}-1) = \hat{d}$ which is even and so $\underline{R}^{*'} \in \mathcal{R}^{EV}_{\hat{d}}$. As $\underline{R}^{*'} \in \mathcal{R}^{EV}_{\hat{d}}$, by (4)

$$s(\underline{R}^{*'}) \geq s(\underline{R^*}) \tag{6}$$

But, from (5), $s(\underline{R}^{*'}) = \hat{s}-1$ and, from (4), $s(\underline{R^*}) = \hat{s}$. Thus, by (6),

$$\hat{s}-1 = s(\underline{R}^{*'}) \geq s(\underline{R^*}) = \hat{s}$$

which is a contradiction.

Case (ii): g sends a message to g' in round $\hat{s}+1$

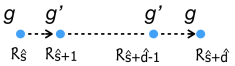


Figure 4: Node g sends a message to node g' in round $\hat{s}+1$: round $\hat{s}$ is the first round of the minimum even length sequence (of length $\hat{d}$) in which g repeats

Refer to Figure 4. By the definition of $\mathcal{R}^{EV}$, the smallest possible value of $\hat{d}$ is 2. However, it is not possible to have $\hat{d} = 2$ in this case as then $\underline{R}^* = R_{\hat{s}}, R_{\hat{s}+1}, R_{\hat{s}+2}$. This would mean that g sends a message to g' in round $\hat{s}+1$. But, we chose g' to be such that g' sends a message to g in round $\hat{s}+\hat{d} = \hat{s}+2$. This cannot happen as g cannot send a message to g' and g' to g in consecutive rounds by the definition of rounds. Thus,

$$\underline{R}^* = R_{\hat{s}}, R_{\hat{s}+1}, \ldots, R_{\hat{s}+\hat{d}-1}, R_{\hat{s}+\hat{d}} \text{ where } \hat{s}+1 < \hat{s}+\hat{d}-1$$

Consider the sequence

$$\underline{R}^{*''} = R_{\hat{s}+1}, \ldots, R_{\hat{s}+\hat{d}-1} \tag{7}$$

As g' receives a message from g in round $\hat{s}+1$ and g' sends a message to g in round $\hat{s}+\hat{d}$, it is clear that $g' \in R_{\hat{s}+1} \cap R_{\hat{s}+\hat{d}-1}$. Thus, $\underline{R}^{*''} \in \mathcal{R}$. As $\hat{d}$ is even, so is $(\hat{s}+\hat{d}-1)-(\hat{s}+1) = \hat{d}-2$ and therefore $\underline{R}^{*''} \in \mathcal{R}^{EV}$. Now, $\underline{R}^* \in \mathcal{R}^{EV}_{\hat{d}}$ and so, as $\underline{R}^{*''} \in \mathcal{R}^{EV}$, we have, by (2),

$$d(\underline{R}^{*''}) \geq d(\underline{R}^*) \tag{8}$$

As $d(\underline{R}^{*''}) = \hat{d}-2$ from (7) and $d(\underline{R}^*) = \hat{d}$ from (3), we have, by (8),

$$\hat{d}-2 = d(\underline{R}^{*''}) \geq d(\underline{R}^*) = \hat{d}$$

This contradiction completes the proof.

□

Theorem 2.2 implies that $R_i = \emptyset$ for $i \geq 2n$, where n is the number of vertices of G. Therefore amnesiac flooding always terminates. The corollary below gives a loose bound on termination time (tighter ones follow later):

Corollary 2.2.1. *Synchronous amnesiac flooding always terminates in fewer than* $2n+1$ *rounds.*

3 Termination Time

We have previously seen that Amnesiac Flooding on any synchronous network will visit every node at most twice and thus, terminate, in at most $2n+1$ rounds. However, we can find much better termination times. This maybe a good time to remind us of classic flooding and its termination time. The classic stateful flooding algorithm maybe summarised as follows:

From a source s, send a message to all its neighbours. Every receiving node forwards the message to all its neighbours, or all neighbours having not just received the message from. If the message is received again, ignore the message and do not forward.
This algorithm will achieve broadcast with termination time in optimal time i.e. in $eccentricity(s)$ (plus potentially one additional ruound) time where s is the initiator of the message. Time optimality is easy to see since the farthest node cannot receive the message before $eccentricity(s)$ rounds.

3.1 Almost Trivial Termination on Bipartite Graphs

As it turns out, proving termination and showing that this time matches the optimal is almost trivial for bipartite graphs. It is easy to see that amnesiac flooding is a bit like doing BFS exploration and one can now think of this exploration in terms of a traversal tree (Figure 5). The exploration happens level wise where each level is explored simultaneously. However, since the graph is bipartite, there are no cross-edges and by the amnesiac flooding property, a message cannot go back up to the previous level. Thus, the messages can only go down to the next level with that whole level getting explored simultaneously. Thus, amnesiac flooding will terminate in the number of levels, which is the same as eccentricity of the initiator node. Thus, amnesiac flooding not only terminates on bipartite graphs, it does so in optimal time. Note that the smallest eccentricity in a graph is known as the *radius* and the largest is the *diameter* and diameter is never more than twice the radius. The following Theorem follows:

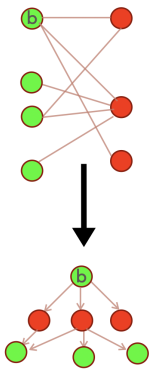


Figure 5: Amnesiac Flooding in a bipartite graph starting from a node b.

Theorem 3.1. *In a bipartite graph G, amnesiac flooding initiated from a node g_0 terminates in rounds = $e(g_0)$, where $e(g_0)$ is the eccentricity of the vertex g_0 in G.*

3.2 Termination on Non-Bipartite Graphs

The situation is a lot more complicated for non-bipartite graphs. Thinking in terms of the traversal, there are now cross edges and since there's no memory, messages can easily traverse back to the parent level from an already traversed level. Hussak and Trehan [7, 9] present and prove the termination bounds (Theorem 3.2). They do so by direct arguments about amnesiac flooding and by defining *ec nodes* (equidistantly connected nodes). *ec nodes* are two neighbouring nodes equidistant from an initiator node and they exist if and only if the graph is non-bipartite. Turau [12] proves similar bounds by constructing a bipartite *auxillary graph* for every non-bipartite graph (by connecting two copies of the original graph that have had their cross edges removed).

Theorem 3.2. *([7], Theorem 12) Let G be a non-bipartite graph with diameter d and let $g_0 \in G$ be an initial node of eccentricity e. Then, amnesiac flooding terminates in j rounds where j is in the range $e < j \leq e + d + 1$.*

Combining Theorems 3.1 and 3.2, and from their proofs, we get the following powerful general theorem:

Theorem 3.3. *Let G be a graph with diameter d and let $g_0 \in G$ be an initial node of eccentricity e. Then, amnesiac flooding terminates in rounds = $e(g_0)$, where $e(g_0)$ is the eccentricity of the vertex g_0 in G, if and only if G is bipartite, otherwise (if and only if the graph is non-bipartite), amnesiac flooding terminates in j rounds where j is in the range $e < j \leq e + d + 1$.*

Given that eccentricity is upper bounded by the diameter of the graph, Corollary 3.3.1 follows:

Corollary 3.3.1. *Let G be a graph with diameter d. Amnesiac flooding initiated from any initiator node on G terminates in at most d rounds if G is bipartite, and in at most $2d + 1$ rounds, if G is non-bipartite.*

One can easily verify that the bounds in Theorem 3.3 are tight: AF in the symmetric bipartite hypercube graph (Figure 1) terminates from any initiator vertex in diameter (equal to eccentricity) rounds, and in the symmetric non-bipartite graphs Triangle (Figure 2) and Petersen graph (Figure 1) in twice the diameter ((equal to eccentricity)) plus 1 rounds. However, when node eccentricity is less than the diameter, on the bipartite line graph (Figure 2, with node b as initiator), AF terminates in eccentricity (less than diameter) time. For non-bipartite graphs, one can construct graphs and choose initiator nodes such that amnesiac flooding can terminate in, for any chosen j, in the range $e < j \leq e + d + 1$. For example, in Figure 6, AF terminates in only $e(= 5) + 1$ rounds when begun from node a

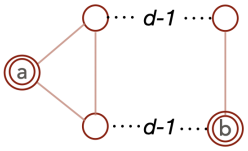


Figure 6: Amnesiac Flooding on graph G begun from node a terminates in $e + 1$ rounds, whereas from node b it terminates in $e+d+1$ rounds where $e = e(a) = e(b)$ is the eccentricity of the nodes and d is the diameter of G.

but in $e(= 5) + d(= 5) + 1$ rounds when begun from node b. The tight nature of the if and only if result of Theorem 3.3 raises the intriguing question of whether there maybe stateless/semi-stateless algorithms that can be used for testing bipartiteness/diameter approximation using a run of amnesiac flooding. However, the dependence on eccentricity (rather than diameter) suggests this maybe challenging.

3.3 Multiple Starters and Multiple Floodings

Though the results in [7] are stated for a single starter, it also states that these can be extended to multiple starters (due to the nature of the proofs) and this is done in the journal version of the paper [9] by an elegant adaptation and restatement of the property of bipartiteness in the context of multiple starters. The proofs and results then follow in a straightforward manner. Extending the definitions of *ec nodes* and *eccentricity* to a set I of starters (from a singleton node), we proposed the notion of $I - Bipartiteness$:

Definition 3.1. *The graph (G, E) is I-bipartite iff (G, E) has no ec-nodes. Equivalently, (G, E) is I-bipartite iff the quotient graph of (G, E), in which the nodes of I are contracted to a single node, is bipartite.*

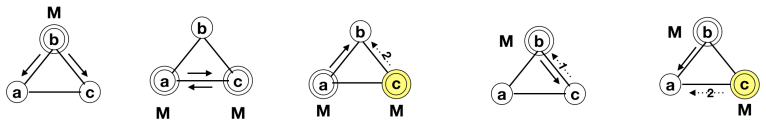
This leads to a restatement of Theorem 3.3 as follows (Theorem 3.4).

Theorem 3.4. *Let $G(V, E)$ be a graph with diameter d and let $I \subseteq V$ be a set of initial nodes. Then, amnesiac flooding terminates in rounds $= e(I)$ if and only if G is bipartite, and otherwise in j rounds where j is in the range $e(I) < j \leq e(I)+d+1$, where $e(I)$ is the eccentricity of the set I (defined as $e(I) = max\{distance(I, g) : g \in V\}$) and d is the diameter of G.*

This result is also tight, and a general condition for meeting the $e(I) + 1$ lower bound on non-bipartite graphs is shown in [9].

Potentially, from a more practical consideration, the behaviour of amnesiac flooding and other related floodings when there are multiple messages of interest in the network, is of interest. This is more so if we are in the *CONGEST* model where messages are limited to (poly)logarithmic size. Assuming each message is of logarithmic size, one can easily execute a constant number of amnesiac floodings in a parallel, stateless manner with all the nice properties. In general, there could be different rules and floodings (distinct from amnesiac flooding) which could be executed. Hussak and Trehan discuss some variants and their termination in [8]. Maybe the latest results on uniqueness (Section 6) could have some bearing on that discussion.

4 Asynchronous Amnesiac Flooding



(a) Round 1 (b) Round 2 (c) Round 3 (d) Round 4 (e) Round 5

Figure 7: Asynchronous *AF* over a Triangle with an adaptive scheduler. Both node *a* and *c* send *M* to each other in round 2. In round 3, *a* sends *M* to *b* but the adversary makes *c* hold the message for one round (shaded node). In the next round, we have a round analogous to round 2 and so on.

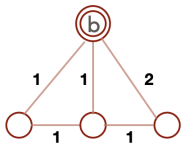


Figure 8: Non-termination of *AF* under a fixed delay on a single edge. The numbers on the edges give the number of rounds required for delivery.

Assuming a completely asynchronous model with completely local independent clocks and arbitrary delivery times, a question arises right off the bat: *For a node, when can one say that messages from two distinct neighbours have been received simultaneously?*. Note that this question is central to executing *AF* - since, in *AF*, the node floods to the other neighbours. This and other considerations led

us to propose the *Round Asynchronous model* as one simplified model to posit amnesiac flooding in [6, 7, 9].

Definition 4.1. ***Round-Asynchronous Model.*** *Computation proceeds in rounds where each round consists of nodes receiving messages scheduled to be delivered to it in that round, does local processing, and outputs (possibly different) messages to its neighbours. For every message, a* scheduling adversary *decides in which future (but unknown to the nodes) round the message is delivered to its destination. The adversary could be* adaptive *i.e. it can decide a delivery time for every message, or* fixed *i.e. for every edge, it decides on a fixed delay for that edge which does not change during the execution of the algorithm.*

For the stronger adaptive model, Figure 7 shows an execution of AF under an adversary strategy that leads to non-terminating broadcast. Remarkably, under the fixed adversary, we found that a single fixed delay on a single edge can lead to non-terminating broadcast as shown in Figure 8.

5 Dynamism and Fault-Sensitivity

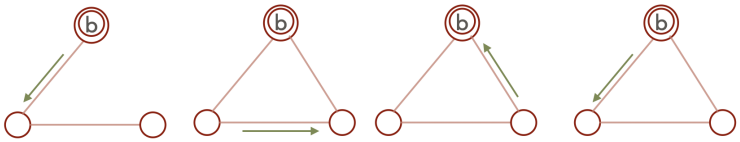


Figure 9: Amnesiac Flooding not terminating on insertion of an edge

In the previous section (Section 4), we saw (in the asynchronous setting), the delay of a single time step on a single edge can lead to non-termination. In a similar flavour, in [3–5], Austin, Gadouleau, Mertzios, and Trehan show that dropping even a single message on a single edge can lead to non-termination in certain cases. This work tries to delve deeper into the structural properties of amnesiac flooding in a formal manner and discovers the sensitivity of amnesiac flooding. Besides the just mentioned single message failure non-termination, the paper also shows potential non-termination in case of a single uni-directional link failure, or a weak byzantine failure (as defined in the paper). Combined with the easy observation that on the family of directed graphs, a uni-directional ring is an obvious topology for non-termination, one can see that amnesiac flooding can be very sensitive. This may not be the full story, however, since one can conjecture that another or a larger number of faults may actually restore terminating broadcast.

Earlier versions [8] have also studied dynamism in the form of node and edge deletion and insertion and conclude that Amnesiac Flooding is stable i.e. continues to do terminating broadcast in case of node and edge deletions (potentially requiring connectivity for broadcast) but cannot guarantee the same in case of edge or node insertions (Figure 9).

6 Amnesiac Flooding is Unique

Uniqueness may be a lightly used English word but Austin, Gadouleau, Mertzios, and Trehan [3–5] show that amnesiac flooding is unique in a formal technical sense: *Amnesiac Flooding is the only algorithm that can solve terminating broadcast under certain reasonable conditions*. Moreover, relaxing any of these conditions allows more algorithms to solve the problem. We believe that even this statement is unique - at least, we are not aware of results in the field of Algorithms design that can state that a single (or even a finite number of algorithms) can solve a particular problem. In fact, one would need to devise a reasonable definition of uniqueness to make such a statement - we do have an easier way to do so in our setting as will be seen shortly.

The formal statement is as follows (from [3–5]):

Theorem 6.1 (Uniqueness of Amnesiac Flooding)**.** *Any terminating broadcast algorithm possessing all of* Strict Statelessness*,* Obliviousness*,* Determinism *and* Unit Bandwidth *behaves identically to Amnesiac Flooding on all graphs under all valid labellings for all source nodes.*

Elaborating, the four conditions are as follows:

1. *Strict Statelessness*: Nodes maintain no information other than their port labellings between rounds. This includes whether or not they were in the initiator set.
2. *Obliviousness*: Routing decisions may not depend on the contents of received messages.
3. *Determinism*: All decisions made by a node must be deterministic.
4. *Unit Bandwidth*: Each node may send at most one message per edge per round.

Violation of even one of the above conditions allows us to propose other terminating broadcast algorithms. For example, a randomised algorithm which chooses between amnesiac flooding and flooding to every neighbour at every occasion of flooding, will eventually achieve terminating broadcast.

In a general setting, potentially one of the first hurdles one can imagine is how to rule out trivial extensions of any algorithm such as delaying execution of a command by a single wait statement, for example. Fortunately, in our setting of true statelessness, this does not seem possible, because delaying execution would necessitate use of memory and state to remember the command that is delayed. Beyond that, proving the result was technically challenging and we hope, this is a result and a direction which the research community will find interesting.

7 Conclusions

Amnesiac Flooding is a stateless version of flooding which was introduced in 2019. The synchronous version of this was shown to terminate in optimal number of rounds in bipartite graphs and in at most twice longer time in non-bipartite graphs. In fact, it was subsequently shown to terminate from any number of starters (simultaneous or non-simultaneous) with the termination time partitioning the graphs on the basis of an extension of bipartiteness called I-bipartiteness. A number of other interesting results, variants and related questions have been proposed in recent times including broader results on memory requirements for broadcast algorithms [10]. A most recent and exciting result, in our opinion, is that amnesiac flooding is the only stateless deterministic oblivious (to message content) algorithm that can achieve terminating broadcast. Amnesiac flooding is thus, technically, a unique algorithm. This poses the question of quantification of algorithmic solutions to particular problems.

References

[1] Amnesiac flooding, October 2025. https://en.wikipedia.org/wiki/Amnesiac_flooding.

[2] James Aspnes. Flooding, February 2019. http://www.cs.yale.edu/homes/aspnes/pinewiki/Flooding.html.

[3] Henry Austin, Maximilien Gadouleau, George B. Mertzios, and Amitabh Trehan. Amnesiac Flooding: Easy to Break, Hard to Escape. In Dariusz R. Kowalski, editor, *39th International Symposium on Distributed Computing (DISC 2025)*, volume 356 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:23, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2025.10, doi:10.4230/LIPIcs.DISC.2025.10.

[4] Henry Austin, Maximilien Gadouleau, George B. Mertzios, and Amitabh Trehan. Amnesiac flooding: Easy to break, hard to escape. *CoRR*, abs/2502.06001, 2025. URL: https://doi.org/10.48550/arXiv.2502.06001, arXiv:2502.06001, doi:10.48550/ARXIV.2502.06001.

[5] Henry Austin, Maximilien Gadouleau, George B. Mertzios, and Amitabh Trehan. Brief announcement: Amnesiac flooding: Easy to break, difficult to escape. In Alkida Balliu and Fabian Kuhn, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2025, Hotel Las Brisas Huatulco, Huatulco, Mexico, June 16-20, 2025*, pages 541–544. ACM, 2025. doi:10.1145/3732772.3733523.

[6] Walter Hussak and Amitabh Trehan. On termination of a flooding process. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019.*, pages 153–155. ACM, 2019. doi:10.1145/3293611.3331586.

[7] Walter Hussak and Amitabh Trehan. On the termination of flooding. In Christophe Paul and Markus Bläser, editors, *37th International Symposium on Theoretical Aspects of Computer Science, STACS 2020, March 10-13, 2020, Montpellier, France*, volume 154 of *LIPIcs*, pages 17:1–17:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.STACS.2020.17.

[8] Walter Hussak and Amitabh Trehan. Terminating cases of flooding. *CoRR*, abs/2009.05776, 2020. URL: https://arxiv.org/abs/2009.05776, arXiv:2009.05776.

[9] Walter Hussak and Amitabh Trehan. Termination of amnesiac flooding. *Distributed Comput.*, 36(2):193–207, 2023. URL: https://doi.org/10.1007/s00446-023-00448-y, doi:10.1007/S00446-023-00448-Y.

[10] Garrett Parzych and Joshua J. Daymude. Memory Lower Bounds and Impossibility Results for Anonymous Dynamic Broadcast. In Dan Alistarh, editor, *38th International Symposium on Distributed Computing (DISC 2024)*, volume 319 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 35:1–35:18, Dagstuhl, Germany, 2024. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. URL: https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.DISC.2024.35, doi:10.4230/LIPIcs.DISC.2024.35.

[11] David Peleg. Time-optimal leader election in general networks. *J. Parallel Distrib. Comput.*, 8(1):96–99, January 1990. doi:10.1016/0743-7315(90)90074-Y.

[12] Volker Turau. Amnesiac flooding: synchronous stateless information dissemination. In *SOFSEM 2021: Theory and Practice of Computer Science: 47th International Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2021, Bolzano-Bozen, Italy, January 25–29, 2021, Proceedings 47*, pages 59–73. Springer, 2021.

THE FORMAL LANGUAGE THEORY COLUMN

by

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano
20133 Milano, Italy
pighizzini@di.unimi.it

The contribution by Antonio Casares presented in this issue of BEATCS studies the placement of the acceptance condition on automata over infinite words and discusses how the community is currently experiencing a shift from using state-based acceptance to using transition-based acceptance, explaining why this shift is taking place. Maybe in the near future, for the reasons discussed in this survey, transition-based omega-automata will become the default model in the community. The survey is derived from the final chapter of Antonio's PhD Thesis.

BEATCS no 147

TRANSITION-BASED VS STATED-BASED ACCEPTANCE FOR AUTOMATA OVER INFINITE WORDS

Antonio Casares*

Abstract

Automata over infinite objects are a well-established model with applications in logic and formal verification. Traditionally, acceptance in such automata is defined based on the set of states visited infinitely often during a run. However, there is a growing trend towards defining acceptance based on transitions rather than states.

In this survey, we analyse the reasons for this shift and advocate using transition-based acceptance in the context of automata over infinite words. We present a collection of problems where the choice of formalism has a major impact and discuss the causes of these differences.

Contents

*University of Warsaw, Poland. Email: antoniocasaressantos@gmail.com
This work was supported by the Polish National Science Centre (NCN) grant "Polynomial finite state computation" (2022/46/A/ST6/00072).

1 Introduction

Automata theory is a central and long-established topic in computer science. The definition of finite automata has barely suffered any modification since the introduction of non-deterministic automata by Rabin and Scott [RS59]. However, the generalisation of automata to infinite words presents less stable definitions, as different modes of acceptance are best suited to different situations. Recently, there has been a shift in the community towards using transitions instead of states to encode the acceptance condition of ω-automata. In this survey, we analyse the reasons for this shift and advocate using transition-based acceptance in the context of automata over infinite words.

Automata over infinite words. An *automaton* over an input alphabet Σ is given by

- a finite set of states Q,
- a set of transitions $\Delta \subseteq Q \times \Sigma \times Q$,
- a set of initial states $Q_{\mathsf{init}} \subseteq Q$, and
- an acceptance condition.

A *run* over a (finite or infinite) word w is a path in the automaton starting in Q_{init} and with transitions labelled by the letters of w. The acceptance condition is thus a representation of the set of paths that are accepting.

If the automaton works over finite words, the acceptance condition usually takes the form of a subset of final states: a run is accepting if it ends in one of them (see Section 5 for further discussions on finite words). For automata over infinite words the situation is more complicated. Several acceptance conditions are commonly used, but they differ in expressive power and the complexity of related problems (see for instance [Bok18]). The main focus of this paper is the following dichotomy: Should we use states or transitions to encode the acceptance condition of automata over infinite words? More formally, we will consider acceptance conditions of one of the following forms.

A *state-based acceptance condition* is a language $Acc \subseteq Q^\omega$. A *transition-based acceptance condition* is a language $Acc \subseteq \Delta^\omega$.Usually, we represent them via a finite set of colours C, a colouring function $\gamma\colon Q \to C$ (resp. $\gamma\colon \Delta \to C$) and a language $Acc' \subseteq C^\omega$. That is, we see automata as transducers $\Sigma^\omega \to C^\omega$, and the acceptance condition is given by a subset of the image. Two languages that are commonly used as acceptance conditions are:

- Buchi = $\{w \in \{-, \bullet\}^\omega \mid w \text{ contains } \bullet \text{ infinitely often}\}$. We may refer to states (resp. transitions) coloured with $\bullet$ as *accepting*.
- coBuchi = $\{w \in \{-, \text{✗}\}^\omega \mid w \text{ contains } \text{✗} \text{ finitely often}\}$.

We show examples of Büchi automata in Figure 1.

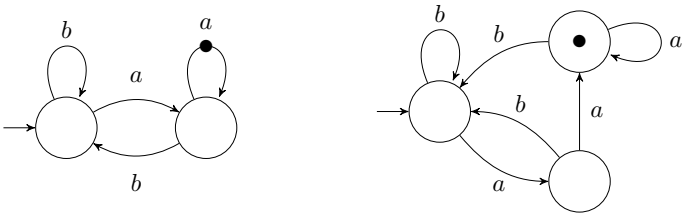


Figure 1: Two Büchi automata recognising the language of words containing infinitely many factors '*aa*'. The automaton on the left uses transition-based acceptance, while the automaton on the right is state-based.

The origins. Automata over infinite words were first introduced by Büchi in the 60s [Büc62], using a formalism that put the acceptance condition over states.[1] The tradition of employing state-based acceptance persisted in all subsequent classic foundational works on ω-automata: Muller's paper at the origin of the Muller condition [Mul63], Landweber's study of the complexity of ω-regular languages [Lan69], McNaughton's works on ω-regular expressions [McN66] and infinite games [McN93], Rabin's decidability result of S2S [Rab69], Wagner's paper introducing a hierarchy of complexity [Wag79], etc. Following this tradition, virtually all handbooks and surveys about automata on infinite objects use state-based acceptance [Eil74, Tho90, Tho97, GTW02, PP04, BK08, Kup18, BCJ18, WS21, Löd21, EB23]. To the best of our knowledge, the only exceptions are the recent book *Games on Graphs* edited by Fijalkow [Fij+25], and the book *An Automata Toolbox* by Bojańczyk [Boj].

The rise of transition-based acceptance. Automata with "effects" on transitions, such as sequential transducers[2] [Sha48, Sect.8][Mea55, Sch61a]

[1] Corroborating this claim can be quite challenging. The use of state-based acceptance can be observed, for instance, in the first line of the proof of Lemma 12 (page 8). In Büchi's 1969 paper with Landweber [BL69a], this is a bit simpler to appreciate in the definitions of Sup$Z$ and U, in the second page of the paper.

[2] Transducers with output on states were also considered by Moore [Moo56]. However, the model with output on transitions popularized by Mealy [Mea55] rapidly became the norm.

or weighted automata [Sch61b] have been considered since the beginnings of automata theory. Transition-based ω-automata made their first, though modest, appearance in the mid-80s. To the best of our knowledge, their first occurrences were in Michel's work on the connection between *Linear Temporal Logic* (LTL) and automata [Mic84], and in Kurshan's paper on the complementation of deterministic Büchi automata [Kur87].[3] In the early 90s, Le Saëc made more systematic use of this model [Saë90, SPW91, VSL95]. He reintroduced transition-based Muller automata under the name of table-transition automata, and characterised which languages admit a unique *morphism-minimal* Muller automaton: those that can be recognised by a Muller automaton with one state per residual of the language [VSL95, Cor. 5.15]. This characterisation no longer holds for state-based automata (see Example 10 for an illustration on how the previous property is sensitive to the placement of the acceptance condition). Despite the works of Le Saëc, transition-based automata were used only scarcely in the following years.

Some notable exceptions to the predominant use of state-based automata in the 2000s are given by a series of works concerning the translation of LTL formulas to automata. In 1999, Couvreur proposed a translation using transition-based generalised Büchi automata [Cou99]. A similar algorithm was the base for the tool `ltl2ba` by Gastin and Oddoux [GO01] (the importance of the use of transition-based automata in this work is discussed in [GL02]). The use of transition-based acceptance in this subarea was further fostered by the tool Spot [DP04, Dur+22], influenced by Couvreur's approach. More recently, transition-based automata have been adopted in the HOA format [Bab+15], and it is the primary model in other tools such as Owl [KMS18] or `ltl3tela` [Maj+19]. We refer to [Dur07, pages 66-67] for an overview of the use of state-based and transition-based approaches to the translation of LTL prior to 2007.

A turning point occurred in 2019, as Abu Radi and Kupferman proved that transition-based history-deterministic coBüchi automata can be minimised in polynomial time [AK19], while Schewe showed that the corresponding problem is NP-complete for state-based automata [Sch20]. Since then, there is an increasing interest for transition-based ω-automata, and, as discussed in Sections 3 and 4, many recent results rely on the use of this model.

Why was the use of state-based acceptance widespread? We may wonder why state-based automata were the ubiquitous model for more than 50 years. Probably the most influential factor is that ω-automata generalise automata over finite words, for which acceptance over states is a natural choice. Some construc-

[3] The possibility of using transition-based acceptance was previously suggested in [Par81, Section 8.2]. Some sources [Red99] mention that transition-based acceptance was already suggested by Redziejowski in 1972 [Red72]; unfortunately we could not get access to this paper.

tions of ω-automata build on automata over finite words, and for some of these, state-based acceptance appears naturally.

One such example is the characterisation of languages recognised by deterministic Büchi automata as limits of languages of finite words [Lan69]. A language $L \subseteq \Sigma^\omega$ can be recognised by a deterministic Büchi automaton if and only for some regular language of finite words $L_{\mathsf{fin}} \subseteq \Sigma^*$ we have:

$$L = \overrightarrow{L_{\mathsf{fin}}} = \{w \in \Sigma^\omega \mid w \text{ contains infinitely many prefixes in } L_{\mathsf{fin}}\}.$$

Building a state-based Büchi automaton from a deterministic automaton recognising L_{fin} is easy: we just need to interpret the final states of the automaton as accepting Büchi states. The converse direction follows similarly.

Structure of the survey. We start by showing in Section 2 that we can switch between state and transition-based acceptance with at most a linear blow-up. However, we already notice a key difference: going from a state-based automaton to a transition-based one does not require adding any additional state, while deciding the minimal number of states required to perform the converse transformation is NP-hard (Proposition 3). In Sections 3 and 4, we study problems on ω-automata and games where the choice between transition-based and state-based acceptance may strongly affect the complexity of a given problem. In Section 5 we explore transition-based acceptance for automata over finite words. Finally, in Section 6 we discuss some of the reasons causing the striking differences between the two models.

Definitions are introduced progressively as needed. The reader may use the hyperlinks on technical terms to quickly see their definition.

2 From states to transitions and vice versa

At first sight, it could seem that there is no great difference between state-based or transition-based acceptance: we can go from one model to the other with at most a linear blow-up. However, transition-based automata are always smaller, and going from a state-based automaton to a transition-based one in an optimal way is NP-hard, as stated in Proposition 3.

Proposition 1. *Every state-based automaton can be relabelled with an equivalent transition-based acceptance condition.*

Proof. Let $Acc \subseteq Q^\omega$ be the acceptance condition of the automaton, and let $\gamma\colon \Delta \to Q$ be the function assigning to each transition (q, a, q') its source state q. Then, (γ, Acc) is an equivalent transition-based acceptance condition. □

In general, we cannot relabel in a similar manner a transition-based automaton to obtain an equivalent state-based one. We can, however, build an equivalent state-based automaton paying a small blow-up on the number of states.

Proposition 2. *Every transition-based automaton admits an equivalent state-based automaton with at most* $|Q||\Delta| + |Q_{\mathsf{init}}|$ *states.*

Proof. Let $\mathcal{A}$ be a transition-based automaton with acceptance $Acc \subseteq \Delta^\omega$. We define the automaton having:

- States: $(Q \times \Delta) \cup Q_{\mathsf{init}}$.
- Transitions: For every transition $t' = q \xrightarrow{a} q'$ in $\mathcal{A}$, we let $(q,t) \xrightarrow{a} (q',t')$, and $q \xrightarrow{a} (q',t')$ if $q \in Q_{\mathsf{init}}$.
- Initial states: Q_{init}.
- Acceptance condition: We define $\gamma\colon Q \to \Delta \cup \{x\}$ by: $\gamma(q,t) = t$ and $\gamma(q_0) = x$ if $q_0 \in Q_{\mathsf{init}}$. The acceptance condition is given by the colouring γ and the language $x \cdot Acc$.

It is immediate to check that the obtained automaton is equivalent to $\mathcal{A}$. □

In both proofs above, the obtained automaton is not only equivalent to the original one, but there is a bijection between the runs of both. We formalise this idea with the notion of *locally bijective morphisms* [Cas+24, Def.3.3].

Given two automata $\mathcal{A}, \mathcal{A}'$ over the same alphabet, a *locally bijective morphism* is a function $\varphi\colon Q \to Q'$ such that:

- $\varphi(Q_{\mathsf{init}}) = Q'_{\mathsf{init}}$,
- for all $(q,a,q') \in \Delta$, $(\varphi(q), a, \varphi(q')) \in \Delta'$,
- for all $(p,a,p') \in \Delta'$ and $q \in \varphi^{-1}(p)$, there is $q' \in \varphi^{-1}(p')$ such that $(q,a,q') \in \Delta$, and
- a run ρ in $\mathcal{A}$ is accepting if and only if $\varphi(\rho)$ is accepting in $\mathcal{A}'$.

Intuitively, if $\varphi\colon \mathcal{A} \to \mathcal{A}'$ is a locally bijective morphism, it means that $\mathcal{A}$ has been obtained from $\mathcal{A}'$ by duplicating some of its states, for instance, via a product construction. For example, the automaton on the right of Figure 1 admits a locally bijective morphism to the automaton on its left.

Proposition 1 implies that for every state-based automaton there is a transition-based automaton of same size admiting a locally bijective morphism to it (the automaton itself). However, in the other direction, deciding whether there is a small state-based automaton admiting a locally bijective morphism towards a given transition-based automaton is hard, already for Büchi automata.

Proposition 3. *The following problem is* NP-complete*:*

Input: *A transition-based Büchi automaton $\mathcal{A}_{\text{tr}}$ and a positive integer n.*
Question: *Is there a state-based Büchi automaton with n states admitting a locally bijective morphism to $\mathcal{A}_{\text{tr}}$?*

Proof. To show NP-hardness, we use the reduction from Vertex Cover given by Schewe to show the NP-completeness of the minimisation of state-based deterministic Büchi automata [Sch10].

Let $G = (V, E)$ be an undirected graph. Consider the Büchi automaton $\mathcal{A}_G$ over the alphabet $\Sigma = V$ with states $Q_G = V$, all of them initial, and transitions $u \xrightarrow{v} v$ for every $(u, v) \in E$, and for $u = v$. For the Buchi condition, all transitions are accepting except the self-loops $v \xrightarrow{v} v$. This automaton recognises the paths in G, allowing repetition of vertices, but that visit at least two different vertices infinitely often.

Let k be the size of a minimal vertex cover of G. We claim that there is a state-based Büchi automaton with $|V|+k$ states admitting a locally bijective morphism to $\mathcal{A}_G$, and that this is optimal. To obtain such a state-based automaton, we duplicate every state v that is part of a given vertex cover. Let $v_\bullet, v_-$ be the two copies of this state, and set $v_\bullet$ to be an accepting state. Among non-duplicated states, transitions are as in $\mathcal{A}_G$. For duplicated states, we let $v_i \xrightarrow{v} v_-$ for $i \in \{-, \bullet\}$ and $u_i \xrightarrow{v} v_\bullet$ for $(u, v) \in E$. It is easy to chech that $\varphi(v_i) = v$ defines a locally bijective morphism.

For the converse direction, let $\mathcal{A}$ be a state-based Büchi automaton and $\varphi\colon \mathcal{A} \to \mathcal{A}_G$ a locally bijective morphism. For every state v in $\mathcal{A}_G$, $\varphi^{-1}(v)$ must contain a non-accepting state, as a run ending in v^ω is rejecting in $\mathcal{A}_G$. We claim that the set of vertices such that $\varphi^{-1}(v)$ contains an accepting state is a vertex cover of G. Indeed, for every edge $(u, v) \in E$, a word ending in $(uv)^\omega$ is accepting in $\mathcal{A}_G$, therefore, either $\varphi^{-1}(u)$ or $\varphi^{-1}(v)$ contains an accepting state.

The problem is in NP, as there is always such an automaton with $2|Q|$ states. For $n < 2|Q|$, it suffices to guess an automaton $\mathcal{A}_{\text{st}}$ with n states and a locally bijective morphism $\varphi\colon \mathcal{A}_{\text{st}} \to \mathcal{A}_{\text{tr}}$. □

In our opinion, the above propositions indicate that state-based acceptance is often innapropriate. We believe that, in an ideal scenario, each state of a minimal automaton should stand for some semantic properties of the language they represent (in the case of automata over finite words, these are the residuals of the language). This cannot be the case for state-based ω-automata, as some states must be allocated to encode parts of the acceptance condition.

3 Minimisation and transformations of automata

In this section we study three problems relating to ω-automata: minimisation, conversion of acceptance condition and determinisation. We discuss how the use of transition-based or state-based acceptance can critically affect these problems.

3.1 Minimisation of coBüchi automata

The *minimisation problem* asks, given an automaton and a number n, whether there is an equivalent automaton with at most n states. This problem admits different variants, depending on the class of automata that constitutes the search space (here we assume that this class is the same for the input and output automata).

In 2010, Schewe showed that the minimisation problem is NP-hard for most types of deterministic state-based ω-automata, including Büchi, coBüchi or parity [Sch10]. It came as a surprise when Abu Radi and Kupferman showed that history-deterministic coBüchi automata can be minimised in polynomial time [AK22] (conference version from 2019 [AK19]). Soon after, Schewe showed that the same problem is NP-hard for state-based automata.[4]

An automaton is *history-deterministic* (abbreviated HD) if there is a resolver $\sigma\colon \Sigma^*\times\Sigma \to \Delta$, such that for every word w accepted by the automaton, the run over w built following the transitions given by σ is accepting. History-deterministic coBüchi automata are as expressive as deterministic ones, but they can be exponentially more succinct [KS15].

Proposition 4 ([AK22],[Sch20])**.** *History-deterministic transition-based coBüchi automata can be minimised in polynomial time.*

The minimisation problem for history-deterministic state-based coBüchi automata is NP-complete*.*

The work of Abu Radi and Kupferman provided the basis of many subsequent results, including new representations for ω-regular languages [ES22, Ehl25], minimisation of HD generalised coBüchi automata [Cas+25], passive learning of HD coBüchi automata [LW25] and characterisations of positional languages [CO24]. The transition-based assumption is essential to all these works.

Schewe's proof of NP-hardness of the minimisation of deterministic state-based Büchi automata [Sch10] strongly relies on putting the acceptance over states. In fact, as we have seen in Proposition 3, what this reduction shows is that finding a minimal state-based automaton that simulates a transition-based one is

[4] Note that the critical difference lies in the output class, as we can convert the input from state-based to transition-based in polynomial time.

NP-hard. It was not until 2025 that the minimisation of deterministic transition-based Büchi and coBüchi automata was shown to be NP-hard, requiring a highly technical proof [AE25].

3.2 Translation from Muller to parity

The complexity of the acceptance condition used by an automaton may greatly affect the computational cost of dealing with these automata. Namely, many problems are PSPACE-hard for Muller automata [HD05], but become tractable for parity automata [Cal+22, Bok18]. Therefore, an important task is to simplify the acceptance condition of a given automaton. In practice, this usually takes the following form: given an automaton using a Muller condition, build an equivalent automaton using a parity condition.

The parity and Muller conditions are defined as follows:

- $\mathsf{parity}(d) = \{w \in \{1, \dots, d\}^\omega \mid \liminf w \text{ is even}\}$.
- $\mathsf{Muller}(\mathcal{F}) = \{w \in C^\omega \mid \mathtt{Inf}(w) \in \mathcal{F}\}$, for $\mathcal{F} \subseteq \mathcal{P}(C)$ a family of subsets and $\mathtt{Inf}(w)$ the set of colours that appear infinitely often in w.

Recently, an optimal transformation has been introduced – based on a structure called the *Alternating Cycle Decomposition* (ACD) – transforming a Muller automaton $\mathcal{A}$ into a parity one [Cas+24]. Formally, it produces a transition-based parity automaton that admits a locally bijective morphism to $\mathcal{A}$ and with a minimal number of states among parity automata admiting such a morphism. This transformation can be performed in polynomial time provided that the ACD can be computed efficiently; this is the case for example if the acceptance condition of $\mathcal{A}$ is generalised Büchi, defined as follows:

- $\mathsf{genBuchi} = \{w \in \mathcal{P}(C)^\omega \mid \bigcup_{A \in \mathtt{Inf}(w)} A = C\}$.

Proposition 5 (Follows from [Cas+24, Thm. 5.35])**.** *Given a generalised Büchi automaton $\mathcal{A}$, we can build in polynomial time a transition-based Büchi automaton admiting a locally bijective morphism to $\mathcal{A}$ that has a minimal number of states among Büchi automata admitting locally bijective morphisms to $\mathcal{A}$.*

However, the optimality result of the ACD-transformation strongly relies on the use of transition-based acceptance in the output automaton, as the previous problem becomes NP-hard for state-based automata.

Proposition 6. *The following problem is* NP-complete*:*

Input: *A state-based generalised Büchi automaton $\mathcal{A}$ and an integer n.*
Question: *Is there a state-based Büchi automaton with n states admitting a locally bijective morphism to $\mathcal{A}$?*

Proof. We can use the same reduction as in the proof of Proposition 3 (which in turn comes from [Sch10]). Indeed, we can replace the transition-based Büchi condition of the automaton $\mathcal{A}_G$ by a state-based generalised Büchi condition. □

3.3 Determinisation of Büchi automata

The determinisation of Büchi automata is a fundamental problem in the theory of ω-automata, studied since the introduction of the model [Büc62]. The first asymptotically optimal determinisation construction is due to Safra [Saf88], which transforms a Büchi automaton into a deterministic Rabin one. In 1999, Redziejowski proposed a variant for building a transition-based automaton from a given ω-regular expression [Red99]. Later on, Piterman [Pit06] and Schewe [Sch09] further improved Safra's construction, reducing the number of states of the final automaton (see also [Red12]). Schewe's construction transforms a Büchi automaton of size n into a deterministic Rabin automaton of size at most $\mathsf{sizeDet}(n)$, which is naturally equipped with a transition-based acceptance condition (with $\mathsf{sizeDet}(n) = o((1.65n)^n)$). In 2009, Colcombet and Zdanowski [CZ09] showed that the Piterman-Schewe construction is tight (up to 0 states!) as we precise now.

Proposition 7 ([CZ09])**.** *There exists a family of Büchi automata $\mathcal{A}_n$ with n states, such that a minimal transition-based deterministic Rabin automaton equivalent to $\mathcal{A}_n$ has $\mathsf{sizeDet}(n)$ states.*

We can obtain a state-based automaton by augmenting the number of states, but doing so we no longer have a matching lower bound. No such tight bounds are known for the determinisation of Büchi automata towards state-based automata.

The complementation and determinisation problems for Büchi and generalised Büchi automata with transition-based acceptance were further studied by Varghese in his PhD Thesis [Var14]. In the works of Schewe and Varghese [SV12, SV14], they point out the suitability of transition-based acceptance for the study of transformations of automata.

4 Games on graphs and strategy complexity

A *game* is given by a directed graph $G = (V, E)$ with a partition of vertices into those controlled by a player Eve and those controlled by a player Adam, a initial vertex and a *winning condition* defined in the same way as the acceptance condition of automata (which can be state-based or transition-based). The players move

a token in turns producing an infinite path, and Eve wins if this path belongs to the winning condition.

An important concept with applications for the decidability of logics [BL69b, GH82] and verification [BCJ18] is that of strategy complexity: how complex is it to represent a winning strategy? The simplest kind of strategies are *positional* ones. A strategy is *positional* if it can be represented by a function $\sigma\colon V \to E$: when in a vertex v controlled by Eve, she plays the transition $\sigma(v)$. More generally, a strategy is said to use *finite-memory* if the choice at a given moment only depends on a finite amount of information from the past, or, said differently, it can be implemented by a finite automaton (we refer to [Fij+25, Section 1.5] for formal definitions).

As already noticed by Zielonka [Zie98], and as we will see next, strategy complexity is quite sensitive to the placement of the winning condition.

4.1 Bipositionality over infinite games

We say that a language $Win \subseteq C^\omega$ is *positional* if for every game with winning condition Win, if Eve has a winning strategy, she has a positional one. A language Win is *bipositional* if both Win and its complement are positional, or, said differently, if both Eve and Adam can play optimally using positional strategies. Depending on whether we consider games with transition-based or state-based winning condition, we will say accordingly *positional over transition/state-based games*.

A celebrated result in the area is the proof of bipositionality of parity languages [EJ91, Mos84]. In 2006, Colcombet and Niwiński proved that these are the only prefix-independent bipositional languages over infinite game graphs [CN06], establishing an elegant characterisation of bipositionality. As indicated in the title of their paper, this characterisation only holds for transition-based games.

Proposition 8 ([CN06])**.** *A prefix-independent language $Win \subseteq C^\omega$ is bipositional over transition-based games if and only if there is $d \in \mathbb{N}$ and a mapping $\phi\colon C \to \{1, \dots, d\}$ such that $w \in Win$ if and only if $\phi(w) \in \mathsf{parity}(d)$.*

Proposition 9 ([Zie98, Section 6])**.** *There is a prefix-independent language that is bipositional over totally-coloured state-based games, but is not equivalent to $\mathsf{parity}(d)$ for any d.*

Proof sketch. An example of such a language is

$$Win = \{w \in \{a, b\}^\omega \mid \text{ both } a \text{ and } b \text{ appear infinitely often in } w\}.$$

Intuitively, if Eve is in a vertex coloured a, she can follow a strategy leading to a vertex coloured b in a positional way (and vice-versa).

From Adam's point of view, if he can win, there are some vertices from which he can force to never produce 'a' or force to never produce 'b' (and this can be done positionally). Removing those vertices, we define a positional strategy recursively. (Note that this can also be done for transition-based games, in fact, from Adam's point of view, *Win* is a Rabin condition, which are positional.) □

The characterisation of bipositionality was generalised to all (not necessarily prefix-independent) languages in [CO24, Thm. 7.1]. A necessary condition for bipositionality is that the language should be recognised by a transition-based deterministic parity automaton with one state per residual of the language. This property is very sensitive to the placement of the acceptance condition, if suffices to consider the language Buchi that cannot be recognised by a state-based automaton with a single state. The next example shows another version of this.

Example 10. *Consider the language*

$$L = \{w \in \{a,b\}^\omega \mid \textit{if letter 'a' occurs in } w \textit{ then it appears infinitely often}\}.$$

This language has two residuals: $\varepsilon^{-1}L$ and $a^{-1}L$. It can be recognised by a transition-based parity automaton (even a Büchi automaton) with two states, as shown in Figure 2. One can check that it also satisfies the other conditions from [CO24, Thm. 7.1], so it is bipositional. However, it is not possible to recognise L with a state-based parity automaton with only 2 states.

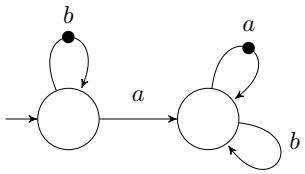


Figure 2: A Büchi automaton recognising the bipositional language of words that either contain no a, or infinitely many a's. This automaton has one state per residual of the language. A state-based parity automaton recognising this language must have at least 3 states.

4.2 Positionality via monotone graphs

Recently, Ohlmann characterised positionality by means of *monotone universal graphs* [Ohl23]. Not only this characterisation concerns positionality over

transition-based games, but the main notion of monotone graph radically uses the colouring on transitions. An ordered edge-coloured graph is *monotone* if whenever $v \xrightarrow{a} u$, $v \leq v'$ and $u' \leq u$, then the edge $v' \xrightarrow{a} u'$ also appears in the graph. Such kind of properties can only be naturally phrased in edge-coloured graphs.

Universal monotone graphs have been used to study the algorithmic complexity of solving different types of games on graphs, such as parity and mean-payoff [Col+22], and the above characterisation has been generalised to the memory of languages [CO25a].

4.3 The memory of ω-regular languages

The *memory* of a language *Win* is the minimal $m \in \mathbb{N}$ such that in any game with objective *Win*, if Eve has a winning strategy, she has one implemented by an automaton with at most m states. A result with major implications in logic is the fact that ω-regular languages have finite-memory [BL69b, GH82].

Recently, Casares and Ohlmann gave an effective way of computing the memory of ω-regular languages [CO25b], based on a characterisation using the notion of *ε-completable parity automata*. The definition of this notion is rooted in the use of transition-based acceptance: A parity automaton is *ε-completable* if for every pair of states q, q' and even colour x of the parity condition, we can either add a transition $q \xrightarrow{\varepsilon:x} q'$ or a transition $q' \xrightarrow{\varepsilon:x+1} q$ without modifying the language recognised by the automaton.

In 2023, Bouyer, Randour and Vandenhove showed that ω-regular languages are exactly those that are arena-independent finite-memory determined (that is, both Eve and Adam admit finite automata implementing strategies in every game with winning condition *Win*) [BRV23, Thm. 7]. The use of transition-based acceptance is key for the construction of a parity automaton recognising a language with the above property [BRV23, Section 5].

In 2021, Casares showed that the smallest automata that can be used for implementing winning strategies in every game using a given Muller language $\mathsf{Muller}(\mathcal{F})$ are exactly deterministic Rabin automata recognising $\mathsf{Muller}(\mathcal{F})$ [Cas22, Thm. 27]. In a related work, Casares, Colcombet and Lehtinen showed that the memory of $\mathsf{Muller}(\mathcal{F})$ coincides with the number of states of a minimal history-deterministic Rabin automaton recognising this language [CCL22, Thm. 5]. Both results only apply to transition-based Rabin automata.

5 What about finite words?

In light of the results above, one naturally wonders whether a shift to transition-based acceptance would also be beneficial for automata on finite words (*DFAs* in the following). Classical finite automata have a robust mathematical theory – notably, every regular language admits a canonical minimal DFA – and state-based acceptance is the undisputed preferred option for them. However, transition-based variants have been considered recently in works about synthesis of LTL over finite traces [Shi+20, Xia+21, Xia+24, Dur+25] and about translations of regular expressions over valuations of atomic propositions [MRD24].

Definition of acceptance. One option to define the acceptance of transition-based finite automata is simply to specify a set of final transitions: a run is accepting if its last transition belongs to this set.[5] More generally, following the definition of ω-automata used in this document, we define the *acceptance condition of a transition-based DFA* as a language $Acc \subseteq \Delta^*$: a run is accepting if it belongs to Acc. If Acc is a regular language, such an automaton accepts a regular language (we can convert it into a classical DFA by a product construction). Using a colouring function $\gamma\colon \Delta \to \{-, \circledcirc\}$ as in the introduction, we can recast acceptance by final transitions as automata using the following condition:

$$Acc_{\mathrm{Last}} = \{w \in \{-, \circledcirc\}^* \mid \text{ the last letter of } w \text{ is } \circledcirc\}.$$

The role of prefix-independence and the empty word. When using the above general model of transition-based DFAs we encounter one inconvenience: the language recognised starting from a given state q may be ill-defined, since the set of runs accepted from q depends on the particular path that led to q from the initial state. Independence from the past of the run is a key property, notably for defining a minimal DFA, where each state corresponds to a left-quotient of the language.

This problem would not arise if the acceptance condition Acc was *prefix-independent*, that is, if for all sequences of transitions u_0 and u:

$$u_0 u \in Acc \iff u \in Acc.$$

However, the only prefix-independent languages of finite words are the empty and the full language, which cannot be used to recognise non-trivial languages. Indeed, if Acc is prefix-independent, then $u \in Acc \iff \varepsilon \in Acc$ for all $u \in \Sigma^*$.

Nevertheless, the language Acc_{Last} is almost prefix-independent, as it satisfies:

$$\text{for all } u_0 \text{ and } u \neq \varepsilon, \quad u_0 u \in Acc_{\mathrm{Last}} \iff u \in Acc_{\mathrm{Last}}\,.$$

[5] In this case, we should also specify whether the empty word is accepted.

This property makes Acc_{Last} well-suited for state-based acceptance, as the acceptance of ε can be encoded in a state, obtaining a definition of acceptance that is agnostic to the way we reach a given state.

Minimal transition-based DFA. It is well-known that the minimal state-based DFA of a regular language $L \subseteq \Sigma^*$ is given by the equivalence classes of the Myhill-Nerode congruence. In order to fit the transition-based setting, we can coarsen this relation, disregarding separations by the empty word:

$$u \dot{\sim}_L v \quad \overset{\text{def}}{\iff} \quad \text{for all } w \neq \varepsilon,\ uw \in L \iff vw \in L.$$

The next lemma is an easy check.

Lemma 11. *The relation $\dot{\sim}_L$ is an equivalence relation over Σ^*. Moreover, if $u \dot{\sim}_L v$, then $ua \dot{\sim}_L va$ for all $a \in \Sigma$.*

In the following, by a transition-based DFA we mean one with acceptance by final transitions, that is, using the acceptance condition Acc_{Last}.

Proposition 12. *Every regular language of finite words has a unique minimal transition-based DFA, which has one state per equivalence class of $\dot{\sim}_L$.*

Proof. Let $L \subseteq \Sigma^*$ be a regular language. Consider the DFA $\mathcal{A}_{\min}$ having as states the $\dot{\sim}_L$-classes of L, with $[\varepsilon]$ the initial state, and transitions $[u] \xrightarrow{a} [ua]$, where accepting transitions are those with $ua \in L$. Moreover, we need to specify whether $\varepsilon \in L$; in the positive case, we let the initial transition of the automaton be accepting. This automaton is well-defined and recognises L thanks to Lemma 11.

Let $\mathcal{A}$ be a transition-based DFA recognising L. For a state q in $\mathcal{A}$, let

$$L_\varepsilon^{\mathcal{A}}(q) = \{w \in \Sigma^+ \mid \text{ the run over } w \text{ from } q \text{ is accepting}\}.$$

It holds that, if u labels a path from the initial state to q, then $L_\varepsilon^{\mathcal{A}}(q) = L_\varepsilon^{\mathcal{A}_{\min}}([u])$. Moreover, $L_\varepsilon^{\mathcal{A}_{\min}}(u) \neq L_\varepsilon^{\mathcal{A}_{\min}}([v])$ if $u \not\dot{\sim}_L v$. Therefore, $\mathcal{A}_{\min}$ has at most as many states as $\mathcal{A}$, and in case of equality, they are isomorphic. □

Proposition 12 implies that transition-based DFAs are not larger than state-based ones. Moreover, they can be strictly smaller, as shown by the following example and Corollay 14 (see also [MRD24, Figs. 2-4]).

Example 13. *Let $\Sigma = \{a, b\}$ and consider the language of words that either have even lenght and end by 'b', or have odd length and end by 'a'. A minimal state-based DFA for this language, with 4 states, is given on the left of Figure 3. Note that the states q_a and q_b are not equivalent, as only one of them is accepting. However, $L_\varepsilon(q_a) = L_\varepsilon(q_b)$ (idem for p_a and p_b). Therefore, we can merge these states, obtaining a transition-based DFA with only 2 states.*

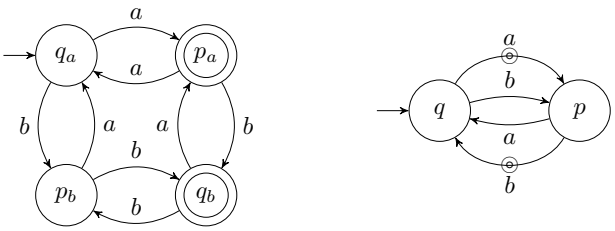


Figure 3: Two automata recognising the language $L = \{w \in \{a,b\}^+ \mid w \text{ is of even lenght if and only if it ends by '}b\text{'}\}$. The automaton on the left is the minimal state-based DFA of L, and the automaton on the right is its minimal transition-based DFA.

Generalising the previous example and using Proposition 12, we obtain:

Corollary 14. *Every transition-based DFA admits an equivalent state-based DFA with at most twice as many states. This bound is tight: there is a family of languages for which a minimal state-based DFA has twice as many states as a minimal transition-based DFA.*

Proof. For the first claim, it suffices to note that the index of the classical Myhill-Nerode congruence is at most twice the index of $\dot{\sim}_L$.

For the second claim, let $\Sigma = \{a,b\}$ and consider the language:

$$L_n = \{w \in \Sigma^+ \mid w \text{ ends by '}b\text{' if and only if } |w| \equiv 0 \bmod n\}.$$

The congruence $\dot{\sim}_L$ has n classes, corresponding to the remainder of $|w|$ modulo n. The Myhill-Nerode congruence has $2n$ classes, as words ending with a different letter are not equivalent. □

We note that such a gap does not appear for non-deterministic automata. Indeed, every transition-based NFA can be converted into an equivalent state-based NFA with only one more state. It suffices to add a sink state, which will be the only accepting state, and duplicate all accepting transitions redirecting one copy towards this sink. (This operation increases the number of transitions though.)

Where does this leave us? As we have seen, transition-based DFAs can be smaller than classical state-based ones. Moreover, most standard constructions adapt to the transition-based setting without problem (determinisation, product construction, removal of ε-transitions, etc). Some of them, such as the conversion of regular expressions, may even benefit from the use of transition-based acceptance [MRD24]. Transition-based DFAs can be of particular interest when used

as an intermediate step for the construction of ω-automata [MRD24], or when the acceptance of the empty word is irrelevant, as in LTL_f semantics.

However, there are signs pointing towards the canonicity of state-based acceptance for DFAs. Notably, the syntactic monoid of a language equals the transition monoid of its minimal state-based DFA [Pin, Prop. 4.28]. It is unclear to us what would be the correct way to recover the syntactic monoid from a transition-based DFA. Other questions regarding the transition-based model remain open. For instance, are there acceptance conditions other than Acc_{Last} that lead to unique minimal DFAs for all regular languages?

6 Outlook: Why all these differences?

We have seen various situations where transition-based acceptance is more advantageous, both for practical and theoretical reasons. The following question arises naturally: What are the fundamental differences between state-based and transition-based models that lead to such contrasting properties?

Composition of transitions. A basic operation at the heart of many reasonings in automata theory is *composition of transitions*. If an automaton contains transitions $p \xrightarrow{a} q$ and $q \xrightarrow{b} r$, one can go from p to r by reading ab, and any "effect" of this path should be the result of concatenating the effects of these two transitions. That is, a suitable automata model should allow to add the transition $p \xrightarrow{ab} r$. For automata over infinite words, the acceptance of the automaton obtained by adding this transition can only be defined in a sensible way by using a transition-based condition.

This composition operation is key for the celebrated connection between automata and algebra. The suitability of transition-based models for algebraic approaches is explicitly mentionned in Michel's work introducing transition-based ω-automata [Mic84, Section II]:

> *Using unstable graphs, instead of a set of nodes that must be traversed infinitely often, is better suited to the algebraic operations we will define [...]* [6]

Similarly, one of LeSaëc's motivations for the use of transition-based automata was to obtain an algebraic proof of McNaughton's theorem for infinite words [SPW91]. The Muller automaton obtained from a given semigroup is naturally transition-based, see [SPW91, page 18] and [Col11, Section 6].

[6] In French in the original: *L'utilisation de graphes instables au lieu d'un ensemble de nœuds dans lequel on doit passer infiniment souvent se prête mieux aux opérations algébriques que nous définirons [...].*

As mentioned in Section 4, composition of transitions is also essential in the fruitful approach for solving and analysing infinite duration games based on universal graphs, which relies on the notions of monotonicity, ε-completion and the technique of saturation (for the latter, see [CF18, Section 4], [Col+22, Section 4.1] or [Ohl23, Section 3.3]).

We note, however, that in the case of finite words, this does not provide strong evidence in favour of transition-based acceptance. Indeed, state-based DFAs also allow for composition of transitions, as the acceptance of a run is only determined by its final destination.

Paths in graphs. As explained in the introduction, an acceptance condition is a representation of a subset of paths in an automaton. A path in a graph is commonly defined as a sequence of edges. In fact, a sequence of vertices does not completely determine a path, as different paths may share the same sequence of vertices. This is the main reason why transition-based automata are more succinct than state-based ones.

Final thoughts

The collection of results presented in this survey indicates that, despite the fact that the size of state-based and transition-based automata only differ by a linear factor, transition-based models are easier to manipulate and have a nicer theory. We therefore advocate adopting transition-based acceptance as the default model for ω-automata.

We expect that the use of transition-based acceptance will ease the finding of automata-based characterisation of classes of languages. This has already been the case, for example, in the characterisation of positional ω-regular languages based on parity automata with a particular structure [CO24, Thm. 3.1].

In the same spirit, it appears that the use of transition-based models will be required for obtaining canonical models of automata over infinite words or trees. Steps in this direction have already been made [ES22, Ehl25, LW25], building on the description of canonical history-deterministic coBüchi automata by Abu Radi and Kupferman [AK22].

Acknowledgements. I warmly thank Thomas Colcombet for many discussions on the benefits of transition-based acceptance, Alexandre Duret-Lutz for valuable comments on automata over finite words and for sharing many historical references, Géraud Sénizergues for pointing me to the works of Bertrand Le Saëc and Pierre Ohlmann for helpful feedback on a draft of this paper.

References

[AE25] Bader Abu Radi and Rüdiger Ehlers. "Characterizing the polynomial-time minimizable ω-automata". In: *Corr* abs/2504.20553 (2025). DOI: 10.48550/ARXIV.2504.20553.

[AK22] Bader Abu Radi and Orna Kupferman. "Minimization and canonization of GFG transition-based automata". In: *Log. Methods Comput. Sci.* 18.3 (2022). DOI: 10.46298/lmcs-18(3:16)2022.

[AK19] Bader Abu Radi and Orna Kupferman. "Minimizing GFG transition-based automata". In: *ICALP*. Vol. 132. LIPIcs. 2019, 100:1–100:16. DOI: 10.4230/LIPIcs.ICALP.2019.100.

[Bab+15] Tomás Babiak, Frantisek Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Kretínský, David Müller, David Parker, and Jan Strejcek. "The Hanoi Omega-Automata format". In: *CAV*. Vol. 9206. 2015, pp. 479–486. DOI: 10.1007/978-3-319-21690-4_31.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. URL: https://mitpress.mit.edu/9780262026499/principles-of-model-checking/.

[BCJ18] Roderick Bloem, Krishnendu Chatterjee, and Barbara Jobstmann. "Graph games and reactive synthesis". In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer International Publishing, 2018, pp. 921–962. DOI: 10.1007/978-3-319-10575-8_27.

[Boj] Mikołaj Bojańczyk. *An automata toolbox*. Version of 7 February, 2025. URL: https://www.mimuw.edu.pl/~bojan/papers/toolbox.pdf.

[Bok18] Udi Boker. "Why these automata types?" In: *LPAR*. Vol. 57. EPiC Series in Computing. 2018, pp. 143–163. DOI: 10.29007/c3bj.

[BRV23] Patricia Bouyer, Mickael Randour, and Pierre Vandenhove. "Characterizing omega-regularity through finite-memory determinacy of games on infinite graphs". In: *Theoretics* 2 (2023). DOI: 10.46298/theoretics.23.1.

[Büc62] J. Richard Büchi. "On a decision method in restricted second order arithmetic". In: *Proc. Internat. Congr. on Logic, Methodology and Philosophy of Science* (1962), pp. 1–11.

[BL69a] J. Richard Büchi and Lawrence H. Landweber. "Definability in the monadic second-order theory of successor". In: *J. Symb. Log.* 34.2 (1969), pp. 166–170. DOI: 10.2307/2271090.

[BL69b] J. Richard Büchi and Lawrence H. Landweber. "Solving sequential conditions by finite-state strategies". In: *Transactions of the American Mathematical Society* 138 (1969), pp. 295–311. URL: http://www.jstor.org/stable/1994916.

[Cal+22] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. "Deciding parity games in quasi-polynomial time". In: *SIAM Journal on Computing* 51.2 (2022), pp. 152–188. DOI: 10.1137/17M1145288.

[Cas22] Antonio Casares. "On the minimisation of transition-based Rabin automata and the chromatic memory requirements of Muller conditions". In: *CSL*. Vol. 216. 2022, 12:1–12:17. DOI: 10.4230/LIPIcs.CSL.2022.12.

[Cas+24] Antonio Casares, Thomas Colcombet, Nathanaël Fijalkow, and Karoliina Lehtinen. "From Muller to parity and Rabin automata: Optimal transformations preserving (history) determinism". In: *TheoretiCS* Volume 3 (Apr. 2024). DOI: 10.46298/theoretics.24.12.

[CCL22] Antonio Casares, Thomas Colcombet, and Karoliina Lehtinen. "On the size of good-for-games Rabin automata and its link with the memory in Muller games". In: *ICALP*. Vol. 229. 2022, 117:1–117:20. DOI: 10.4230/LIPIcs.ICALP.2022.117.

[Cas+25] Antonio Casares, Olivier Idir, Denis Kuperberg, Corto Mascle, and Aditya Prakash. "On the minimisation of deterministic and history-deterministic generalised (co)Büchi automata". In: *CSL*. Vol. 326. 2025, 22:1–22:18. DOI: 10.4230/LIPICS.CSL.2025.22.

[CO25a] Antonio Casares and Pierre Ohlmann. "Characterising memory in infinite games". In: *Log. methods comput. sci.* 21.1 (2025). DOI: 10.46298/LMCS-21(1:28)2025.

[CO24] Antonio Casares and Pierre Ohlmann. "Positional ω-regular languages". In: *LICS*. ACM, 2024, 21:1–21:14. DOI: 10.1145/3661814.3662087.

[CO25b] Antonio Casares and Pierre Ohlmann. "The memory of ω-regular and BC(Σ_2^0) objectives". In: *ICALP*. Vol. 334. 2025, 149:1–149:18. DOI: 10.4230/LIPICS.ICALP.2025.149.

[Col11] Thomas Colcombet. "Green's relations and their use in automata theory". In: *LATA*. Vol. 6638. 2011, pp. 1–21. DOI: 10.1007/978-3-642-21254-3_1.

[CF18] Thomas Colcombet and Nathanaël Fijalkow. "Parity games and universal graphs". In: *Corr* abs/1810.05106 (2018). URL: http://arxiv.org/abs/1810.05106.

[Col+22] Thomas Colcombet, Nathanaël Fijalkow, Pawel Gawrychowski, and Pierre Ohlmann. "The theory of universal graphs for infinite duration games". In: *Log. Methods Comput. Sci.* 18.3 (2022). DOI: 10.46298/lmcs-18(3:29)2022.

[CN06] Thomas Colcombet and Damian Niwiński. "On the positional determinacy of edge-labeled games". In: *Theor. Comput. Sci.* 352.1-3 (2006), pp. 190–196. DOI: 10.1016/j.tcs.2005.10.046.

[CZ09] Thomas Colcombet and Konrad Zdanowski. "A tight lower bound for determinization of transition labeled Büchi automata". In: *ICALP*. 2009, pp. 151–162. DOI: 10.1007/978-3-642-02930-1_13.

[Cou99] Jean-Michel Couvreur. "On-the-fly verification of linear temporal logic". In: *World congress on formal methods in the development of computing systems*. Vol. 1708. 1999, pp. 253–271. DOI: 10.1007/3-540-48119-2_16.

[Dur07] Alexandre Duret-Lutz. "Contributions à l'approche automate pour la vérification de propriétés de systèmes concurrents". PhD thesis. Université Pierre et Marie Curie (Paris 6), 2007. URL: https://www.lrde.epita.fr/~adl/dl/adl/duret.07.phd.pdf.

[DP04] Alexandre Duret-Lutz and Denis Poitrenaud. "SPOT: an extensible model checking library using transition-based generalized Büchi automata". In: *MASCOTS*. IEEE Computer Society, 2004, pp. 76–83. DOI: 10.1109/MASCOT.2004.1348184.

[Dur+22] Alexandre Duret-Lutz, Etienne Renault, Maximilien Colange, Florian Renkin, Alexandre Gbaguidi Aisse, Philipp Schlehuber-Caissier, Thomas Medioni, Antoine Martin, Jérôme Dubois, Clément Gillard, and Henrich Lauko. "From Spot 2.0 to Spot 2.10: What's new?" In: *CAV*. Vol. 13372. 2022, pp. 174–187. DOI: 10.1007/978-3-031-13188-2_9.

[Dur+25] Alexandre Duret-Lutz, Shufang Zhu, Nir Piterman, Giuseppe De Giacomo, and Moshe Y. Vardi. "Engineering an LTLf synthesis tool". In: *CIAA*. To appear. 2025. URL: http://arxiv.org/abs/2507.02491.

[Ehl25] Rüdiger Ehlers. "Rerailing automata". In: *Corr* abs/2503.08438 (2025). DOI: 10.48550/ARXIV.2503.08438.

[ES22] Rüdiger Ehlers and Sven Schewe. "Natural colors of infinite words". In: *FSTTCS*. Vol. 250. 2022, 36:1–36:17. DOI: 10.4230/LIPIcs.FSTTCS.2022.36.

[Eil74] Samuel Eilenberg. *Automata, languages, and machines. A*. Pure and applied mathematics. Academic Press, 1974. URL: https://www.worldcat.org/oclc/310535248.

[EJ91] E. Allen Emerson and Charanjit S. Jutla. "Tree automata, mu-calculus and determinacy (extended abstract)". In: *FOCS*. 1991, pp. 368–377. DOI: 10.1109/SFCS.1991.185392.

[EB23] Javier Esparza and Michael Blondin. *Automata theory: An algorithmic approach*. MIT Press, 2023. URL: https://michaelblondin.com/automata/.

[Fij+25] Nathanaël Fijalkow, C. Aiswarya, Guy Avni, Nathalie Bertrand, Patricia Bouyer, Romain Brenguier, Arnaud Carayol, Antonio Casares, John Fearnley, Paul Gastin, Hugo Gimbert, Thomas A. Henzinger, Florian Horn, Rasmus Ibsen-Jensen, Nicolas Markey, Benjamin Monmege, Petr Novotný, Pierre Ohlmann, Mickael Randour, Ocan Sankur, Sylvain Schmitz, Olivier Serre, Mateusz Skomra, Nathalie Sznajder, and Pierre Vandenhove. *Games on graphs: from logic and automata to algorithms*. Ed. by Nathanaël Fijalkow. Cambridge University Press, 2025. URL: https://arxiv.org/abs/2305.10546.

[GO01] Paul Gastin and Denis Oddoux. "Fast LTL to Büchi automata translation". In: *CAV*. Vol. 2102. 2001, pp. 53–65. DOI: 10.1007/3-540-44585-4_6.

[GL02] Dimitra Giannakopoulou and Flavio Lerda. "From states to transitions: improving translation of LTL formulae to Büchi automata". In: *FORTE*. Vol. 2529. 2002, pp. 308–326. DOI: 10.1007/3-540-36135-9_20.

[GTW02] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, eds. *Automata logics, and infinite games*. Springer, Berlin, Heidelberg, 2002. DOI: 10.1007/3-540-36387-4.

[GH82] Yuri Gurevich and Leo Harrington. "Trees, automata, and games". In: *STOC*. 1982, pp. 60–65. DOI: 10.1145/800070.802177.

[HD05] Paul Hunter and Anuj Dawar. "Complexity bounds for regular games". In: *MFCS*. 2005, pp. 495–506. DOI: 10.1007/11549345_43.

[KMS18] Jan Kretínský, Tobias Meggendorfer, and Salomon Sickert. "Owl: A library for ω-words, automata, and LTL". In: *ATVA*. Vol. 11138. 2018, pp. 543–550. DOI: 10.1007/978-3-030-01090-4_34.

[KS15] Denis Kuperberg and Michał Skrzypczak. "On determinisation of good-for-games automata". In: *ICALP*. 2015, pp. 299–310. DOI: 10.1007/978-3-662-47666-6_24.

[Kup18] Orna Kupferman. "Automata theory and model checking". In: *Handbook of Model Checking*. Ed. by Edmund M. Clarke, Thomas A. Henzinger, Helmut Veith, and Roderick Bloem. Springer International Publishing, 2018, pp. 107–151. DOI: 10.1007/978-3-319-10575-8_4.

[Kur87] R.P. Kurshan. "Complementing deterministic Büchi automata in polynomial time". In: *Journal of computer and system sciences* 35.1 (1987), pp. 59–71. DOI: https://doi.org/10.1016/0022-0000(87)90036-5.

[Lan69] Lawrence H. Landweber. "Decision problems for omega-automata". In: *Math. Syst. Theory* 3.4 (1969), pp. 376–384. DOI: 10.1007/BF01691063.

[Löd21] Christof Löding. "Automata on infinite trees". In: *Handbook of automata theory*. Ed. by Jean-Éric Pin. 2021, pp. 265–302. DOI: 10.4171/AUTOMATA-1/8.

[LW25] Christof Löding and Igor Walukiewicz. "Minimal history-deterministic co-Buchi automata: Congruences and passive learning". In: *Corr* abs/2505.14304 (2025). DOI: 10.48550/ARXIV.2505.14304.

[Maj+19] Juraj Major, Frantisek Blahoudek, Jan Strejcek, Miriama Sasaráková, and Tatiana Zboncáková. "Ltl3tela: LTL to small deterministic or nondeterministic Emerson-Lei automata". In: *ATVA*. Vol. 11781. 2019, pp. 357–365. DOI: 10.1007/978-3-030-31784-3_21.

[MRD24] Antoine Martin, Etienne Renault, and Alexandre Duret-Lutz. "Translation of semi-extended regular expressions using derivatives". In: *CIAA*. Vol. 15015. 2024, pp. 234–248. DOI: 10.1007/978-3-031-71112-1_17.

[McN93] Robert McNaughton. "Infinite games played on finite graphs". In: *Annals of Pure and Applied Logic* (1993). DOI: 10.1016/0168-0072(93)90036-D.

[McN66] Robert McNaughton. "Testing and generating infinite sequences by a finite automaton". In: *Information and control* 9.5 (1966), pp. 521–530. DOI: 10.1016/S0019-9958(66)80013-X.

[Mea55] George H. Mealy. "A method for synthesizing sequential circuits". In: *Bell Syst. Tech. J.* 34(5) (1955), pp. 1045–1079.

[Mic84] Max Michel. "Algebre de machines et logique temporelle". In: *STACS*. 1984, pp. 287–298. DOI: 10.1007/3-540-12920-0_26.

[Moo56] Edward F. Moore. "Gedanken-experiments on sequential machines". In: *Automata studies*. Ed. by C. E. Shannon and J. McCarthy. 34. 1956, pp. 129–153.

[Mos84] Andrzej W. Mostowski. "Regular expressions for infinite trees and a standard form of automata". In: *SCT*. 1984, pp. 157–168. DOI: 10.1007/3-540-16066-3_15.

[Mul63] David E. Muller. "Infinite sequences and finite machines". In: *Symposium on Switching Circuit Theory and Logical Design*. 1963, pp. 3–16. DOI: 10.1109/SWCT.1963.8.

[Ohl23] Pierre Ohlmann. "Characterizing positionality in games of infinite duration over infinite graphs". In: *TheoretiCS* 2 (2023). DOI: 10.46298/theoretics.23.3.

[Par81] David Park. "Concurrency and automata on infinite sequences". In: *Theoretical computer science*. Ed. by Peter Deussen. 1981, pp. 167–183. DOI: 10.1007/BFb0017309.

[PP04] Dominique Perrin and Jean-Eric Pin. *Infinite words - Automata, semigroups, logic and games*. Vol. 141. Pure and applied mathematics series. 2004.

[Pin] Jean-Eric Pin. *Mathematical foundations of automata theory. Notes of the MPRI lectures.* Version of March 2025. URL: https://www.irif.fr/~jep/PDF/MPRI/MPRI.pdf.

[Pit06] Nir Piterman. "From nondeterministic Büchi and Streett automata to deterministic parity automata". In: *LICS*. 2006, pp. 255–264. DOI: 10.1109/LICS.2006.28.

[RS59] M. O. Rabin and D. Scott. "Finite automata and their decision problems". In: *IBM journal of research and development* 3.2 (1959), pp. 114–125. DOI: 10.1147/rd.32.0114.

[Rab69] Michael O. Rabin. "Decidability of second-order theories and automata on infinite trees". In: *Transactions of the American Mathematical Society* 141 (1969), pp. 1–35. URL: http://www.jstor.org/stable/1995086.

[Red12] Roman R. Redziejowski. "An improved construction of deterministic omega-automaton using derivatives". In: *Fundam. Informaticae* 119.3-4 (2012), pp. 393–406. DOI: 10.3233/FI-2012-744.

[Red99] Roman R. Redziejowski. "Construction of a deterministic-automaton using derivatives". In: *RAIRO Theor. Informatics Appl.* 33.2 (1999), pp. 133–158. DOI: 10.1051/ITA:1999111.

[Red72] Roman R. Redziejowski. *The theory of general events and its application to parallel programming*. IBM Nordic Laboratory, 1972.

[Saë90] Bertrand Le Saëc. "Saturating right congruences". In: *RAIRO* 24 (1990), pp. 545–559. DOI: 10.1051/ita/1990240605451.

[SPW91] Bertrand Le Saëc, Jean-Eric Pin, and Pascal Weil. "Semigroups with idempotent stabilizers and applications to automata theory". In: *Int. J. Algebra Comput.* 1.3 (1991), pp. 291–314. DOI: 10.1142/S0218196791000195.

[Saf88] Schmuel Safra. "On the complexity of ω-automata". In: *FOCS*. 1988, pp. 319–327. DOI: 10.1109/SFCS.1988.21948.

[Sch10] Sven Schewe. "Beyond hyper-minimisation—minimising DBAs and DPAs is NP-complete". In: *FSTTCS*. Vol. 8. 2010, pp. 400–411. DOI: 10.4230/LIPIcs.FSTTCS.2010.400.

[Sch20] Sven Schewe. "Minimising good-for-games automata is NP-complete". In: *FSTTCS*. Vol. 182. 2020, 56:1–56:13. DOI: 10.4230/LIPIcs.FSTTCS.2020.56.

[Sch09] Sven Schewe. "Tighter bounds for the determinisation of Büchi automata". In: *FOSSACS*. 2009, pp. 167–181. DOI: 10.1007/978-3-642-00596-1_13.

[SV14] Sven Schewe and Thomas Varghese. "Determinising parity automata". In: *MFCS*. 2014, pp. 486–498. DOI: 10.1007/978-3-662-44522-8_41.

[SV12] Sven Schewe and Thomas Varghese. "Tight bounds for the determinisation and complementation of generalised Büchi automata". In: *ATVA*. 2012, pp. 42–56. DOI: 10.1007/978-3-642-33386-6_5.

[Sch61a] Marcel Paul Schützenberger. "A remark on finite transducers". In: *Inf. Control.* 4.2-3 (1961), pp. 185–196. DOI: 10.1016/S0019-9958(61)80006-5.

[Sch61b] Marcel Paul Schützenberger. "On the definition of a family of automata". In: *Inf. Control.* 4.2-3 (1961), pp. 245–270. DOI: 10.1016/S0019-9958(61)80020-X.

[Sha48] Claude E. Shannon. "A mathematical theory of communication". In: *Bell Syst. Tech. J.* 27 (1948), pp. 623–656.

[Shi+20] Yingying Shi, Shengping Xiao, Jianwen Li, Jian Guo, and Geguang Pu. "SAT-based automata construction for LTL over finite traces". In: *APSEC*. IEEE, 2020, pp. 1–10. DOI: 10.1109/APSEC51365.2020.00008.

[Tho90] Wolfgang Thomas. "Automata on infinite objects". In: *Handbook of theoretical computer science, volume B: formal models and semantics*. Ed. by Jan van Leeuwen. Elsevier and MIT Press, 1990, pp. 133–191. DOI: 10.1016/B978-0-444-88074-1.50009-3.

[Tho97] Wolfgang Thomas. "Languages, automata, and logic". In: *Handbook of formal languages, volume 3: Beyond words*. 1997, pp. 389–455. DOI: 10.1007/978-3-642-59126-6_7.

[VSL95] Do Long Van, Bertrand Le Saëc, and Igor Litovsky. "Characterizations of rational omega-languages by means of right congruences". In: *Theor. Comput. Sci.* 143.1 (1995), pp. 1–21. DOI: 10.1016/0304-3975(95)80022-2.

[Var14] Thomas Varghese. "Parity and generalised Büchi automata. Determinisation and complementation". PhD Thesis. University of Liverpool, 2014.

[Wag79] Klaus Wagner. "On ω-regular sets". In: *Information and control* 43.2 (1979), pp. 123–177. DOI: 10.1016/S0019-9958(79)90653-3.

[WS21] Thomas Wilke and Sven Schewe. "ω-automata". In: *Handbook of automata theory*. Ed. by Jean-Éric Pin. European Mathematical Society Publishing House, 2021, pp. 189–234. DOI: 10.4171/Automata-1/6.

[Xia+21] Shengping Xiao, Jianwen Li, Shufang Zhu, Yingying Shi, Geguang Pu, and Moshe Vardi. "On-the-fly synthesis for LTL over finite traces". In: *AAAI* 35.7 (2021), pp. 6530–6537. DOI: 10.1609/aaai.v35i7.16809.

[Xia+24] Shengping Xiao, Yongkang Li, Xinyue Huang, Yicong Xu, Jianwen Li, Geguang Pu, Ofer Strichman, and Moshe Y. Vardi. "Model-guided synthesis for LTL over finite traces". In: *VMCAI*. Vol. 14499. 2024, pp. 186–207. DOI: 10.1007/978-3-031-50524-9_9.

[Zie98] Wiesław Zielonka. "Infinite games on finitely coloured graphs with applications to automata on infinite trees". In: *Theoretical Computer Science* 200.1-2 (1998), pp. 135–183. DOI: 10.1016/S0304-3975(98)00009-7.

The Logic in Computer Science Column

BY

Anuj Dawar

Department of Computer Science and Technology
University of Cambridge, Cambridge, CB3 0FD, UK
anuj.dawar@cl.cam.ac.uk

I am pleased to introduce myself as the new editor of the *Logic in Computer Science Column* in the Bulletin. I am stepping into the shoes of Yuri Gurevich, who has been running this column since 1988 and that is a daunting task. I aim to do my best to live up to the standards that he has set. If you wish to contribute a column or have suggestions for other excellent expositors who might be able to, I would love to hear from you. Please do get in touch.

To kick things off, I offer a column of my own. This is based on an invited talk that I gave at the LICS 2025 conference in Singapore in June 2025. It was after returning from the conference that I ran into Quisani, Yuri's old student who wanted to hear what I had talked about. We had a fascinating conversation and you can read all about it below.

BEATCS no 147

Notions of Width: Variables, Pebbles and Supports

Anuj Dawar

Abstract

Given a formula, what is the smallest number of variables with which it can be equivalently written? What seems like an abstruse question in syntactic manipulation turns out to have significance in a variety of areas of theoretical computer science. The number of variables in a formula emerges as an important measure related to notions of width arising in fields as varied as database theory; combinatorics and graph theory; and permutation groups. I explore how these notions are related to each other and the exploration will take us through a diverse landscape of topics, from comonads to lower bounds in circuit complexity.

1 Width

Quisani: Hello, I'm Quisani. I'm a former student of Yuri Gurevich and I often meet him here for chats about Logic in Computer Science. I don't believe I have met you before.
Author: Hello, it's nice to meet you. My name is Anuj Dawar, and I am new here. I've heard a lot about your chats. Yuri has asked me to take over his responsibilities here, so I expect to be coming here regularly and seeing more of you.

Q: Does that mean that Yuri is retiring from his role?

A: Not completely. He will be back from time to time, so you will see more of him. But, I expect to take over from him in due course. It is daunting to step into his shoes, but I hope we can find plenty to chat about from the world of Logic in Computer Science.

Q: I look forward to it. Do you have any new and interesting stories from that world to share today?

A: I have recently returned from the LICS 2025 conference in Singapore [4] where I gave an invited talk and I thought I might tell you what I talked about. The title of my talk was *Notions of Width: Variables, Pebbles and Supports*.

Q: Sounds intriguing. What do you mean by *width*?

A: The word *width* is used to mean many different things, even within the fields of logic and combinatorics. Not all of these uses of the word are related to each other. But, in my talk I was trying to show that many different measures of complexity that go by this name are, in fact, related and sometimes surprisingly so. I start by asking the following question: given a first-order formula φ, say in the language of graphs, what is the smallest number of variables with which φ can be equivalently rewritten?

Q: I suppose I should see this as an algorithmic problem: given φ, calculate the smallest k such that φ is equivalent to a formula with just k variables. This sounds to me like it should be uncomputable, given that equivalence is undecidable.

A: You are quite right. But, I am not so much interested in computing the minimum but using the number of variables as a measure of the complexity of a formula. So, let us say that a formula φ has *width* k if no subformula of φ has more than k *free* variables. This is sometimes called the *syntactic width* of the formula.

Q: Yes, I see. If that is the case, then it is clear that you can rename bound variables in such a way that you use no more than k distinct variables. This means that the formula is equivalent to one in L^k, the fragment of first-order logic that uses only the variables $x_1, \ldots, x_k$. I remember a conversation with Ian Hodkinson [23] about this logic many years ago. We also talked about infinitary logics: $L^k_{\infty\omega}$ which is the closure of L^k under infinitary conjunctions and disjunctions and the logic $L^\omega_{\infty\omega}$ which is obtained as the union of $L^k_{\infty\omega}$ as k varies.

A: Indeed, I studied these logics back when I was a graduate student and you talked about some of that work with Ian. I remember once giving a talk in which I mentioned the logic L^k and a member of the audience asking me: "what is the point of a logic that is not closed under the renaming of bound variables"? And this is one reason I emphasise the characterization in terms of limiting the *width*: the maximum number of free variables in any subformula. That's what's important and not the actual names of the variables. It helps explain the intuition of the resource we are limiting, i.e. the number of things we can simultaneously refer to and connect. But, what I want to tell you about today is how this notion of width connects to many other notions of width that arise in logic, combinatorics and complexity, not to mention database theory and constraint programming. Let me start by telling you a bit about conjunctive queries in databases.

2 Conjunctive Queries

Q: I have heard of *conjunctive queries*. Can you remind me what they are?

A: From the point of view of a logician, it is best to describe this as a fragment of first-order logic. For simplicity, let's assume that we are always working in the language of graphs. That is, we only have one binary relation E and no other relation or function symbols. The formulas of first-order logic can then be described by the following syntax:

$$\varphi := x = y \mid E(x, y) \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \neg\varphi \mid \exists x\varphi \mid \forall x\varphi.$$

Each formula $\varphi(\mathbf{x})$ with free variables $\mathbf{x}$ defines a *query*, which we think of as a function from graphs G to relations $\varphi^G \subseteq G^{\mathbf{x}}$. Conjunctive queries are then those that can be defined by formulas without negation, disjunction or universal quantification. Once we have disallowed these operations from our formulas, we can also dispense with equality. Do you see why?

Q: Yes, when $x = y$ appears in a conjunction, we can simply replace all occurrences of y with x and get rid of y. This way we also get rid of a variable, which is useful since we are being careful with how many we need.

A: Excellent! This leaves us with this highly reduced syntax.

$$\varphi := E(x, y) \mid \varphi \wedge \varphi \mid \exists x\varphi.$$

Can we say anything useful with this? Suppose I want to express the property of a directed graph that it contains a walk of length five. Can you see how to do it?

Q: I can. A walk of length five is just a sequence of six vertices, not necessarily distinct, such that there is an edge from each vertex to its successor in the sequence. I would express it with something like this:

$$\exists x_1 \cdots \exists x_6 (E(x_1, x_2) \wedge \cdots \wedge E(x_5, x_6)).$$

A: That certainly works and note how you had to use six variables. But, with a careful re-use of variables, you can get away with using just two. Here's how:

$$\exists x \exists y (E(x, y) \wedge \exists x (E(y, x) \wedge \exists y (\cdots))), \tag{1}$$

where I'm sure you can fill in the $\cdots$.

Q: I see. Instead of having the quantifiers all in front like I did, you push them in as far as you can.

A: Poizat [33] calls the logic L^k an "enemy of the prenex normal form" for this reason.

Q: There is nothing special about the fact that the walk is of length five. For a walk of length n, in prenex normal form I would need $n + 1$ variables but I can

always get away with just two. But this contrast with prenex normal form is not just about conjunctive queries. What's special about them?

A: Conjunctive queries are widely studied in database theory as they correspond to the *select-project-join* queries which are said to be the most common kind of queries on relational databases [1]. And we care about the number of variables because it is tied to the complexity of the natural algorithm for evaluating the query in a database. Let's take the example from (1). Picture the parse tree of the formula, which we can see in Figure 1.

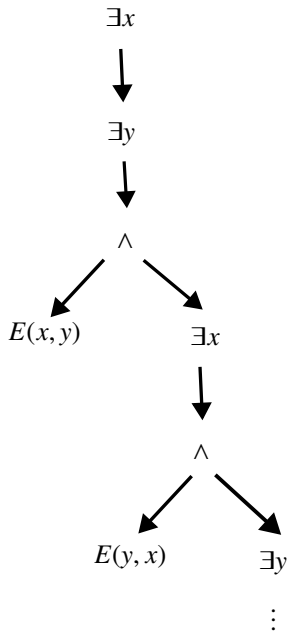


Figure 1: The parse tree of the formula $\exists x \exists y (E(x,y) \wedge \exists x (E(y,x) \wedge \exists y (\cdots)))$.

Now, imagine the process of evaluating this formula bottom up in a graph G. With each node we can associate the relation that is defined in G by the query given by the subformula below that node. The operations at the nodes are *select* (the atomic formulas), *project* (the existential quantifiers) and *join* (for the conjunctions)—hence the name.

Q: I see, and the fact that each subformula has at most two free variables means all the relations are at most binary.

A: Exactly. This limits the complexity of the evaluation process. Most importantly, it limits the space complexity as storing relations of higher arity takes up a lot

of space. There is much work in database theory aimed at optimizing queries. Cutting down the number of variables in a conjunctive query is an important optimization as it greatly reduces the size of the intermediate results that we have to consider. That is a topic for another day. If you want to know how the number of variables may be optimized, I would suggest looking at this paper [7].

For now, I want to look at structural questions about conjunctive queries. Chandra and Merlin [9] noted that the problem of conjunctive query evaluation is the same as *structure homomorphism*.

Q: As I recall, for a pair of structures $\mathbb{A}$ and $\mathbb{B}$ in the same vocabulary, a homomorphism from $\mathbb{A}$ to $\mathbb{B}$ is a function $h : A \to B$ from the universe A of $\mathbb{A}$ to the universe B of $\mathbb{B}$ that preserves all relations.

A: Yes, in particular, if $\mathbb{A}$ and $\mathbb{B}$ are graphs, this means that whenever (u, v) is an edge in $\mathbb{A}$ then $(h(u), h(v))$ is an edge in $\mathbb{B}$. So, what Chandra and Merlin observe is that from any conjunctive query φ we can construct a structure M_φ such that for any $\mathbb{A}$, we have $\mathbb{A} \models \varphi$ if, and only if, there is a homomorphism from M_φ to $\mathbb{A}$. Can you see how to construct M_φ from φ?

Q: I suppose so. If we put φ in prenex normal form, it is a list of existential quantifiers followed by a conjunction of atomic formulas. We create M_φ by taking the collection $x_1, \ldots, x_n$ of quantified variables as elements, and then put an edge between x_i and x_j whenever an atomic formula $E(x_i, x_j)$ appears in the body.

A: That's right. Then, a homomorphism from M_φ to $\mathbb{A}$ is exactly an evaluation of the existential quantifiers of φ in $\mathbb{A}$ which makes all conjuncts true.

Q: It's funny how we considered the prenex normal form of φ, which you said is the opposite of minimizing the number of variables. What does this have to do with the syntactic width of the formula?

A: It turns out that the syntactic width of φ is related to an important structural parameter of the graph M_φ: its *treewidth*. The treewidth of a graph G is sometimes described as a measure of how tree-like G is. Roughly speaking, G has treewidth k if its edges can be covered by subgraphs of at most $k + 1$ vertices in a tree-like fashion. I won't give a detailed definition of it here as it is well-studied in the literature.

Q: Yes, I have heard a lot about it. I understand that it plays an important role in parameterized algorithms.

A: It also has a central role in structural graph theory and often pops up in logic. If you want to know more I recommend the book on *Sparsity* by Nešetřil and Ossona de Mendez [30].

Q: So, how does the treewidth of M_φ connect with the syntactic width of φ?

A: If the treewidth of M_φ is less than k, then φ can be written with at most k variables. Indeed, you can turn a tree decomposition of M_φ into a formula quite directly and the number of variables is just one more than the width of the decomposition. You can find details of this and much related material in this paper by Dalmau et al. [10].

Q: I like how we've related a purely combinatorial parameter from graph theory to a syntactic measure of the complexity of formulas. But, I suppose this is because of the special nature of conjunctive queries. As we noted, they are almost already just structures with a string of existential quantifiers in front. Can you say something about the syntactic width of more general formulas? As I recall, the reason for studying logics like L^k and $L^\omega_{\infty\omega}$ has to do with *fixed-point logics*.

A: You are absolutely right. Let's turn to these next.

3 Fixed-Point and Counting Logics

A: *Fixed-point logic*, which we'll abbreviate FP, is an extension of first-order logic with a mechanism for recursion.

Q: I know it well. I have often discussed this with Yuri [5], and it also came up in my chat with Ian Hodkinson all those years ago. I know that what is called the Immerman-Vardi theorem [24, 35] tells us that a property of finite ordered structures is decidable in polynomial time if, and only if, it is definable in FP. Yuri told me that this was also independently discovered by Livchak [27].

A: That's all true. But, if you don't restrict yourself to ordered structures there are simple properties that are not definable.

Q: Well, yes. Every formula of FP is equivalent to one of $L^\omega_{\infty\omega}$ and Kolaitis and Vardi [26] proved that the latter logic has a 0-1 law so cannot express, for instance, that a structure has an even number of elements.

A: Indeed, simple counting properties are not definable in FP. This led Immerman [24] to propose that an extension of FP with a mechanism for counting, which we now call FPC, might express all polynomial-time properties. Even this turns out not to be the case but it requires a sophisticated construction to come up with such a property. Still, the problems definable in FPC are a really interesting class in their own right and have been much studied (see [11] for some insights). What I want to mention today is that just like every formula of FP is equivalent to one of $L^\omega_{\infty\omega}$, so every formula of FPC is equivalent to one of $C^\omega_{\infty\omega}$—the *infinitary logic with counting*. The idea, as in the former case, is that recursion can be unfolded into an infinitary disjunction while keeping the number of variables bounded.

Q: So, how does counting appear in the infinitary logic?

A: This is a good question and one of the reasons for considering the translation of FPC because the counting mechanism in $C^\omega_{\infty\omega}$ is much easier to explain than the one used in FPC. First of all, C^k is the logic obtained from first-order logic by *(i)* allowing *counting quantifiers*, i.e. we can write $\exists^i x\varphi$ which is to be read as "there exist at least i elements x such that φ"; and *(ii)* restricting ourselves to formulas using only the variables $x_1, \ldots, x_k$.

Q: I suppose, as before, we don't really care about the names of the variables. The important thing is that we are restricting ourselves to formulas in which no subformula has more than k free variables. That is to say, we are again talking about syntactic width.

A: Yes, but the counting quantifiers make a difference. Note that in the formula $\exists^i x\varphi$, the i is some constant. To be precise, there is one such quantifier for each natural number.

Q: But then the counting quantifier $\exists^i$ can be replaced by a sequence of i ordinary existential quantifiers, so the logic C^k is a fragment of first-order logic.

A: That's right, but replacing the counting quantifiers in this way blows up the number of variables. So, while it is true that C^k is a fragment of first-order logic, it is not contained in any fragment of bounded syntactic width.

The fact we are interested in is that for each formula φ of FPC there is a constant k such that if two structures $\mathbb{A}$ and $\mathbb{B}$ are not distinguished by any formula of C^k then they cannot be distinguished by φ: either φ is true in both or false in both. Let's write $\mathbb{A} \equiv^k \mathbb{B}$ to denote that the two structures satisfy the same sentences of C^k.

Q: This reminds me of the discussion about FP and $L^\omega_{\infty\omega}$ we mentioned earlier. There, I remember it was an important result that the corresponding equivalence for L^k was itself definable in FP.

A: Similar facts can be established about FPC and the C^k-equivalence relations. I recommend the book length treatment of this by Martin Otto [32]. In some ways the C^k-equivalence relations are much more interesting though. It turns out that the family of equivalence relations $\equiv^k$ has many natural characterizations. It pops up quite independently in many contexts. It goes by the name of the Weisfeiler-Leman equivalences and has equivalent characterizations in terms of combinatorics, logic, algebra and linear optimization. There is a vast literature on them if you want to read more. A starting point might be these papers [25, 15]. There is also increasing interest in the topic from people working in the machine learning community [29].

From my point of view, each $\equiv^k$ is an approximation of the graph isomorphism relation, with the approximation getting finer as k increases.

Q: And this refinement process doesn't stop at any finite k? I suppose if it did, we would know that graph isomorphism is decidable in polynomial time, since each individual relation $\equiv^k$ is definable in FPC and hence in P.

A: In fact, we do know that there is no k for which $\equiv^k$ is the same as isomorphism. This was proved by Cai, Fürer and Immerman [8] in their landmark paper which showed that FPC is strictly weaker than P.

Q: So, is this parameter k one of the notions of width you were talking about?

A: It is. We sometimes speak of the Weisfeiler-Leman dimension of a graph G to denote the smallest k for which C^k distinguishes G from all non-isomorphic graphs. We also say that a class C of structures has *bounded counting width* (see [16]) if it is definable in $C^k_{\infty\omega}$. I now want to give you another characterization of the equivalence relation $\equiv^k$ that links back to our discussion of conjunctive queries.

4 Counting Homomorphisms

A: For a pair of graphs G and H, let's write $\hom(H, G)$ to denote the *set* of homomorphisms from H to G and $|\hom(H, G)|$ for the cardinality of this set. It is a classical result of Lovász [28] that two finite graphs G_1 and G_2 are isomorphic if, and only if, $|\hom(H, G_1)| = |\hom(H, G_2)|$ for all finite graphs H. Dvořák [21] showed a similar characterization of the equivalence relations $\equiv^k$. To be precise, $G_1 \equiv^k G_2$ if, and only if, $|\hom(H, G_1)| = |\hom(H, G_2)|$ for all finite graphs H of treewidth less than k.

Q: I see. This then connects *counting width* to *treewidth*. And, because we are talking of homomorphisms, it takes us back to conjunctive queries.

A: Yes, for a graph H of treewidth less than k, we can write a conjunctive query φ_H with at most k variables that says of a graph G that *there exists* a homomorphism from H to G. Can you see how we might modify this to a formula that *counts* the number of homomorphisms?

Q: I would do something like this. Convert φ_H into prenex normal form: say $\exists \mathbf{x} \theta$ and then if we want to say that there are at least i homomorphisms, I would then change it to $\exists^i \mathbf{x} \theta$. But, wait a minute, can I do that? In the definition of C^k, we had counting quantifiers to count the number of instantiations of a variable x satisfying a formula θ. Here I want to count the number of instantiations of a tuple $\mathbf{x}$.

A: That's not a problem. One can show that a quantifier counting tuples can be defined using just ordinary counting quantifiers without increasing the number of variables (see [15]). So, we can treat your formula $\exists^i \mathbf{x}\theta$ as a formula of C^k and Dvořák's theorem tells us that if a pair of graphs G and H is distinguished by any formula of C^k then they are distinguished by one of this particular form, which I will call a *counting conjunctive query* of width k.

Q: And then, it easily follows that any formula of $C^k_{\infty\omega}$ is a (possibly infinite) Boolean combination of counting conjunctive queries of width k. I suppose this is a kind of normal form for this logic. That's neat.

A: Dvořák's theorem (and indeed, Lovász') can be put in a much more general framework, in the language of category theory.

Q: I once had a discussion [6] with Yuri and Andreas about the merits of category theory. Are you a fan or a sceptic?

A: I am not a category theorist myself but I have had the fortune to work with some excellent experts on the topic. In particular, in a paper I wrote with Samson Abramsky and my student Pengming Wang [2], we defined what we called the *pebbling comonad*. Essentially this is a way of transforming a structure $\mathbb{A}$ to a structure $P_k\mathbb{A}$ which we can think of as the (infinite) tree unfolding of $\mathbb{A}$ of width k. The transformation is a *comonad* which means it gives rise to a category which we call the *Kleisli category* of P_k in which a morphism from $\mathbb{A}$ to $\mathbb{B}$ is a (standard) homomorphism from $P_k\mathbb{A}$ to $\mathbb{B}$. It turns out that such a morphism exists precisely when every k-variable conjunctive query that is true in $\mathbb{A}$ is true in $\mathbb{B}$. What's more, there is an isomorphism between $\mathbb{A}$ and $\mathbb{B}$ in this category if, and only if, $\mathbb{A} \equiv^k \mathbb{B}$. Thus, in a precise sense, equivalence in C^k is the isomorphism relation that corresponds to morphisms that preserve k-variable conjunctive queries.

Q: That neatly ties in the different notions of width we have been looking at. And, I suppose treewidth is also a natural part of this picture.

A: It turns out that a structure $\mathbb{A}$ admits a *coalgebra* for the comonad P_k if, and only if, it has treewidth less than k.

Q: You said that you obatin a generalization of the theorems of Dvořák and of Lovász from this comonad?

A: Not from this particular comonad, but we do know that this is an example of a much more general phenomenon. Say that a locally finite category $\mathcal{A}$ is *combinatorial* if two objects a and b in $\mathcal{A}$ are isomorphic if, and only if, $|\hom(c, a)| = |\hom(c, b)|$ for all objects c of $\mathcal{A}$. In a joint paper of mine with Tomáš Jakl and Luca Reggio [13], we give fairly general conditions in terms of the existence of pushouts and a proper factorization system for a category to be combinatorial. The theorems of Dvořák and of Lovász and some others from the literature [22]

are examples. In particular, the theorem of Dvořák follows from showing that the Eilenberg-Moore category of the comonad P_k satisfies these conditions.

Q: Why did you call P_k the *pebbling* comonad?

A: The name comes from *pebble games* which are a very common construct in finite model theory. There is a k-pebble one-sided game that characterizes the relation $\mathbb{A} \Rightarrow^k \mathbb{B}$ which holds if every k-variable conjunctive query true in $\mathbb{A}$ is true in $\mathbb{B}$ and there is a k-pebble bijection game which characterizes the relation $\mathbb{A} \equiv^k \mathbb{B}$. The rules of the game involve two players called *Spoiler* and *Duplicator* placing pebbles in turn on the elements of the structure. Hence the name pebble games. For an account of how these games gave rise to the pebbling comonad I recommend reading this paper [12].

Q: You have given me a lot of material to read today.

A: We have covered a lot of ground. Essentially, we have seen that there is a notion of width that gives a gradation of the relation of homomorphism and a corresponding gradation of the relation of isomorphism. These gradations have many different characterizations. For homomorphisms, we looked at the number of variables in a conjunctive query; the treewidth of graphs and the number of pebbles in a one-sided game. For isomorphisms, we have the number of variables in a counting logic formula, the Weisfeiler-Leman dimension of graphs and the number of pebbles in a two-sided bijection game. These are the notions of width we've talked about and they are all intertwined.

I do have another related notion of width I want to introduce you to. This is what I call the *support* of gates in circuits and it involves some surprising connections. This will take us on a journey through some material on circuit complexity and lead us to some quite amazing new results. Do you want to carry on?

Q: I'm up for it.

5 Circuits and Supports

A: I don't know if you are familiar with the basic setup of circuit complexity. In complexity theory, decision problems are languages, for example over a binary alphabet, so $L \subseteq \{0,1\}^*$. You can consider the nth *slice* of the language (i.e. the membership problem in L of strings of length n) as a Boolean function $L_n : \{0,1\}^n \to \{0,1\}$ and any such finite Boolean function can be computed by a circuit C_n. The circuit is made of inputs labelled with Boolean variables $x_1, \ldots, x_n$ and gates labelled by Boolean functions like AND, OR, NOT and sometimes richer functions such as threshold or majority gates and with one designated output gate.

Q: This is all fairly clear. Indeed, we can define such a family of circuits for any language L, whether decidable or not.

A: That's true. To restrict ourselves to decidable languages we would have to impose a *uniformity* condition specifying that C_n is computable from n. But, the hope in circuit complexity is that we can deduce something about the complexity of L by combinatorial properties of the circuits C_n. In particular, if L is in the complexity class P, then there is a circuit family $(C_n)_{n\in\omega}$ deciding L where the size of C_n is bounded by some polynomial in n and moreover the function taking n to C_n is itself computable in polynomial time. The hope is that for NP-complete problems we might be able to show the impossibility of circuits of polynomial size, without worrying about uniformity.

But, as a logician and a finite model theorist, I am interested in decision problems which are not necessarily sets of strings but classes of abstract structures—say graphs.

Q: Of course. The standard way to present a graph as input to a Boolean circuit would be by its adjacency matrix. So, the nth slice of a graph class C would be the collection of graphs in C which have n vertices and this is given by a Boolean function $f_n : \{0,1\}^{n^2} \to \{0,1\}$ and computed by a circuit with n^2 Boolean inputs, one for each entry in the adjacency matrix.

A: Not every such function is a slice of graph class, though. Do you see why?

Q: I do. A given graph can be represented by an adjacency matrix in more than one way. It depends how we order the vertices. So the function f_n might take different values on different representations of the same graph. What we want to do is define an equivalence relation $\sim$ on strings in $\{0,1\}^{n^2}$ where two strings are equivalent whenever they represent the same graph and then require that $f_n(x) = f_n(y)$ whenever $x \sim y$. Isn't this the same thing as saying the f_n is invariant under graph isomorphisms?

A: It is. It will be useful to describe this as a permutation invariance property of f_n. Recall that S_n is the symmetric group on n elements: the collection of all permutations of the set $\{1,\ldots,n\}$. Thinking of a string $x \in \{0,1\}^{n^2}$ as a matrix $(x_{ij})_{i,j\in[n]}$, for any permutation $\pi \in S_n$ define $\pi(x)$ to be the string $y = (y_{ij})_{i,j\in[n]}$ where $y_{ij} = x_{\pi(i)\pi(j)}$. Then, say f_n is invariant if $f(x) = f(\pi(x))$ for all $x \in \{0,1\}^{n^2}$ and $\pi \in S_n$.

Q: I can see that this is the same as what I was defining. A permutation in S_n is just a re-ordering of the vertices of the graph and applying the permutation to the adjacency matrix as you have defined means it's still an adjacency matrix for the same graph. So, a family of circuits $(C_n)_{n\in\omega}$ where each C_n computes a Boolean function $f_n : \{0,1\}^{n^2} \to \{0,1\}$ that is invariant defines exactly an isomorphism-

closed class of graphs. And this class is in P if, and only if, the family of circuits satisfies the conditions you mentioned before.

A: Of course it is not obvious, given a circuit C_n whether or not it is invariant. So, we define a syntactic condition that guarantess invariance. Say that C_n is *symmetric* if any permutation in S_n applied to its inputs $(x_{ij})_{i,j\in[n]}$ can be extended to an *automorphism* of C_n. In other words, for each $\pi \in S_n$, there is an automorphism of C_n which, in particular, takes any input labelled x_{ij} to $x_{\pi(i),\pi(j)}$.

Q: . I can see that if a circuit is symmetric, and the automophisms fix the output gate, then the function it computes is invariant. Is the converse true?

A: Well, yes and no. No, in the sense that there are definitely circuits computing an invaraint function that are not symmetric. Yes, in the sense that every invariant function is computed by *some* symmetric circuit.

For the first statement, consider the following example. Take a circuit that simply takes the conjunction of all its inputs. So, it consists of a single AND gate, which is also the output gate and all inputs x_{ij} feed into it. It evaluates to 1 if, and only if, the input graph has all possible edges. This is clearly invariant and symmetric. But, we can compute the same function with a circuit using only AND gates of fan-in 2 by making a binary tree of the gates with the root as the output gate and the inputs at the leaves. This computes the same function and is therefore invariant but clearly not symmetric in the sense we defined.

Q: And, for the second statement, I suppose we can take an invariant circuit and convert it into a symmetric one computing the same function.

A: We can. The construction is a bit technical but not difficult. In particular, it depends to some extent on what functions we allow at the gates. But it is worth pointing out that it involves, in general, an exponential blow-up in the size of the circuit. One thing that the above example shows us though is that fan-in matters. To talk meaningfully of symmetric circuits we should allow gates of unbounded fan-in.

Q: Do we know that the exponential blowup is necessary? Or is it simply that the best way we currently know of doing this translation is exponential?

A: We can show it's impossible to do better. This is not too hard to show if all you have as gates are the standard Boolean basis of AND, OR and NOT gates. It's a bit more involved if you allow threshold or majority gates. In either case, it is a consequence of results in [3] which I want to come back to in a little bit.

Q: So, why are we interested in symmetric circuits? It seems like the real condition we want to think about is invariance and you have just told me that the restriction to symmetric circuits is sufficient but not necessary for invariance.

A: The interest comes from looking at definability in logic. Suppose you have a first-order sentence φ in the language of graphs. Then for each n you can produce from φ a circuit C_n which decides for an input n-vertex graph G whether G satisfies φ. Moreover, the circuit you get from φ is of polynomial-size, bounded depth and it is *symmetric*.

Q: Yes, I see that. I believe there was some work on this in the 1980s [19].

A: That's right. Also some interesting work extending it to infinitary logic by Otto [31]. When you add counting quantifiers to the mix, however, things get even more interesting. Do you see what happens if you start with a formula φ of C^k?

Q: If I follow the inductive construction of the circuit, where existential quantifiers become big OR gates and universal quantifiers are big AND gates...Oh, I see. For a counting quantifier it makes sense to introduce threshold gates. So, formulas of C^k will translate into circuits of bounded depth but with threshold gates. And the size of the circuits is $O(n^k)$.

A: Very good. And formulas of FPC translate into families of symmetric circuits of polynomial size with threshold gates, but possibly of unbounded depth. What's more, we can get a converse statement. Any class of graphs decided by a polynomial-time uniform family of symmetric circuits with threshold gates is definable in FPC. So, it is an exact circuit characterization of the logic FPC. This is the main result of [3].

Q: That really bolsters your claim that FPC is a natural class to consider. Does the logic $C^{\omega}_{\infty\omega}$ give you a non-uniform version of this?

A: Not quite, but it does give us something interesting. The issue of non-uniformity is dealt with in [3] by allowing *numerical relations*, and we'll leave that aside for now. When we start with infinitary formulas in $C^{\omega}_{\infty\omega}$, there is no guarantee that the resulting circuit will be of polynomial size. But, it is necessarily of polynomial *orbit size*.

Q: What is orbit size?

A: Suppose C_n is a symmetric circuit taking as input graphs on n vertices. Then S_n embeds into the automorphism group of C_n by definition. For a gate g in C_n, the *orbit* of g is the collection of gates g' which are mapped to g by some automorphism. The orbit size of C_n is then just the maximum size of any orbit of a gate in C_n.

Q: Yes, I think I can see the connection. If I start with a sentence φ of $C^k_{\infty\omega}$, I get an equivalent formula φ_n of C^k for any fixed size n of graphs. I suppose the size of φ_n is not necessarily bounded by a polynomial in n which is why we don't get polynomial-size circuits.

A: But, the circuit we get from φ_n has orbits of size at most n^k. This is because there is a gate $\psi[\mathbf{a}]$ in the circuit for each subformula ψ of φ_n along with an assignment of vertices $\mathbf{a}$ to the free variables of ψ. The orbit of the gate $\psi[\mathbf{a}]$ is then obtained by varying the assignment $\mathbf{a}$.

Q: I see. And since any subformula has at most k free variables, there are at most n^k assignments and this gives you an upper bound on the orbit size. I like the way we have circled back to the notion of syntactic width.

A: In fact, symmetric circuits give us another related notion of width. That is to say, there is a combinatorial parameter of symmetric circuits which is a counterpart to the syntactic width of formulas. It plays a crucial role in the translation [3] of poly-size symmetric circuits into FPC. It's what we call *supports*.

But, before we get to that, I wanted to note that the translation from symmetric circuits into FPC or into $C^{\omega}_{\infty\omega}$ means that when we prove something is not expressible in these logics, it is a circuit lower bound result. Indeed, we can now think of the bijection games that are used to prove inexpressibility in FPC as a method for proving circuit lower bounds. You can read about this in some generality in my paper with Greg Wilsenach [18] where we use this method to prove lower bounds on *algebraic circuits*.

Q: What are algebraic circuits?

A: We'll come back to that in a moment. First I want to tell you about supports.

Do you remember the orbit-stabilizer theorem?

Q: You mean from permutation group theory? Let's see: if you have a group Γ which is a subgroup of S_n, the Γ-orbit of an element $i \in [n]$ is the collection of elements j which map to i under some permutation in Γ and the stabilizer of i is the subgroup Δ_i of Γ consisting of those permutations that fix i. The theorem then says that the index $[\Gamma : \Delta_i]$ is exactly the size of the orbit of i.

A: That's right. More generally, we consider any action of Γ on a set X (it doesn't just have to be $[n]$) and we can look at the size of the orbit of any element $x \in X$. In our case, I want to consider the case where $\Gamma = S_n$ and we look at its action on a symmetric circuit C_n and think about the orbits of gates.

Q: I see, so if the circuit has orbit size at most n^k, that means that the stabilizer group of any gate g has index in S_n of at most n^k. In other words the stabilizer group is big. Many permutations fix g. I can see this is true for the circuits we constructed from formulas. There each gate was of the form $\psi[\mathbf{a}]$ and clearly any permutation of the vertices that fixes all vertices in the tuple $\mathbf{a}$ fixes the gate. Since the tuple is small (at most k elements) there are lots of such permutations. What you're saying is that something like this is true in all symmetric circuits, not just those that we construct from formulas.

A: In fact, it turns out something stronger is true. Say that a set $X \subseteq [n]$ is a support of a group $\Delta \leq S_n$ if every permutation that fixes all the elements of X is in Δ.

Q: So, just like the elements in the tuple **a** form a support for the stabilizer group of the gate $\psi[\mathbf{a}]$.

A: Like that. But, we can show that in any symmetric circuit in which all gates have an orbit of size at most n^k, the stabilizer of every gate has support of size at most k. So, we associate with a circuit C a measure we call its support size, which is the size of the largest support of any gate. This is the parameter that is the counterpart to width, and which we use to show lower bounds.

In particular, we can directly use bijection games to argue that some properties are not decidable by symmetric circuits with small support. This is just an adaptation of their use in showing that the properties are not definable in $C^{\omega}_{\infty\omega}$. But, we can also use them in the context of circuits where there is no immediate connection to logic.

Q: Are we now turning to algebraic circuits?

6 Algebraic Circuits

A: Algebraic circuits, sometimes also called arithmetic circuits, are a model of computation used to study computations over a field (such as the real numbers or complex numbers) where we treat addition and multiplication as operations of unit cost.

Formally, an algebraic circuit over a field K and a set of variables X is a directed acyclic graph where every input (i.e. node of indegree 0) is labelled by an element of X or an element of K, and every internal node is labelled either $+$ (a sum gate) or $\times$ (a product gate). A distinguished *output* gate can then be seen as computing a polynomial in the ring $K[X]$. The minimal size circuit computing a polynomial p tells us the minimal number of operations needed to compute p. Another way of thinking about it is that the circuit is a compact representation of the polynomial. Potentially much more compact than the usual representation as a sum of monomials.

Q: This seems quite different from the world of Boolean circuits. We are no longer looking at the complexity of decision problems or languages.

A: Still, there is an algebraic analogue of the question of whether $\mathsf{P} \neq \mathsf{NP}$, which is known as Valiant's conjecture. It is usually stated in the form $\mathsf{VP} \neq \mathsf{VNP}$.

Q: What are VP and VNP?

A: Formally, we are looking at the complexity of *families* of polynomials $(p_n)_{n\in\omega}$ with $p_n \in K[x_1, \ldots, x_n]$. The family is in VP if there are circuits C_n computing p_n and the size of the circuits grows only polynomially in n. VNP has a definition as families of polynomials whose coefficients can be computed by polynomial-size circuits. There is a vast literature on algebraic circuits and this tutorial [34] is an excellent entry point.

Q: I take it you want to define a symmetric version of these circuits. But, the inputs here are no longer graphs. What are the symmetries you are looking at?

A: In many of the families of polynomials that are often studied in the field of algebraic complexity, the variables are the entries of a matrix. That is, we can index the variable set X as $X = \{x_{ij} \mid 1 \leq i \leq m; 1 \leq j \leq n\}$. In particular, when they form a square matrix, i.e. $m = n$. The classic examples are the *determinant polynomial*:

$$\mathrm{Det}(X) = \sum_{\sigma\in S_n} \mathrm{sgn}(\sigma) \prod_{i\in[n]} x_{i\sigma(i)};$$

and the *permanent polynomial*:

$$\mathrm{Per}(X) = \sum_{\sigma\in S_n} \prod_{i\in[n]} x_{i\sigma(i)}.$$

Q: Why are these *classic* examples?

A: They are the most studied in the field of algebraic complexity. The determinant is known to be in VP and the permanent is VNP-complete. This means that Valiant's conjecture is equivalent to the statement that the permanent is not in VP. Indeed, much work in algebraic complexity is about understanding the difference between these two families of polynomials.

Q: And I take it there are differences in terms of symmetric circuits?

A: Yes, let's define what we mean by symmetry. Say we have a circuit C computing a polynomial $p \in K[X]$ and Γ is a group of permutations of X. We say that $p[X]$ is Γ-invariant if for each $\pi \in \Gamma$ the polynomial p^π we get by permuting the variables of p according to Γ is identical to p.

And, we say that the circuit C is Γ-symmetric if every permutation of Γ applied to the inputs of C can be extended to an automorphism of C.

Q: So it is very much like the Boolean symmetric circuits we talked about. For $\mathrm{Det}(X)$ and $\mathrm{Per}(X)$, the set $X = \{x_{ij} \mid 1 \leq i, j \leq n\}$ and so the full group of permutations of X looks like S_{n^2}. What is the group Γ?

A: We could, for example, consider the group S_n acting simultaneously on the rows and columns of the matrix as we did for Boolean circuits. So, Γ is the

group of permutations $\pi \in S_n$ acting on X by the action that takes x_{ij} to $x_{\pi(i)\pi(j)}$. Then, both $\mathrm{Det}(X)$ and $\mathrm{Per}(X)$ are Γ-invariant. The determinant and permanent of a matrix are not changed if you permute the entries by applying the same permutation to the rows and columns. We say that these polynomials are *square symmetric*.

Q: And what can we say about symmetric circuits for them?

A: We showed [17] that there are polynomial size square symmetric circuits for $\mathrm{Det}(X)$ but provably none for $\mathrm{Per}(X)$. This shows an exponential separation in the complexity of these two families of polynomials in a restricted model, which is the first such separation I know of. The method of proof for the lower bound really uses the connection with Boolean circuits and $C^{\omega}_{\infty\omega}$. In fact, we show that any family of circuits for $\mathrm{Per}(X)$ has exponential *orbit size*.

Q: I will have to read up more about it. When we talked of Boolean circuits, the inputs were adjacency matrices of graphs. So the natural notion of symmetry was permutations of the vertices, which corresponds to what you called square symmetric. For general matrices, is there any reason to privilege this particular permutation group?

A: It's still quite a natural collection of permutations to consider, but you are right, we could consider others. For instance, consider the group $S_n \times S_n$, i.e. the direct product of S_n with itself. This has an action on the matrix X of variables where the pair (π, σ) takes x_{ij} to $x_{\pi(i)\sigma(j)}$. In other words we can apply a permutation to the rows of the matrix and possibly a different permutation to its column. Then $\mathrm{Per}(X)$ is invariant under this group action, but $\mathrm{Det}(X)$ is not. We say that $\mathrm{Per}(X)$ is *matrix symmetric* but $\mathrm{Det}(X)$ is not.

Q: I see that. In fact the determinant of a matrix is not changed if you apply one permutation to the rows and another permutation of the same sign to the columns. But, if you apply permutations of different signs to the rows and columns, it would change the sign of the determinant.

A: When you evaluate a polynomial $p(X)$ at values of X which are in the set $\{0, 1\}$, you can understand it as computing a parameter of a graph, provided that $p(X)$ has the right symmetries. So, if $p(X)$ is square symmetric, then substituting for X the adjacency matrix of a graph gives a value which really only depends on the graph.

Q: What about when it is matrix symmetric?

A: In that case, it is best to think of a $\{0, 1\}$ valuation of X as putting in the *biadjacency matrix* of a bipartite graph. That is, we have one set A of vertices corresponding to the rows and another set B corresponding to the columns. Then, any graph property (or more generally any graph parameter) should be invariant under permuting A and B separately.

Q: In this case, there is no reason to restrict ourselves to square matrices. The sets A and B could have different sizes.

A: They could. So, let's go back to the more general case where $X = \{x_{ij} \mid 1 \leq i \leq m; 1 \leq j \leq n\}$ and consider the action of the group $\Gamma = S_m \times S_n$ on this where (π, σ) takes x_{ij} to $x_{\pi(i)\sigma(j)}$ for $\pi \in S_m$ and $\sigma \in S_n$. It turns out that we can give a fairly complete characterization of families of polynomials that have Γ-symmetric circuits in this sense. Again, let's call them *matrix symmetric*.

Q: What's this complete characterization?

A: It's in terms of what we call *homomorphism polynomials*. So, fix an $a \times b$ matrix M with entries in the field K and the set of variables $X = \{x_{ij} \mid 1 \leq i \leq m; 1 \leq j \leq n\}$. Note that a and b are not related to the parameters m and n. We now define the polynomial

$$\hom_M(X) = \sum_{f:[a]\to[m],g:[b]\to[n]} \prod_{i\in[a],j\in[b]:M_{i,j}\neq 0} M_{i,j} x_{f(i),g(j)}.$$

Q: Why is this called a homomorphism polynomial?

A: Consider the special case when the matrix M only has entries in $\{0, 1\}$. We can then think of it as the biadjacency matrix of a graph F_M on a pair of sets of vertices A of size a and B of size B. Then $\hom_M(X)$ is a polynomial in the variables X which, when we substitute for X the biadjacency matrix of a bipartite graph G with vertiex bipartition (U, V) evaluates to the number of homomorphisms from F_M to G. That is, only counting homomorphisms that take A into U and B into V.

Q: I can see that. So, we generalize the homomorphism counting parameter to matrices M which are not necessarily $\{0, 1\}$. I suppose I can think of such matrices as weighted bipartite graphs.

A: And then $\hom_M(X)$ is in some sense counting "*weighted homomorphisms*". These are polynomials that come up quite naturally in algebraic complexity. For instance [20] argue that they give rise to the first natural VP-complete families.

Q: Nice. And how do they come up in characterizing matrix-symmetric families of polynomials?

A: This is a theorem we proved in some recent work that I did with Benedikt Pago and Tim Seppelt [14].

Theorem 1. *A family of polynomials $p(X)$ is computed by a family of matrix symmetric circuits of polynomial orbit size if, and only if, there is a k such that each p is a linear combination of homomorphism polynomials $hom_M(X)$ of graphs F_M of treewidth less than k.*

Q: So bounded treewidth pops up again in the characterization. We're back at the same combinatorial notions of width.

A: Yes, you could look at this theorem as an analogue of the normal form for $C^{\omega}_{\infty\omega}$ we obtain as a consequence of Dvořák's result. So, $C^{\omega}_{\infty\omega}$ defines exactly the properties of graphs decided by families of symmetric circuits with polynomial size orbits. And, from Dvořák's theorem we get that all such properties are obtained as infinite Boolean combinations of homomorphism counts from a class of graphs of bounded treewidth.

Q: I do see the analogy. But, we are looking at somewhat different notions of symmetry.

A: It is indeed just an analogy at the moment. I don't see how you could derive one result from the other. But it is compelling and shows the natural connection between notions of width and symmetry.

7 Conclusion

Q: Well, we have gone over a lot of material and you have given me really a lot to go away and read and digest. I think it is time for a break.

A: We have covered a lot of ground, some of it very new. The take away is that *width* in its various forms is a measure of complexity (of graphs, of formulas, of circuits) which ties to many natural and independently discovered measures in combinatorics, logic and complexity. It turns out to be useful in proving surprising unconditional lower bounds in circuit complexity and other areas.

Q: I will go away and chew on that.

A: And there is so much ground that we didn't cover. There are notions of width in the field of *constraint satisfaction*. There is *submodular width* and *hypertree width*. They are not unrelated to what we have talked about but we didn't have time to go over them.

Q: It will have to be another time. I look forward to carrying on our conversations.

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[2] S. Abramsky, A. Dawar, and P. Wang. The pebbling comonad in finite model theory. In *32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–12, 2017. doi:10.1109/LICS.2017.8005129.

[3] M. Anderson and A. Dawar. On symmetric circuits and fixed-point logics. *Theory Comput. Syst.*, 60(3):521–551, 2017.

[4] L. Birkedaal and B. Köning, editors. *ACM/IEEE Symp. Logic in Computer Science*. IEEE, 2025.

[5] A. Blass and Yu. Gurevich. Two forms of one useful logic: Existential fixed point logic and liberal datalog. *Bull. EATCS*, 95:164–182, 2008.

[6] A. Blass and Yu. Gurevich. Who needs category theory? *Bull. EATCS*, 124, 2018.

[7] S. Bova and H. Chen. How many variables are needed to express an existential positive query? *Theory Comput. Syst.*, 63:1573–1594, 2019. doi:10.1007/S00224-018-9884-Z.

[8] J-Y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410, 1992.

[9] A.K. Chandra and P.M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *STOC*, pages 77–90, 1977.

[10] V. Dalmau, Ph.G. Kolaitis, and M.Y. Vardi. Constraint satisfaction, bounded treewidth, and finite variable logics. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming, CP'02*, Lecture Notes in Computer Science, pages 311–326. Springer-Verlag, 2002.

[11] A. Dawar. The nature and power of fixed-point logic with counting. *ACM SIGLOG News*, 2(1):8–21, 2015.

[12] A. Dawar. Constraint satisfaction, graph isomorphism, and the pebbling comonad. In Alessandra Palmigiano and Mehrnoosh Sadrzadeh, editors, *Samson Abramsky on Logic and Structure in Computer Science and Beyond*, pages 671–699. Springer Verlag, 2023.

[13] A. Dawar, T. Jakl, and L. Reggio. Lovász-Type Theorems and Game Comonads. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*. IEEE, 2021. doi:10.1109/LICS52264.2021.9470609.

[14] A. Dawar, B. Pago, and T. Seppelt. Symmetric algebraic circuits and homomorphism polynomials. arXiv 2502.06740, 2025.

[15] A. Dawar and D. Vagnozzi. Generalizations of k-Weisfeiler-Leman stabilization. *Moscow Journal of Number Theory and Combinatorics*, 9:229–252, 2020.

[16] A. Dawar and P. Wang. Definability of semidefinite programming and Lasserre lower bounds for CSPs. In *2017 32nd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*. IEEE, 2017. doi:10.1109/LICS.2017.8005108.

[17] A. Dawar and G. Wilsenach. Symmetric Arithmetic Circuits. In *47th International Colloquium on Automata, Languages, and Programming*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 36:1–36:18, 2020. doi:10.4230/LIPIcs.ICALP.2020.36.

[18] A. Dawar and G. Wilsenach. Lower bounds for symmetric circuits for the determinant. In *13th Innovations in Theoretical Computer Science Conference, ITCS*, volume 215 of *LIPIcs*, pages 52:1–52:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ITCS.2022.52.

[19] L. Denenberg, Y. Gurevich, and S. Shelah. Definability by constant-depth polynomial-size circuits. *Information and Control*, 70:216–240, 1986.

[20] A. Durand, M. Mahajan, G. Malod, N. de Rugy-Altherre, and N. Saurabh. Homomorphism polynomials complete for VP. *Chic. J. Theor. Comput. Sci.*, 2016.

[21] Z. Dvořák. On recognizing graphs by numbers of homomorphisms. *Journal of Graph Theory*, 64:330–342, 2010.

[22] M. Grohe. Counting Bounded Tree Depth Homomorphisms. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 507–520. ACM, 2020. doi:10.1145/3373718.3394739.

[23] I. M. Hodkinson. Finite variable logics. *Bull. EATCS*, 51:111–140, 1993.

[24] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[25] Sandra Kiefer. The Weisfeiler-Leman algorithm: an exploration of its power. *ACM SIGLOG News*, 7(3):5–27, 2020.

[26] Ph. G. Kolaitis and M. Y. Vardi. Fixpoint logic vs. infinitary logic in finite-model theory. In *LICS*, pages 46–57, 1992.

[27] A. Livchak. The relational model for process control. *Automated Documentation and Mathematical Linguistics*, 4:27–29, 1983.

[28] L. Lovász. Operations with structures. *Acta Math. Acad. Sci. Hungar.*, 18:321–328, 1967.

[29] Ch. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe. Weisfeiler and Leman go neural: Higher-order graph neural networks. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*, pages 4602–4609. AAAI Press, 2019. doi:10.1609/AAAI.V33I01.33014602.

[30] Jaroslav Nešetřil and Patrice Ossona De Mendez. *Sparsity: graphs, structures, and algorithms*, volume 28. Springer Science & Business Media, 2012.

[31] M. Otto. The logic of explicitly presentation-invariant circuits. In *Computer Science Logic, 10th International Workshop, CSL '96, Annual Conference of the EACSL*, pages 369–384, 1996.

[32] M. Otto. *Bounded Variable Logics and Counting — A Study in Finite Models*, volume 9 of *Lecture Notes in Logic*. Springer-Verlag, 1997.

[33] B. Poizat. Deux ou trois choses que je sais de L_n. *Journal of Symbolic Logic*, 47(3):641–658, 1982.

[34] A. Shpilka and A. Yehudayoff. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends in Theoretical Computer Science*, 5(3-4):207–388, 2010. doi:10.1561/0400000039.

[35] M. Y. Vardi. The complexity of relational query languages. In *Proc. of the 14th ACM Symp. on the Theory of Computing*, pages 137–146, 1982.

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 147

Logic For High School Mathematics Teachers

R. Ramanujam
Azim Premji University, Bengaluru, India
ramanujam.r@apu.edu.in

Abstract

This article addresses the question of whether concepts from formal logic can meaningfully contribute to mathematics education at the school level. We report on an attempt to work with high school teachers of mathematics in India on notions from logic such as: the notion of truth relative to a structure, construction of models for a set of sentences, consistency of procedures in algebra, and reasoning about algorithms used in school and comparing those algorithms systematically. Broadly, we tried to engage teachers in using ideas from logic to help with students' misconceptions, and to help students with reasoning about procedures.

1 Background

Our story begins at a workshop for high school teachers, the theme of which was *problem solving in algebra and geometry*. Typically, in such workshops teachers engage in problem solving, discuss strategies used in problem solving, and then move on to discussions on their use in classroom pedagogy.

In a particular session, there was a discussion on problem solving techniques that had come up in different contexts, and while we were listing them, one teacher V made the remarkable assertion that there was no unifying method combining these techniques, especially in the context of algebra. He contended that they were "tricks" and best learned as such. Some teachers agreed with him, while several others disputed this assertion. However, when called upon to present a unifying method, they found it hard to even argue that any such method exists, let alone present one, much to the glee of the (by now growing) camp of "algebra is a collection of tricks" enthusiasts.

At this point, M, another teacher, wondered whether this was a problem specific to algebra, and whether the situation was different with respect to geometric

reasoning. She added that geometry was about proofs, and that should help. I turned to V and asked whether his geometry was also a bunch of tricks. V was emphatic that geometry was "logical," entirely based on axioms and proofs, and hence all the techniques used were unified into one whole. The ensuing discussion led to a consensus among teachers that the axiomatic method in geometry provided a sound basis for problem solving as well.

I intervened and reminded teachers that algebra was entirely axiomatic in its development. A teacher protested that this was true of "abstract algebra" studied at university, but not algebra in school mathematics which was only about "solving equations." This led to a general discussion on the place of school algebra within the broader study of algebra (at university), but it was clear that there was an uneasy realisation among the teachers that algebra could not be axiomatic and at the same time be a bunch of tricks.

School algebra is largely about setting up and solving systems of linear equations, and trying to solve quadratic and cubic equations; all this is carried out over the reals. I presented a proposition to teachers that a uniform algorithm, which could be implemented as a computer program, could solve any of these problems; I asked whether they believed this proposition to be true. An overwhelming subset of the teachers answered yes. When I asked them for reasons why they thought so, they broadly referred to computers being "powerful." One of the dissenters referred to *Gödel's Incompleteness Theorem*: she said that Gödel had shown that we could not even solve "general problems" involving "just arithmetic," so solving difficult cubic equations on the reals "simply could not be done."

The incident led to the realisation that while high school teachers of mathematics see reasoning as being fundamental to mathematics, they do not necessarily believe that there is logical structure in reasoning. Further, they do not perceive any connection between logic and algorithms or procedures. This realisation raises the following questions.

1. Is formal logic indeed necessary, or even helpful, in the teaching / learning of mathematics?

2. Mathematical learning at school involves learning a range of rules and procedures. Can logic help in seeing these as *inference systems* and provide teachers with structure (rather than seeing them as isolated tricks)?

While answers to these questions would eventually have an impact on the larger questions relating to mathematics school curricula and to helping children learn mathematics and logic, our effort was more modest. Our goal was to see whether mathematical logic could help to improve the teachers' own content knowledge and pedagogical content knowledge of mathematics. My own positive conviction

in this regard was tempered by some doubt. A series of workshop opportunities for interacting with mathematics teachers led to a strengthening of this conviction, sharing which is the main aim of this article.

A caveat: this work was not exclusively focussed on logic. I have been working on mathematics school education for a long time and was working with high school teachers of mathematics anyway. The opportunity this provided for discussions about formal logic and introducing mathematical logic to them also led to the identification of a number of ways logic could help to clarify mathematical content to themselves, and to help them address student misconceptions more effectively. These are the insights I attempt to share in this article.

Some of the ideas here were presented in the context of a discussion on school curriculum published in 2023 [18].

2 Logic in School Mathematics

Despite logic and reasoning being considered central to mathematics and computing education, they play a largely peripheral role in high school or undergraduate curricula; at least, this is the case in India.

The content areas of high school mathematics have remained principally unchanged since the days of the industrial revolution: arithmetic, algebra, geometry, trigonometry, probability and statistics, and an introduction to differential and integral calculus. There are a few chapters on combinatorics and propositional logic, but they stand in isolation from the rest of the chapters. In terms of relative space, arithmetic in the early years, algebra and geometry in the middle years, and calculus in the later years occupy the maximum [16]. While there are differences across national curricula, the principal structure remains similar.

In what follows, we focus on the curriculum of the *Central Board of Secondary Education (CBSE)* in India, and that of the *Tamil Nadu State Board of Education* in India, essentially because of our familiarity with them and relative ignorance of syllabi elsewhere. However, even a superficial study of curricula across the world shows broad similarities to the Indian CBSE. In India, all the schools in a state (such as Tamil Nadu) affiliated with the state board follow the textbook of that board. In the case of Tamil Nadu, this constitutes nearly 75 % of the schools. Of the remaining, most are affiliated with the CBSE. Thus the CBSE textbook is used by a number of schools across the country from many states.[1]

While the focus of this article is on logic for school teachers of mathematics, and not on the mathematical content they teach, it is necessary to have an understanding of the way school the mathematics syllabus is structured and textbooks are designed,

[1]The story is more complex, as there are boards of education other than these, but this simplification suffices for our discussion.

in order to understand teacher beliefs about the role of logic in mathematics education.

In this curriculum, the study of logic is principally limited to *geometric reasoning* and *using the language of propositional logic*. We discuss these below.

2.1 Geometric Reasoning

Geometry occupies significant space in the secondary school curriculum, in grades 9 and 10. In grade 9, it constitutes nearly half of the instruction period (75 out of 160 scheduled lessons). In grade 10, it reduces to less than a quarter (30 out of 160). *Proofs* are accorded importance in the study of geometry. There are theorems and proofs in other areas such as number theory and algebra, but these are not as explicitly discussed as they are in geometry. Some algebraic identities are stated without proof, whereas geometric propositions are (almost) always proved.

Geometry begins with a historical introduction to Euclid. The syllabus document [14] states:

> Euclid's method of formalising observed phenomena into rigorous mathematics with definitions, common/obvious notions, axioms/postulates, and theorems. The five postulates of Euclid. Equivalent versions of the fifth postulate. Showing the relationship between axiom and theorem.

Subsequently a number of propositions on lines, angles, triangles, quadrilaterals, and circles are stated and proved. Geometry in grade 10 is similar, with the additional notions of congruence and similarity for triangles and tangents in the case of circles. There are also topics combining these, such as on cyclic quadrilaterals. The proofs are largely rigorous but informal. Some of the proofs, such as the ones using congruence, require some depth of reasoning using many assertions proved earlier. An appendix of each textbook (of grades 9 and 10) discusses the nature of deductive proofs, stressing the role of formal derivations and the distinction between verification and proof of statements.

In this regard, geometry instruction employs logic purposefully even while not stressing on the *language* of logic or on formal deduction in an inference system.

2.2 Propositional Reasoning

The CBSE syllabus for grade 11 includes a unit titled *Mathematical Reasoning*. The document [14] states:

> Mathematically acceptable statements. Connecting words / phrases – consolidating the understanding of "if and only if (necessary and

> sufficient) condition", "implies", "and/or", "implied by", "and", "or", "there exists" and their use through a variety of examples related to real life and Mathematics. Validating the statements involving connecting words – difference between contradiction, converse, and contrapositive.

The textbook chapter introduces the language of propositional logic, and explains the use of connectives very well. The principal focus is on implication: getting students to appreciate the distinction between $p \implies q$ and $q \implies p$, to understand inclusive disjunction and to realise that a single counterexample suffices to falsify a universally quantified statement. Considerable effort is spent in formalising intuitive statements in the language of propositional logic. It is also important to point out that geometry is extensively used as the terrain from which propositions are picked.

For instance, an exercise [15, Chapter 14, page 344] asks the students to *write the following statement in five different ways, conveying the same meaning*.

> If a triangle is equiangular, then it is an obtuse angled triangle.

For someone formalising this as $p \implies q$ (which is what the book teaches the student to do), there are indeed equivalent forms to try, such as $\neg q \implies \neg p$. If a student tries to use the language of geometry starting with a triangle ABC and proceeding further, it is much less clear if they would get anywhere with this exercise. (Equating differently expressed properties by virtue of logical equivalence is an important logical exercise itself. Our remark here is not to downplay this, but to point to the limited use of formalisation.)

2.3 Teacher Beliefs Related to Logic

Given that this is the logical content that teachers are expected to teach, it is not surprising that they consider geometry as the central location of logic in mathematics learning. They see that logic is essentially about seeing how the theorems proved are deduced from clearly stated axioms. Logic is not associated with discovery but with writing in the formal language of mathematics. Indeed, during later discussions, some teachers asserted that logic was a kind of mathematical bureaucracy, arising only out of an insistence on stating things in a strictly formal (and unfriendly) way. This is despite the textbook being relatively informal: in reality, the axioms and inference rules of geometry are never formally spelt out, an informal argument is preferred over a formal deduction in the textbook.

Indeed, the task of reasoning and proving is important in the mathematics classroom, and this is best done in an informal intuitive way, rather than by insisting on formal deductions. There is extensive literature in mathematics education

research on students' reasoning and structure in proofs [9, 11, 19]. Invariably, these researchers point out that developing *intuition* is more important for mathematics learning than formal proofs. So it is not surprising that mathematics teachers are of similar opinions.

We asked teachers whether the language of logic plays any significant role in teaching geometry, or whether we teach deductive systems by way of geometry. Their answers to both of these questions were largely negative. Some authors argue that mathematical logic is not necessary at all for teaching proofs in mathematics [1], and some education researchers offer nuanced arguments on specific aspects of logic being relevant in the teaching of proofs [6].

On the other hand, learning the correct use of propositional connectives and quantifiers unambiguously constitutes logic learning, and this is also indispensable for obtaining fluency in the language of mathematics. However, this is also compromised in classroom practice [10]. Indeed, in our conversations with teachers, many wondered what the use of all this "Boolean logic" was, when they had to teach difficult concepts in differential and integral calculus.

To understand why they have such a reaction to propositional logic, it may be instructive to look at the *placement* of the logic syllabus unit in the curricular structure. The other syllabus units among which Boolean logic is placed involve topics such as Trigonometric Functions, Complex Numbers, the Binomial Theorem, Conic Sections, Frequency Distributions, Limits, and Derivatives. In fact, in the CBSE's textbook, the chapter on propositional logic follows the chapter on limits and derivatives. After solving exercises computing the derivatives of functions such as $x/(\sin x)$, the students encounter the relationship between $p \implies q$ and $q \implies p$. It is hardly surprising that they are left wondering whether this is merely an interlude between differentiation and integration. While students are happy to take any "easy" topic that comes their way, the deeper issues pass them by, and the chapter on propositional reasoning merely becomes another set of (thankfully simple) rules to be learned and forgotten soon. Teachers are broadly in sympathy with this viewpoint as well.

If teachers see very little use for logic in school mathematics, and even see the little that is on offer as being ineffective, should we conclude that logic is irrelevant to school mathematics?

This initiated a different conversation between us: rather than seeing logic as a topic in mathematics to be taught to students, should we explore whether there were demands from the teaching of algebra, geometry, number systems, and so on, that logic could address? This required us to go a little deeper into logic. What followed was a formal introduction for the teachers on first order logic, interpreted over the integers and the reals.

3 The Content

In this section, we describe our efforts in trying to come up with an appropriate *Logic Curriculum* for mathematics teachers. The idea was not to aim for "direct impact" on the system in the sense of a curricular reform or textbook writing, but an indirect one, by enriching teachers' understanding of logic so that they could use it in pedagogic contexts offered by the curriculum and textbook.

The initial ambition was to devise a systematic course for teachers, one that would eventually be approved by the state education machinery. But conducting a course of any significant duration for teachers was difficult for a variety of reasons, the principal one being teachers' availability, and the other being official permissions. Instead, what we did was to incorporate logic sessions in a series of teacher workshops on mathematics. This meant that some of the content had to be repeated, and also that it was not the same set of teachers present in all the workshops. However, there was a core of more than 10 teachers present through all engagements who provided continuity and helped maintain coherence to the effort.

We also need to point out that in India, elementary school teachers have a degree in education but they do not necessarily have a disciplinary specialization. Teachers at the secondary stage are qualified in both education and mathematics. At the senior secondary stage, teachers come with a Master's degree in mathematics. In general, most undergraduate mathematics curricula in India do not include a course on mathematical logic, but incorporate some logic into a foundational course and some set theory into a course on real analysis. Again, when students are grappling with completeness of ordered fields, logic tends to take the backseat. As a result, very few teachers have learned logic formally, nor is it in any way emphasized in their education courses (which center on pedagogy rather than content). In particular, *none* of the teachers in our group had learned mathematical logic formally.

The first step was to identify a logical language in which much of school mathematics can be expressed. First order logic with addition and multiplication (later extended with exponentiation) works for most of algebra, geometry, and number theory in school. At the senior secondary level, when trigonometric functions, limits, and differential calculus are studied, this is inadequate, but our discussions were mainly with high school teachers.

Setting up a language and presenting its syntax and semantics was largely new to teachers. We considered the same sentences interpreted over whole numbers, over integers, and over reals; and the differences were greatly appreciated by teachers. We also saw equations as sentences when quantified appropriately, and steps in solving equations as implications between sentences. What perhaps caught the attention of teachers most was the possibility of algorithms that could check whether an equation had a solution or not. It was also a surprise to them that it

was possible to design algorithms over the reals but not the integers. (Admittedly, these theorems were only stated without proof; however, some understanding was possible mainly because the teachers had the language for it.) We also discussed the idea of correctness proofs of algorithms, with the Euclidean algorithm being a good example. We had hoped to show this proof by formalising it in an interactive theorem proving tool, but this turned out to be too difficult in the time we had.

4 Students' Difficulties

This series of interactions led to teachers identifying a range of mathematical difficulties faced by students where logic might be of significant help. In each of the cases, the teachers felt that they could offer better explanations after their own understanding of mathematical logic. While these difficulties are acknowledged and discussed extensively in the mathematics education literature, what was perhaps novel and interesting in our discussions was in the suggested use of mathematical logic for addressing these difficulties.

1. **The use of variables:** We ask middle school students to solve equations of the form $x + 3 = 5$, and they learn that x is a specific unknown number. Then they go on to consider equations such as $x + y = 5$, when x can be one of many numbers, somehow dependent on the value that y takes. We also ask them to "see" that $x + y = y + x$, but now x can be *any* number whatsoever. There is worse to come when we ask them to consider the line given by the equation $x = k$, where k is some constant [3]. All these examples are legitimate and reasonable uses of x in mathematical contexts, but being syntactically disciplined about quantifying x appropriately depending on the context can solve much of the confusion. For teachers, it was a matter of realizing that we could instead work with sentences such as "Is there an x such that $x + 3 = 5$?" and "Given *any* y, can you find *some* x such that $x + y = 5$?", etc. This can be followed up with the question "Can you find more than one such x?" and then go on to discussing an *expression* for y in terms of x.

2. **True or false:** Durand-Guerrier [7] discusses the proposition "Any number that ends in 4 is divisible by 4." Some students consider such statements true as well as false since they hold for some numbers and do not hold for some other numbers. I have once come across a student who, when asked whether $x < y$ implies $x^2 < y^2$, answered that it was half-true since it was true for exactly half the numbers. Without getting quite philosophical, it is possible to discuss the notion of a mathematical sentence being true or

false, depending on the restrictions we can place on the values taken by the variables occurring in the sentence.

3. **Truth in a structure:** Is $x^2 - 2 = 0$? Or more precisely, does that equation have a solution? Well, that depends on which number system you are solving the equation in. But this is a source of endless confusion to students.

   The problem is not only that of truth being structure-dependent, but of interpretation of operations and functions being structure-dependent as well. This confusion gets considerably worse when we consider statements such as "Multiplication is the same as repeated addition." This works on positive integers and it is how students learn arithmetic in primary school. But this reasoning is unhelpful when they need to learn $\sqrt{2} \cdot \sqrt{3}$, and so on. This is where the syntax–semantics distinction taught in logic proves to be very helpful.

4. **Equation solving:** A classic problem asks students to evaluate $(x^2-1)/(x-1)$ at $x = 1$. The student answering that the function evaluates to 2 gets shocked when made to realize that the function cannot be evaluated at $x = 1$. The same student, further on, is asked to compute the limit of $(x^2 - 1)/(x - 1)$ as $x \to 1$. Now the limit is indeed 2. The student is left with a bad taste in their mouth: something is surely wrong!

   There are several issues here. Typically the equations, when interpreted as equating functions, require both sides to be defined over the same domain. In the case of limits, the equation is over numbers. The inference system at work in both the cases is not the same, but we do not distinguish them.

5. **Heuristic advice**: One of the side effects of the trouble above is the typical advice given to students to always substitute the answer they obtain in the given equation and verify. This does help in the instance of function evaluation above, since the student can immediately verify that the substitution $x = 1$ leads to division by zero.

   Equation solving comes with a range of heuristics such as grouping terms, moving all terms involving the same variable to one side, changing the sign across the equality symbol, and so on. However, very rarely does a student get any assurance on the reliability of these heuristics, and whether they suffice in all contexts. The answer to this cannot be given without studying the underlying inference systems on equational reasoning.

6. **Functional variation:** The student learns the definition of the *sine* function as the ratio of the 'opposite side' to the hypotenuse of a right-angled triangle. Later on, the student gets to graph the function as an oscillating curve. What

is varying here? Are we saying something about the universe of right-angled triangles? This again is a cause of confusion for many students. It is a much harder question to address, and the limited first order reasoning we discussed was not sufficient to articulate an explanation.

7. **Proving vs. checking:** Typically, proofs in school mathematics involve demonstrations of the kind that some universally quantified statement is true in some implicit structure. The quantification could be over numbers, triangles, circles, and so on. (Note that even the assertion of an infinity of primes is of the form $\forall x \exists y$.) On the other hand, students are used to checking that a property holds for some object: for instance, that 137 is a prime, or that the perpendicular bisectors of a given triangle have a point of concurrency, and so on. Yet the former isn't called a proof whereas the latter is termed a *theorem*. These are clearly logical issues, and they can easily be cleared up with some understanding of logic.

There are many such sources of confusion and incomprehension in school mathematics. We mention these here to point out that teachers could see logic as being helpful towards addressing them.

5 Reasoning About Algorithms

What follows is a different discussion, one that we had with *middle school teachers*, where we did not discuss first order logic, but the use of logic in a more informal sense, emphasizing reasoning rather than logic.

If there is one thing that characterizes school mathematics for a student, it is the learning of algorithms in one context after another; be it long division, factorization of quadratic expressions, matrix inversion, computing compound interest, or finding the standard deviation of given data, and so on. These are largely seen to be disparate, to be learned and applied in context. It is not an exaggeration to say that the teaching and learning of these algorithms dominates school mathematics almost to the exclusion of their declarative content. Moreover, the conceptual vs. procedural debate dominates the discourse on mathematics education.

We mention algorithms to point out that *reasoning* about algorithms is crucial for bridging the conceptual–procedural divide (where it exists), and that this is an indispensable role for logic. Moreover, it is such reasoning that is central to **computational thinking**, a term of great current interest. Indeed it is when algorithms are taught and learned as procedures without any explicit reasoning about them that mathematics becomes difficult for many. The position paper on

teaching of mathematics in the *2005 (Indian) National Curriculum Framework* [16] asserts:

> . . . we have repeatedly referred to offering a multiplicity of approaches, procedures, solutions. We see this as crucial for liberating school mathematics from the tyranny of the one right answer, found by applying the one algorithm taught. When many ways are available, one can compare them, decide which is appropriate when, and in the process gain insight.

When we have a multiplicity of methods, they need to be compared and analyzed to determine which one works best when, and this is the conceptual understanding of importance to mathematics learning. Ideally when students come up with algorithms, compare theirs with other algorithms and argue about them, they acquire confidence in the use of these algorithms. Logic has a great deal to contribute in this regard.

Opportunities for such reasoning are present throughout school. We can already see this with very small children, less than 5 years old: the child is given (say) 20 coins, plays with them, and is then asked to find all the 20 coins and carefully put them in a box. When asked "Have you found all the coins?", they learn to count and make sure. Further, when again asked "How do you know you have counted all the coins?" initially they re-count, but later come up with grouping strategies so that verification is easier. When a child who has distributed 20 toffees among 5 children is asked "How many do you have?" she answers readily. When asked further "How many do each of your friends have?" she needs to reason explicitly by symmetry, assuming that the algorithm she used for toffee distribution treated all children symmetrically. Typically she would have given one toffee to each child in a full round, repeating this for four rounds.

At primary school, students get asked how many different pairs of whole numbers add to (say) 10. When a student offers the solution, there is an implicit question to her: how do you know you have counted them all? This process of verification involves reasoning, and when it becomes a habit, shows the value of formal reasoning as it helps the student to catch routine mistakes. The transition from "do you know" to "how do you know it" is important for mathematics, and reasoning provides the natural vehicle for carrying the student through the transition.

One can list a range of algorithms in number theory, algebra, geometry, trigonometry, calculus, statistics, and business mathematics and ask, in each case, how do students reason about the correctness of the algorithm they learn. Much better, if and when they get to devise algorithms of their own, one can ask how they argue that their method works. While formal proofs of correctness may neither be feasible nor required, argumentation is important, and logic can help in this.

All this, of course, is even more relevant to computing education. Reasoning about procedures is at the heart of computational thinking [17], and the connection to logic is generally missed in school. When a small child is asked to add 2, 104, and 78, and regroups them to first get 80 and then adds it to 104, this re-ordering is a crucial element of computational thinking. Over time a student builds up such rules, and reflection on them and their applicability is a logical exercise, very much necessary for meta-cognition.

Students learning programming need to also learn to build correctness arguments (even if not proofs). Reasoning about programs requires understanding the syntax–semantics distinction, and checking whether a claim is true or not at different points of program execution. Logic plays an indispensable role here [8].

In our interactions, teachers could come up with multiple learning tasks that would encourage children to informally argue the correctness of algorithms used in class. We did not discuss formalisation of these arguments very much.

6 In Conclusion

Experiential accounts of educational interventions are of very limited value. What we need is research based on strong data from classroom practices, analyzed in appropriate theoretical frameworks. Our discussion here should be seen as stressing the need for such research, one that examines whether a logician's perspective may be able to influence mathematics and computing education at the school level positively; and if so, understand how, in a nuanced manner.

As a rule, mathematics educators seem to be largely unaware of the interaction between mathematical logic and school mathematics, and what logic might mean at different stages of schooling. While there is extensive literature on how students reason in mathematics classes, this is not examined in the light of logic in itself.

The aforementioned position paper on teaching of mathematics in the *2005 (Indian) National Curriculum Framework* [16] talks about *compartmentalisation* as one of the major problems of the mathematics education milieu, with little interaction between mathematics teachers at university and those in high school; or between the latter and teachers in the elementary school. Conversations between high school teachers and university teachers involved in formal methods in computing can perhaps be of benefit to both.

7 Acknowledgements

The author thanks the school students and mathematics teachers who have interacted with him extensively and taught him with great patience about school

education. Furthermore, the author thanks the Tamil Nadu Science Forum for always making room for such interactions, and Dennis Komm and Thomas Zeume for their encouragement to write this article and for immensely helpful feedback on the first draft.

References

[1] M. Aristidou: Is mathematical logic really necessary in teaching mathematical proofs? *Athens Journal of Education*, 7(1):99–122, 2020. DOI: https://doi.org/10.30958/aje.7-1-5

[2] J. Baldwin: Logic across the high school curriculum. https://homepages.math.uic.edu/~jbaldwin/pub/logicnov14.pdf. Last accessed 8 Aug 2020.

[3] R. Banerjee and K. Subramaniam: Evolution of a teaching approach for beginning algebra. *Educational Studies in Mathematics*, 81(2):351–367, 2012. DOI: https://doi.org/10.1007/s10649-011-9353-y

[4] R. Charles: Big ideas and understandings as the foundation for elementary and middle school mathematics. *Journal of Mathematics Education Leadership*, 7(3), 2005. https://www.cambridgemaths.org/Images/266726-big-ideas.pdf. Last accessed 12 Jan 2023.

[5] F. Chellougui, V. Durand-Guerrier, and A. Meyer: Discrete mathematics, computer science, logic, proof, and their relationships. In V. Durand-Guerrier, R. Hochmuth, E. Nardi, and C. Winslow: *Research and Development in University Mathematics Education*, Routledge, 2021. DOI: https://doi.org/10.4324/9780429346859

[6] V. Durand-Guerrier, P. Boero, N. Douek, S. S. Epp, and D. Tanguay: Examining the role of logic in teaching proof. In G. Hanna and M. de Villiers (eds): *Proof and Proving in Mathematics Education.* New ICMI Study Series, vol 15. Springer, Dordrecht 2012. DOI: https://doi.org/10.1007/978-94-007-2129-6_16

[7] V. Durand-Guerrier: The place of logic in school curricula and advocacy for logic at school. Presented at the Panel on Logic Education. In *9th Indian Conference on Logic and Applications (ICLA 2021)*, available from https://www.isichennai.res.in/~sujata/icla2021/panelslides/durand-guerrier.pdf Last accessed 30 Sep 2025.

[8] F. Chellougui, V. Durand-Guerrier, and A. Meyer: Discrete mathematics, computer science, logic, proof, and their relationships. In V. Durand-Guerrier, R. Hochmuth, E. Nardi, and D. Winslow (eds): *Research and Development in University Mathematics Education*, Routledge, 2021. DOI: https://doi.org/10.4324/9780429346859

[9] S. S. Epp: The role of logic in teaching proof. *The American Mathematical Monthly*, 110(10):886–899, 2003. DOI: https://doi.org/10.1080/00029890.2003.11920029

[10] S. S. Epp: The language of quantification in mathematics instruction. In L. V. Stiff and F. V. Curcio (eds.): *Developing Mathematical Reasoning in Grades K–12*. Chapter 12, Reston, VA: NCTM, 1999. https://condor.depaul.edu/sepp/NCTM99Yrbk-ss.pdf Last accessed 30 Sep 2025.

[11] G. Hanna: Rigorous proof in mathematics education. OISE Press, Toronto, 1983. DOI: https://doi.org/10.1023/A:1012737223465

[12] C. Hurst: Big challenges and big opportunities: The power of 'big ideas' to change curriculum and the culture of teacher planning. In *Proceedings of the 37th annual conference of the Mathematics Education Research Group of Australasia*, pp. 287–294. Sydney, 2014. https://files.eric.ed.gov/fulltext/ED572609.pdf Last accessed 30 Sep 2025.

[13] S. Jayasree, K. Subramaniam, and R. Ramanujam: Coherent formalisability as acceptability criterion for students' mathematical discourse. *Journal of Research in Mathematics Education*, 2022. DOI: https://doi.org/10.1080/14794802.2022.2041469

[14] National Council for Educational Research and Training: *Mathematics (Classes XI–XII)*, https://ncert.nic.in/pdf/syllabus/Preliams2.pdf Last accessed 30 Sep 2025.

[15] National Council for Educational Research and Training: *Mathematics Textbook for Class XI*, https://ncert.nic.in/textbook.php?kemh1=0-14 Last accessed 30 Sep 2025.

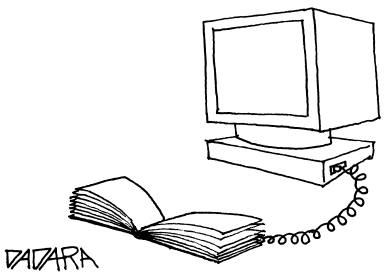
[16] Position paper of the National Focus Group on Teaching of Mathematics, NCERT 2006, https://ncert.nic.in/pdf/focus-group/math.pdf Last accessed 30 Sep 2025.

[17] R. Ramanujam. Computational thinking: the new buzz. *At Right Angles*, 13:23–31, 2022. https://publications.azimpremjiuniversity.edu.in/3747/ Last accessed 30 Sep 2025.

[18] R. Ramanujam: Big ideas from logic for mathematics and computing education. In *Proceedings of the 10th Indian Conference on Logic and its applications (ICLA 2023)* Springer FoLLI series in LNAI Vol 13963, pp. 79–91, 2023. DOI: https://doi.org/10.1007/978-3-031-26689-8_6

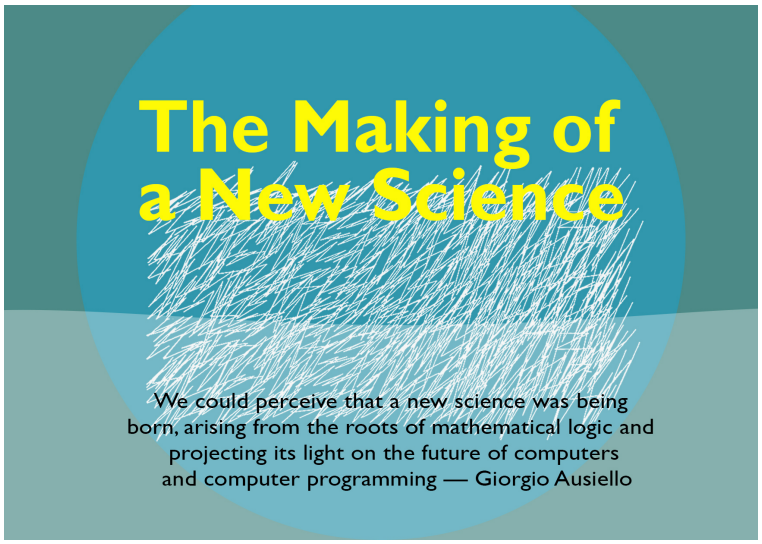
[19] G.-C. Rota: The phenomenology of mathematical proof. *Synthese*, 111:183–196, 1997. DOI: https://doi.org/10.1023/a:1004974521326

News and Conference Reports



The Making of a New Science

Andrew Goodall and Jaroslav Nešetřil



An exhibition, entitled 'The Making of a New Science,' on the early history of theoretical computer science is being held from June to October 2025 at Galleria Chodba at Malostranské náměstí 2/25 in Prague, curated by the authors of this article. The origins, motivation and sources of this exhibition are explained in a short text prefacing the exhibition catalogue (to be published by the Karolinum Press) which, with small modifications, is reprinted here. The exhibition opened on 18 June and will run until October 2025, a few weeks after the start of the winter semester at Charles University, thereby giving an opportunity for incoming and returning students at the School of Computer Science of the Faculty of Mathematics and Physics to see the exhibition. The opening of the exhibition was coordinated with a major event hosted by Charles University in Prague from 23 to 27 June this year, the 57th ACM Symposium on Theory of Computing (STOC 2025). Galleria Chodba is located in the very centre of Prague and our exhibition has been attracting the attention not only of students and scientists but of the general public (among them the many tourists visiting Prague at this time of year).

Preface to the exhibition

It is very rare that one can make an exhibition tracing the entire history of an established scientific discipline. Theoretical Computer Science (TCS for short) – which only emerged in the 1970s as a recognized field per se out of its origins in logic, mathematics and engineering – has developed and matured so quickly that this is possible; or, better, we can at least try to present the history of the field in a concise and yet uncrowded way.

TCS (or the Theory of Computing, Computability Theory, or even Algorithms and Complexity, to give just a few of the several names that are used to refer to the discipline) develops the abstract concepts and mathematical models that underpin computation, with a particular focus on the design, analysis, and efficiency of algorithms. It has developed in the last few decades from humble origins into a field which is taught at most universities worldwide, and which spans a large part of technological development today. One can say even more: perhaps nowhere in contemporary science is there such a direct line from theory to development and then to praxis and its consequent impact on everyday life. Our aim here can only be to sketch parts of this evolution of TCS. We concentrate on the early origins in the 1920s to c. 1990 (with occasional glimpses beyond); we thereby also limit our scope by concentrating on the theory that established itself in the

twentieth century, omitting some trends which have proved to be very active more recently. This is more concretely described below. We are fortunate that we can build our exhibition around two earlier instances. One is the excellent book The Making of a New Science (Springer 2023) by Giorgio Ausiello. This book, by one of the pioneering leaders and organizers of the development of TCS in Europe, is both a factual report of the history of TCS until 1980 and a memoir replete with personal details, making it fascinating reading for non-specialists and specialists alike. Prof. Ausiello put us in contact with the people behind our second source, the exhibition '50 Years of Theoretical Computer Science' shown at the ICALP '22 conference in Paris. We thank Sandrine Cadet and Sylvain Schmitz, organizers of this exhibition, for allowing us to use and modify their material.

Our exhibition 'The Making of a New Science' consists of thirteen large panels accompanied by an explanatory text. The following outline may give an idea of the exhibition for those unable to make it to Prague to see it.

The first panel (bearing the title of the exhibition, **The Making of a New Science**) pays tribute to Prof. Giorgio Ausiello, featuring a brief quotation from his book along with quotes from other recognized computer scientists, and two of Vera Molnár's digital plotter drawings (from the series *Letters from my mother*).

The second panel (**On the shoulders of giants**) picks out some of the major milestones and key figures in the theory of computation. This of course goes back to star mathematicians (and scientists) of the early twentieth century (we omitted traces of the field to be found in previous centuries). This earlier history is recorded in the reels of black-and-white "film" while more recent names are recorded in the array of colour "polaroids". The panel is complemented by a list of awardees of the Nevanlinna–Abacus medal of the International Mathematical Union (established in 1982 and given at the International Congress of Mathematicians ever since).

The next panel **Pioneers in TCS** continues the focus on key figures of the field, featuring a spotlight on Maurice Nivat (transferred directly from the Paris exhibition 50 Years of Theoretical Computer Science), whose individual story has counterparts for each and every one of the notable figures in TCS listed in the remainder of the panel. These lists give in chronological order all the winners of four major annual international prizes: the A. M. Turing Award, Gödel Prize, EATCS Award and Donald E. Knuth Prize. Seeing these lists, one gets a better feeling of how the development of TCS has been a product of collective endeavour by a remarkable and numerous set of individuals.

Brussels, 1972 reworks a panel from the Paris exhibition '50 Years of Theoretical Computer Science,' supplementing it by facsimiles of key documents reproduced in the appendix to Ausiello's *The Making a New Science*. Described are the key events leading to the creation of the European Association of Theoretical Computer Science (EATCS), highlighting the role played by Maurice Nivat and

others in bringing it to fruition.

The foundation of new institutions led to the creation of a new type of conference, one with refereed contributions selected by a programme committee. Such conferences became the dominant medium for scientific communication in TCS, and a (non-exhaustive) selection of the most long-running, active and prestigious are displayed in the panel **Early conferences in TCS**. The penultimate panel of the exhibition (see below) features an editorial by Moshe Vardi discussing the role of conferences in computer science, which is very different from mathematics.

The next five panels are devoted to particular areas of TCS in their early development. The choice of topics is of course rather arbitrary as current TCS is a very broad field. But on the history there is a consensus. We included material from the Paris panels on computational complexity, logic and complexity, automata and a schematic overview of algorithms that have shaped the world; on the other hand, we left out the Paris material on zero-knowledge proofs, fine-grained complexity, model checking, the science of programming, machine-checked proofs, and quantum computing. We are limited to thirteen panels and some of these topics will be treated in a follow-up exhibition.

With these constraints, our panels are **Automata theory**, describing the development of this area from Alan Turing on; **Computational complexity (I)**, describing the birth of complexity classes and "complexity" as we know it today; **Computational complexity (II)**, describing the influence of logic and the logical side of TCS; and **Algorithms (I)**, representing some of the plethora of beautiful algorithms from which the theory of algorithms has evolved. These four panels are adapted from the Paris panels, to which we have added photographs of some of the key figures involved. Additionally, in the Algorithms (I) panel we highlighted seven major algorithms from those "shaping the world", while, in the Automata theory panel we included diagrams illustrating the evolution of a pair of elementary cellular automata (rules 30 and 110).

The next two panels are the only panels devoted to specific problems. **Algorithms (II)** treats a topic dear to all Czech mathematicians and computer scientists as it describes the role of Borůvka and Jarník in the development of the minimum spanning tree algorithm (as well as the Steiner tree algorithm). The panel **Advanced graph theory in TCS** describes the story of expanders – another TCS saga. Expanders represent a key structure in the theory of algorithms and their construction involves beautiful and difficult mathematics.

The last two panels document some of the bewilderingly extensive activity in TCS. The panel **Publishing research** includes material from the Paris exhibition panel focusing on ICALP publications, to which we have added a discussion about the distinctive practice of theoretical computer of using conferences and their proceedings as the primary mode of communication rather than journals: we reproduce an editorial by Moshe Vardi in Communications of the ACM, concern-

ing a still very current topic of debate. The final panel **Selected textbooks in TCS, 1966-1999** gives a panorama of various twentieth century textbooks, both basic and advanced, many appearing in new editions to the present day, and all of them influential in directing the course of TCS as it entered the twenty-first century.

Remarks or questions related to this exhibition are welcome and can be sent to the gallery email address, galleriachodba@iuuk.mff.cuni.cz.

It remains for us to thank the School of Computer Science of the Faculty of Mathematics and Physics for supporting Galleria Chodba as well as the UNCE project 'Language, image, gesture: forms of discursivity' (a joint project with the Faculty of Arts, Charles University).

A Scientific Story (1985-2025) from WDAG to DISC

Michel Raynal[⋆] & Nicola Santoro[◇]

[⋆] Univ Rennes (IRISA, Inria, CNRS) France
[◇] Carleton University, Ottawa, Canada

1 What is Distributed Computing?

Distributed computing is the science of cooperation. More precisely a distributed algorithm is composed of a set of computing entities (imposed to the programmers) each providing its own input, that have to cooperate to a common goal depending on the set of inputs (usually in the presence of adversaries such as asynchrony and failures) [2, 3].

. If follows that, in distributed computing, possibility/impossibility results and algorithmic techniques differ from what is encountered in sequential or parallel computing. This characterizes the distributed computing community. DISC and PODC are the very top conferences devoted to distributed computing.

2 Distributed Computing *vs* Parallel Computing

Distributed computing is sometimes confused with a form of parallel computing. Rougly speaking parallelism is an extension of sequential computing the aim of which aim is to obtain efficient programs. Let us notice that any parallel algorithm could be executed sequentially on a mono-processor(of course in a very inefficient way). This is not the case in distributed computing. Both distributed computing and parallel computing involve concurrency.

3 At the Very Beginning: When DISC was WDAG

3.1 The Start

What is now named DISC began in 1985, a sibling to the 4-years old PODC that, at the time, was the official (and, until then, the only) outlet for the theoretical aspects of the field. Like PODC (and later SIROCCO), the new conference started in Ottawa; created by Nicola Santoro with the collaboration of Eli Gafni, its focus was on the "algorithmic" aspects of distributed computing, and its name at birth was *International Workshop on Distributed Algorithms on Graphs* (hence the acronym *WDAG*[1]).

The creation of WDAG was motivated by a strong sense of dissatisfaction felt in regards with what was perceived as an unbalanced official "coverage" of the distributed algorithms research efforts and outputs, and a restricted view of what was important in distributed computing. In this view, the most important concern in the field appeared to be not on general problem solving by the distributed computational entities composing the system but rather on coping with their failures. In other words, unlike practically all the germane computational fields (e.g., sequential computing, parallel computing, computational geometry, etc) the main concern from the start was on fault-tolerance, and the core problems investigated were obviously *consensus* and its many reformulations. Furthermore, the model used in the research almost exclusively assumed the system to be *fully synchronous* and the communication topology of the entities to be the *complete graph*. Most of the algorithmic focus was on the solution of these problems. This in spite of the fact that the majority of problems investigated at the time (especially in the European community and by researchers from combinatorics and discrete mathematics) were about networks and graphs (e.g., spanning-tree construction, routing, etc.), information diffusion (e.g., broadcast, converge-cast, gossip, etc.), control (e.g., symmetry breaking, election, mutual exclusion, etc.), where the concern was to determine the minimal system conditions necessary first of all for solving the problem (solving it in presence of faults was a subsequent concern). Interestingly, most of these investigations were concerned with the more realistic assumptions of *asynchronous* systems and (often unknown) *different graph topologies*.

It was based on this perceived divide that the decision to start WDAG was made, and to make the difference more explicit, the words "Algorithms" and "Graphs" were added to the name. So a new outlet for the DC community with a slightly different view was born (see Photo 1), and its (hard cover!) proceedings, edited by Eli Gafni (EG) and Nicola Santoro (NS), were published in 1986 by

[1] A typical mistake has been to assume that the "G" in the acronym was part of the word "algorithm".

Carleton University Press.

Figure 1: Participants to the first WDAG. Among them: Jan van Leeuwen, Christian Lavault, Jan Pachl, Joe Peters, Shmuel Zaks, Yehuda Afek, and ... PhD student Shay Kutten.

3.2 The WDAG Years: 1985-1998

Following the very positive response to the first WDAG, the decision to continue (initially on a biennial basis) was the natural result. The "de facto" steering committee (including now Jan van Leeuwen) decided the location of the workshop to be moved to Europe and, in order to get a wide dissemination, the proceedings to be published by Springer LNCS. So, since its second edition (1987), WDAG became an European event, eventually under the official sponsorship of EATCS (the European Association for Theoretical Computer Science, see below).

Interestingly, with each edition, new members were permanently co-opted into the de-facto steering committee which became a large collaborative collective (a mini distributed system); in particular, starting with Michel Raynal (WDAG 3), we had Sam Toueg and Paul Spirakis (WDAG 5), Shmuel Zaks (WDAG 6), and Paul Vitanyi (WDAG 8) all join the SC collective.

In all these years, the differences between WDAG and PODC were several, from their organizative structure and the choice of geographical locations (Europe for WDAG and Canada at that time for PODC), to the emphasis on content (algorithmic for WDAG and general principles for PODC); a fundamental distinction remained for quite a long time their basic "nature": one was a *workshop*, while the other a *symposium*. The latter difference went beyond the words used; it was

Figure 2: The authors of this paper at WDAG 3.

rather felt and experienced by the participants. Slowly over the years, the differences started to disappear.

3.3 From WDAG to DISC: 1998

Eventually, according to the papers submitted (and accepted) and the scientific interests of the people attending the workshop, it appeared that the meaning of the words *Workshop* and *Graphs* was too restrictive and no longer conveyed the spirit of this scientific meeting, which had become much broader.

So, in 1998, the PC chair Shay Kutten, proposed to replace the name *Workshop*

(a) WDAG 4: NS, Shmuel Zaks, Sergio Rajsbaum, Shlomo Moran.

(b) WDAG 9: Hagit Attiya, Shmuel Zaks, NS.

on Distributed Algorithms on Graphs by *International Symposium on Distributed Computing*, in short DISC. The Business meeting fully agreed on this proposal, and WDAG 12 became DISC 12.

Figure 4: WDAG 12 = DISC 12.

The reader will find the full list of WDAG workshops and DISC symposia (with years, places and programs committee chairs) at "dblp DISC".

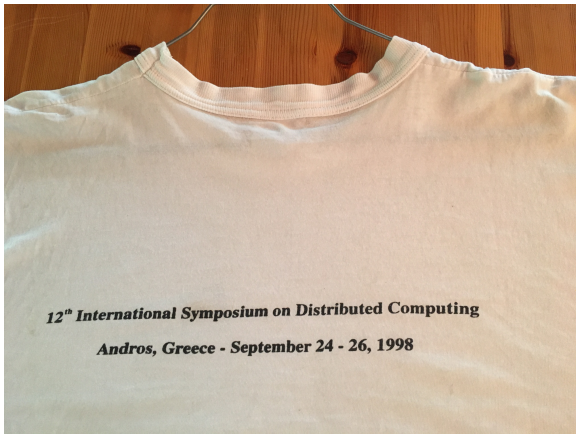


Figure 5: A historical piece: the very first DISC T-shirt

4 DISC 20th Anniversary

This venue of DISC was Stockholm. To celebrate this 20th anniversary of the DISC Symposia three famous speakers were invited to give a talk, namely Leslie Lamport, Nancy Lynch, and Michael Rabin.

5 Sharing Dijkstra Award with PODC

The Edsger W. Dijkstra Prize in Distributed Computing is named for Edsger Wybe Dijkstra (1930-2002), a pioneer in the area of distributed computing. His foundational work on concurrency, semaphores, mutual exclusion, deadlock, finding shortest paths in graphs, fault-tolerance, self-stabilization, among many other contributions comprises one of the most important supports upon which the field of distributed computing is built. No other individual has had a larger influence on research in principles of distributed computing. The prize is given for outstanding papers on the principles of distributed computing, whose significance and impact on the theory and/or practice of distributed computing has been evident for at least a decade.

The Prize was initially sponsored by the ACM Symposium on Principles of Distributed Computing (PODC). In 2003 the chair and the vice-chair (Alex Shvartsman) of the steering committee of DISC initiated a procedure for the price to be awarded by both PODC and DISC. The current chair of the PODC steering committee (Mark Tutle) agreed, but as PODC depends on ACM, the head of ACM required DISC to be part of an official organisation. After a short discussion the head of EATCS agreed to support DISC and since then the award is presented annually, with the presentation taking place alternately at ACM PODC and EATCS DISC.

6 From Springer LNCS to LZI LIPIcs

An important change appeared in 2017 (31st DISC) where the steering committee decided to have the proceedings produced by LIPIcs (Leibniz International Proceedings in Informatics), that is a series of high-quality conference proceedings across all fields in informatics established in cooperation with Schloss Dagstuhl - Leibniz Center for Informatics. Among other points, this favoured free open access to all the DISC articles.

7 Which Future?

Prediction is very difficult, especially about the future.
Niels Bohr (1885–1962).

8 The Guardians of the DISC Temple

(Past Chairs of the DISC Steering Committee)

Sam Toueg, Cornell, USA, 1996-1998
Shmuel Zaks, Technion, Israel, 1998-2000
André Schiper, EPFL, Switzerland, 2000-2002
Michel Raynal, Irisa, France, 2002-2004
Alex Shvartsman, University of Connecticut, 2004-2007
Rachid Guerraoui, EPFL, Switzerland, 2007-20098
Nicola Santoro, Carleton University, Canada, 2009-2011
Sergio Rajsbaum, UNAM, Mexico, 2011-20138
Antonio Fernandez Anta , IMDEA Netw., Spain, 2013-2015
Shlomi Dolev, Ben Gurion University, Israel, 2015-2017
Roberto Baldoni, Sapienza Universita di Roma, Italy, 2017-2018
Yoram Moses, Technion, Israel, 2018-2020
Andréa Richa, Arizona State University, USA, 2020-2022
Jukka Suomela, Aalto University, Finland, 2022-2024
Hagit Attiya, Technion, Israel 2024-2026.

9 What is a Good Conference?

Of course this is an opinion and not a definition. A good conference is located in a nice place, and is the product of good submissions and good referees. It is also a place where people discuss on their reseach topics (or any other topic!) and where we can meet scientific people among which some of them will become good friends[2] with which we have discussions that can last several years, improving our view of the topics we are working on, etc. In some sense, a good conference is like a (scientific) family meeting composed of researchers interested in common topics.

[2]This notion of frienship has nothing to do the notion of "friend" as imposed by Facebook.

10 What is a Good Paper?

This is an important question. The answer to this question given below is from [2]. At the very beginning (when I (MR) was younger, i.e. in the previous millennium!) my answer was mainly based on an objective numerical criterion, namely, a good paper is a paper with numerous citations". Later, I was saying "a paper that won the best paper award in a top conference". Still later I was saying "a paper that won a prize devoted to more than ten years old papers", etc. But over time, none of these integer-based definitions fully satisfied me, and I started thinking about the papers that I myself consider as very important papers ... and I discovered that those were papers I was a little bit kindly jealous ... not to be a co-author! This was because those are papers I like to read (and reread) because they are nicely written, their ideas go beyond their technical content, they introduce new ideas in a simple and efficient way, and have a very strong impact on the community. This is the effect of good papers: everyone makes them "theirs", assimilating them and passing their essence to students.

Finally, to conclude, considering that the pair research/teaching defines the golden coin that lies at the heart of Universities all over the world, and as research feeds teaching (and vice-versa) it is important to consider that "Teaching is not an accumulation of facts" as stated by L. Lamport in [1].

References

[1] Lamport L., Teaching concurrency. *ACM Sigact NEWS*, 40(1):58-62 (2009)

[2] Raynal M., About informatics, distributed Computing, and our Job: a personal view. *Proc. 30th Int'l Colloquium on Structural Information and Communication Complexity (SIROCCO'23)*, Springer LNCS 13892, pp. 44-56 (2023)

[3] Raynal M., Mutual exclusion vs consensus: both sides of the same coin? *Bulletin of the European Association of Theoretical Computer Science (EATCS)*, Vol.140, 14 pages (2023)

Report on BCTCS 2025

The 41st British Colloquium for Theoretical Computer Science

14–16 April 2025, University of Strathclyde

Alasdair Lambert, Fredrik Nordvall Forsberg, and Sean Watters

The British Colloquium for Theoretical Computer Science (BCTCS) is an annual forum for researchers in theoretical computer science. A central aspect of BCTCS is the training of PhD students, providing an environment for students to gain experience in presenting their work, to broaden their outlook on the subject, and to benefit from contact with established researchers. The scope of the colloquium includes all aspects of theoretical computer science, including automata theory, algorithms, complexity theory, education, semantics, formal methods, concurrency, types, languages and logics.

BCTCS 2025 was hosted by the University of Strathclyde and held from 14th to 16th April 2025. The event featured an interesting and wide-ranging programme. A total of 31 contributed talks — predominantly by PhD students — were presented at the meeting alongside six keynote talks. Wednesday afternoon was focused on theoretical computer science education. We are grateful to The Scottish Informatics and Computer Science Alliance for sponsoring the meeting, and to the Heilbronn Institute for Mathematical Research provided for support in the form of travel bursaries to a number of PhD students and early career researchers.

BCTCS 2026 will be hosted by the University of Birmingham from 30th March to 1st April 2026. Researchers and PhD students wishing to contribute talks concerning any aspect of Theoretical Computer Science are cordially invited to do so. Further details are available from the BCTCS website at www.bctcs.ac.uk.

Invited Talks

Jess Enright (University of Glasgow)
How temporal locality can help us solve problems on temporal graphs

A temporal graph is a graph whose edges are active at only some (specified, known) times. This ability to capture changes over time provides a useful model of real dynamic networks, but renders many problems studied on temporal graphs more computationally complex than their static counterparts.

To contend with this, there has been recent work devising parameters with respect to which temporal problems become tractable. One such parameter is vertex-interval-membership width. Broadly, this gives a bound on the number

of vertices we need to keep track of at any time. We will outline how one uses this sort of parameter algorithmically, define a generalisation of this parameter, and give a meta-algorithm for both parameters. This simplifies proving that a temporal problem is tractable for either parameter.

This is joint work with Laura Larios-Jones, Sam Hand, and Kitty Meeks.

Rob van Glabeek (University of Edinburgh)
Ensuring Liveness Properties of Distributed Systems with Justness

Often fairness assumptions need to be made in order to establish liveness properties of distributed systems, but in many situations they lead to false conclusions. Here I present ongoing work aiming at laying the foundations of a theory of concurrency that is equipped to ensure liveness properties of distributed systems without making fairness assumptions. Instead I advocate assuming justness, a strong progress property that is essentially weaker and more realistic than fairness. When assuming justness, contemporary process algebras and temporal logics fail to make distinctions between systems of which one has a crucial liveness property and the other does not. This necessitates an overhaul of these frameworks.

Nicolai Kraus (University of Nottingham)
Dependent type theory: from propositions and sets to spaces

Type theories (in the sense of Martin-Löf and Coquand) are programming languages with type systems that are expressive enough to formulate mathematical statements. Several proof assistants, such as Agda, Lean, and Coq/Rocq, make use of this version of the Curry-Howard correspondence.

There are various interpretations of type theories, assigning different meanings to types. Traditionally, types were interpreted as propositions or sets, but since the advent of homotopy type theory, they are also seen as spaces. As a consequence, the correspondence between mathematical statements and types is not unique. For example, a group is usually represented as a carrier set with a unit element and an operation satisfying associativity, identity, and inverse laws. However, in homotopy type theory, it can alternatively be defined as a so-called "pointed connected 1-type".

In this talk, I will explain the ideas and insights that eventually led to homotopy type theory, along with an approach to computer formalizations that it enabled.

Conor Mc Bride (University of Strathclyde)
Talking Out Of School

Since 2008, I have been teaching first year undergraduates in semester one: fresh out of school! I have always taught more mathematical topics, which is to say that whatever topics I have been tasked with teaching I have made more mathematical.

Arriving from school, many expect to be taught to the test and few have much left of their sense of adventure. In the interests of prompting discussion, I'll share some tales from the trenches of trouble, tactics, and technology. Pilot light ignition is a hard problem. I have not solved it, but I have had moments. Misery loves company, and perhaps together we can manage more optimism than we can muster in isolation.

Jakub Opršal (University of Birmingham)
Homotopy and complexity of graph colouring

I will talk about an emerging new application of homotopy theory in computational complexity of discrete problems. This approach is build on the question: *How does the topology of the solution space of a computational problem influence its complexity?*

The idea emerged from investigations of the complexity of variants of graph colouring, including approximate graph colouring. A colouring of a graph with k colours is an assignment of colours to vertices under which no edge is monochromatic. Graph 3-colouring is a prototypical example of an NP-complete problem listed by Karp in his seminal paper. There are many variations on this problem whose complexity remains widely open. For example, although it is generally believed that colouring a 3-colourable graphs with a fixed number of colours is NP-hard, only hardness of colouring of such a graph with 5-colours is known (and shown only in 2019).

The talk will focus on several related results about variations of graph colouring that share a common theme of using a topological method based on tools from topological combinatorics and on ideas of Lovász [J. Comb. Theory, Ser. A, 25(3):319-324, 1978]. These ideas formalise the notion of a solution space, and hence provide a rigorous framework for the study of the above fundamental question

Elizabeth Polgreen (University of Edinburgh)
Making the most of large language models for program synthesis (when you want correct answers)

Since the release of ChatGPT, large language models (LLMs) have been dominating the discourse around code generation. In contrast, the best-performing solvers in the world of formal program synthesis are still based around enumerative algorithms. So, are we behind the times? Are LLMs the answer to all our synthesis problems? In this talk, I will present two approaches to making the most of LLMs in formal domains. First, I will present a technique based on combining enumerative program synthesis with guidance from an LLM [1]. Focusing on the Syntax-Guided Synthesis (SyGuS) competition benchmarks, we found that, whilst LLMs can produce syntactically correct SyGuS expressions, they fail to produce

semantically correct solutions for more than half of the benchmarks. We propose a novel enumerative synthesis algorithm, which integrates calls to an LLM into a weighted probabilistic search, allowing 2-way exchange of information between the two components, and increasing the number of benchmarks solved to 80% (and outperforming the state-of-the-art solver). Second, I will discuss our work using LLMs to generate models of systems for verification [2]. Here, we enable LLMs to generate syntactically correct code, even in programming languages that barely feature in the training data, by combining an HCI technique called natural program elicitation and a Max-SMT solver. We apply our approach to generating models of systems in the UCLID5 verification language, improving syntactic correctness from $< 10\%$ to $> 80\%$ on a set of textbook problems. Whilst I don't believe LLMs are the answer to all our synthesis problems, I think our results demonstrate that perhaps techniques used in formal synthesis and programming languages could be the answer to many of the problems facing LLMs...!

[1] Guiding Enumerative Program Synthesis with Large Language Models. Yixuan Li, Julian Parsert, and Elizabeth Polgreen.

[2] Synthetic Programming Elicitation and Repair for Text-to-Code in Very Low-Resource Programming Languages. Federico Mora, Justin Wong, Haley Lepe, Sahil Bhatia, Karim Elmaaroufi, George Varghese, Joseph E. Gonzalez, Elizabeth Polgreen, and Sanjit A. Seshia.

Contributed Talks

Quantale Enriched Semantics for Graph Mathematical Morphology
Ignacio Bellas Acosta (University of Leeds)

L-fuzzy relations, functions from a set $X \times X$ to a lattice L, have been used in mathematical morphology to capture the behaviour of structuring elements of grey-scale and colour-based images. However, L-fuzzy relations fail to capture the graphical behaviour found in graph-based mathematical morphology. In this talk we address this issue, developing a relational framework within quantale enriched category theory. In this framework, bi-modules (also referred to as profunctors in the literature) generalise L-fuzzy relations, addressing the graph nature of the problem. However, this generalisation comes with limitations on the definition of the converse and complement operations. Assuming the underlying quantale is Girard, we show the existence of a local adjunction of operations that play the role of the converse. Furthermore, we show the linear negation induces two pairs of adjoint operations that serve as generalisations of the complement operation. This talk is based on joint work with John G. Stell.

Malin Altenmüller (University of Edinburgh)
A data type of intrinsically plane graphs

Plane graphs are those that can be drawn on the sphere without any edges crossing. They are subject of interest not only in graph theory but also as a combinatorial representation of string diagrams for certain monoidal theories that are sensitive to their topology (for example non-symmetric monoidal categories). I will present an inductive data type of plane graphs which uses a graph's spanning tree as a skeleton and stores the remaining edges alongside it. The order in which we store these additional edges is crucial as it enforces the planarity property of the entire graph. We encode the planarity information as part of a graph's type, thus all graphs of the type are intrinsically plane. Additionally, operations on plane graphs such as substitution preserve the planarity property by definition.

Jungho Ahn (Durham University)
A coarse Erdős-Pósa theorem

An induced packing of cycles in a graph G is a set of vertex-disjoint cycles such that G has no edge between distinct cycles of the set. The classic Erdős-Pósa theorem asserts that for every positive integer k, every graph contains k vertex-disjoint cycles or a set of $O(k \log k)$ vertices which intersects every cycle of G. We generalise this classic Erdős-Pósa theorem to induced packings of cycles. We show that there exists a function $f(k) = O(k \log k)$ such that for every positive integer k, every graph G contains an induced packing of k cycles or a set X of at most $f(k)$ vertices such that the closed neighbourhood of X intersects every cycle in G. Our proofs are constructive and yield a polynomial algorithm.

This is based on a joint work with Pascal Gollin, Tony Huynh, and O-joung Kwon.

Pete Austin (University of Liverpool)
Temporal Explorability Games

Temporal graphs extend ordinary graphs with discrete time that affects the availability of edges. We consider solving games played on temporal graphs where one player aims to explore the graph, i.e., visit all vertices. The complexity depends majorly on two factors: the presence of an adversary and how edge availability is specified. We demonstrate that on static graphs, where edges are always available, solving explorability games is just as hard as solving reachability games. In contrast, on temporal graphs, the complexity of explorability coincides with generalized reachability (NP-complete for one-player and PSPACE- complete for two player games). We further show that if temporal graphs are given symbolically, even one-player reachability and thus explorability and generalized reachability games are PSPACE-hard. For one player, all these are also solvable in PSPACE and for two players, they are in PSPACE, EXP and EXP, respectively.

Jakub Bachurski (University of Cambridge)

Breaking records: structural subtyping as a language design principle

Subtyping is a classical programming language feature, usually associated with nominal subtyping in object-oriented programming. Structural subtyping — where the subtyping relation is derived from the shared structure of types — is an alternative. Structural typing is present in many forms in modern languages, but it is often incomplete without subtyping. Regardless, subtyping often ends up omitted because of its inherent complexity.

I argue that with the introduction of algebraic subtyping — a new framework for ML-style type inference — the time has come to revisit structural subtyping. In my thesis I propose a novel language design, Fabric, and demonstrate the relevance of structural subtyping to modern design problems. I propose a new approach to typing array programs, show how to capture record extension within algebraic subtyping without row polymorphism, and compile Fabric into WebAssembly with a focus on efficient core data structures. I aim to show structural subtyping narrows the expressive gap between static and dynamic languages, it helps us statically type check dynamic programs, and its ideas can improve how we program generally.

Justus Becker (University of Birmingham)

Proof translations for structurally different sequent calculi of intuitionistic modal logic

The proof theory of intuitionistic and modal logics has been extensively studied in recent decades; the advent of labelled and nested systems allowed to come up with very expressive and versatile cut-free complete systems. A good way to compare different calculi for the same logic is by establishing effective translations between derivations of these. Effective translations are functions between derivation trees that are definable via an algorithm, allowing for a more constructive completeness proof and a way to extract new systems out of this translation.

It is known that nested and tree-labelled sequents have the same expressive power, meaning that a calculus employing one can also be expressed in terms of the other formalism. This, for example, enables one to distinguish two calculi written in different formalisms.

The propositional modal logic IK has been developed in the 1980s and 90s as the intuitionistic variant of the modal logic K. Meanwhile, a lot of different proof systems have been developed for this logic.

Here, I will compare the fully labelled calculus labIK, that internalises the birelational semantics for intuitionistic modal logic, with the Maehara-style nested calculus NIKm. The sequents of these calculi are structurally inequivalent, which will lead to a translation that is not preserving the structure of sequents. It requires one, for instance, to add some admissible rules explicitly or to edit derivation trees

before translating them. This is also due to labIK being fully invertible, unlike the simple nested sequent calculus.

This result sheds some light on how structurally different calculi behave, especially comparing a fully invertible calculus and a system with non-invertible rules. We can see, for example, how retrieving lost information in backtracking translates to having all information available in the full calculus. This result can also be applied to many other intuitionistic modal logics.

Géraldine Brieven (University of Liège)
From Visualization to Coding: Practicing Graphical Loop Invariants in CAFÉ 2.0

This talk focuses on a programming approach relying on an informal and graphical version of the Loop Invariant to write the code. This approach is taught in the context of a CS1 course where students are exposed to various C programming language concepts and algorithmic aspects. The key point is to imagine a problem-solving strategy (the Graphical Loop Invariant) prior to coding, which then becomes reasonably straightforward when based on this graphical representation.

This talk presents the rules for building a sound and accurate Graphical Loop Invariant as well as the programming methodology. As such, our programming approach might be seen as a first step towards considering formal methods in programming courses without making any assumption on students mathematical background, as it avoids mathematical notation entirely. We also discuss CAFÉ 2.0, a learning tool we developed to support teaching and practicing the Graphical Loop Invariant concept. Finally, we provide insights into how students adopt this approach and utilize it in CAFÉ 2.0 to solve problems.

Paola Bruscoli (University of Bath)
Linking Proof Theory to Game Design

In the past few years I have started proposing individual projects to UG/PGT students in Computer Science in areas of structural proof theory, substructural logics and at the intersection with game development. Most students have a good acquaintance with game engines for their personal interests, however, they may be less exposed to formal logics than desirable during their studies. By appealing to what they know well, my aim is to provide them with pathway leading to acquire new knowledge on some proof systems, with the idea that there could be some benefit for the (simple) game design. For example: 1) knowing the dynamics of some specific game, can we understand it in terms of proofs and derivations in some logic? 2) is there any benefit for the programmer, in implementing the rules of the game in specific logical languages, as opposed to the (low level) language that game engines provide?

Harry Bryant (Swansea University)
Proof Checking for SMT-solving and its application in the Railway Domain

Railways are safety critical therefore the programs that control them must be safe and secure. Alongside the testing process there are tools that apply formal verification techniques that prove these systems meet the requirements and standards. Many of these railway verification tools utilise both SAT solvers and Satisfiable Modulo Theory (SMT) solvers to determine whether a given Safety Property holds. These solvers are very complex, and there is the danger that they prove the correctness of a verification condition when it does not hold. To increase trust and robustness, the objective of this PhD project is to produce a bespoke independent verified checker for our Industrial Partner for those verification results. This will determine whether the certificates produced by Z3 are correct and therefore implies that the theorem in question is valid — producing a validated log rather than a yes/no answer. Since the standards in the railway industry are high, we want as well to have a verified checker of proofs. Therefore, we want to show that if the checking procedure confirms that the proof is valid, then the theorem in question holds. Our Proof Checker is being developed by using the Proof Assistant Coq to prove the correctness of the procedures. Currently a proof-of-concept SAT checker for Ladder Logic Verification has been developed. The basis of both the SAT and SMT proof scripts in Z3 rely on Resolution and therefore work initially focused on writing a checking procedure for Unit Resolution in Coq. This has been proven in Coq to be correct and a mini checker extracted to OCaml. Unit Resolution was chosen because it is the foundation of the old Z3 proof script format. The new format utilises Reverse Unit Propagation (RUP) instead — built on the principle of Resolution. This allows us to use the proof of Unit Resolution as part of the proof that the RUP rule is correct. The proof-of-concept has been written and successfully tested on Industrial examples, with the proof of correctness ongoing. By using formal methods in the railway domain, it gives us assurances that the interlockings are correct before they go through the testing process – saving both time and resources. However, the engineers want a guarantee that the theorem prover Z3 is correct. Therefore, by developing a checking tool it gives further trust and robustness in the development process. The proof-of-concept will be extended to obtain a fully verified proof checker for Z3 proofs to fully meet the needs of the railway verification tools.

Dhurim Cakiqi (University of Birmingham)
Algorithmic syntactic causal identification

Causal identification in causal Bayes nets (CBNs) is an important tool in causal inference allowing the derivation of interventional distributions from observational distributions where this is possible in principle. However, most existing

formulations of causal identification using techniques such as d-separation and do-calculus are expressed within the mathematical language of classical probability theory on CBNs. However, there are many causal settings where probability theory and hence current causal identification techniques are inapplicable such as relational databases, dataflow programs such as hardware description languages, distributed systems and most modern machine learning algorithms. We show that this restriction can be lifted by replacing the use of classical probability theory with the alternative axiomatic foundation of symmetric monoidal categories. In this alternative axiomatization, we show how an unambiguous and clean distinction can be drawn between the general syntax of causal models and any specific semantic implementation of that causal model. This allows a purely syntactic algorithmic description of general causal identification by a translation of recent formulations of the general ID algorithm through fixing. Our description is given entirely in terms of the non-parametric ADMG structure specifying a causal model and the algebraic signature of the corresponding monoidal category, to which a sequence of manipulations is then applied so as to arrive at a modified monoidal category in which the desired, purely syntactic interventional causal model, is obtained. We use this idea to derive purely syntactic analogues of classical back-door and front-door causal adjustment, and illustrate an application to a more complex causal model.

Alec Critten (Swansea University)
Developing user propagators for graph-based SMT reasoning

Satisfiability modulo theories (SMT) solvers are automated reasoning tools known for their expressivity, efficiency, and robust scalability. They are widely used in industrial contexts for verification of computer programs and as part of quality assurance processes. In my PhD research, I am developing a SMT formalisation of graph theory to be applied to a setting in the UK railway industry. The construction of such a custom theory is possible with the recent introduction of the "user-propagator" interface in the Z3 solver. The application of this work is to support the Nodes-Edges Model of graphs that Siemens Mobility has developed to model diagrams of railway systems (known as "scheme plans"), and enable reasoning about properties over these graphs such as reachability and shortest-path in order to satisfy safety constraints ("design rules"). In this talk I will cover ongoing implementation work to bring this theory to fruition through user-propagators.

Aven Dauz (University of Strathclyde)
Modelling cybersecurity games with compositional game theory

Compositional game theory (CGT) and its corresponding Haskell DSL "open-games-engine", have been used for modelling microeconomic games, auctions, and smart contracts. The primary advantages of this compositional approach are

leveraging modularity and code-reuse to construct larger games. Incidentally, game theory has been used to model complex attack-defense scenarios in cybersecurity, with the simplest case modelling the strategic interaction between a single attacker and defender. Scaling these models to accurately reflect real-world attacks and create effective defense strategies remains an active area of research. In this talk I will present an application of CGT for building attack-defense games from smaller components, with a live-code demonstration showing how to use the DSL and generate equilibrium analytics. I'll construct games in the Bayesian and stochastic setting, and show two complete models: a honeypot allocation game and a DDoS attack-defense game between blockchain mining pools.

Abhishek De (University of Birmingham)
Bounded Henkin Quantifiers and the Exponential Time Hierarchy

In logic, quantifiers typically have an implicit linear order of dependencies. Henkin introduced a more general framework for quantifier prefixes, allowing for partial orderings among quantifiers. These quantifiers, now known as Henkin quantifiers, have been extensively studied in logic and have found significant applications in linguistics, arithmetic, and descriptive complexity. Surprisingly, bounded versions of Henkin quantifiers, which restrict the range of these quantifiers, have not been explored. This paper defines bounded Henkin quantifiers and examines their properties from a complexity-theoretic perspective. In particular, we show that the set of predicates definable by quantifier-free formulas prefixed by a bounded Henkin quantifier is exactly NEXP. The proof goes via machine models defined using bounded Henkin quantifiers. Finally, we show that formulas with Henkin quantifiers define a complexity class contained in Δ_2 in the exponential hierarchy and define a natural complete problem for this class.

Lewis Dyer (University of Glasgow)
The similar connected partition problem

We introduce a new graph partitioning problem where given a graph G, equipped with a function f assigning a number to each vertex, we aim to partition vertices of G into a specified number of parts such that each part is connected, satisfies upper and lower part size bounds, and such that the function values of pairs of vertices within a part differ by at most some specific part width δ. We introduce some early algorithms and hardness results for small part sizes, and show this problem is solvable in XP-time parameterised by treewidth and in FPT-time when parameterised by treewidth and maximum part size. We then briefly outline some possible future variants of the problem and discuss some applications in statistical inference for spatial data.

Aidan Evans (University of Cambridge)

An Algebraic Characterization of NC1

Krebs et al. (2007) gave a characterization of the complexity class TC0 as the class of languages recognized by a certain class of typed monoids. The notion of typed monoid was introduced to extend methods of algebraic automata theory to infinite monoids and hence characterize classes beyond the regular languages. We advance this line of work beyond TC0 by giving a characterization of NC1. This is obtained by first showing that NC1 can be defined as the languages expressible in an extension of first-order logic using only unary quantifiers over regular languages. The expressibility result is a consequence of a general result showing that finite monoid multiplication quantifiers of higher dimension can be replaced with unary quantifiers in the context of interpretations over strings, which also answers a question of Lautemann et al. (2001).

Murdoch Gabbay (Heriot-Watt University)
A shallow arithmetisation of first-order validity

Arithmetisation means translating validity in a model M of some assertion φ, into some property RM of a polynomial $P(\varphi)$, such that (to a high degree of certainty) φ is valid in M if and only if $P(\varphi)$ has property RM. Arithmetisation is useful in cryptography, because many cryptographic protocols work by exploiting properties of finite fields to prove things like "R holds of a polynomial p".

We note a compositional shallow translation from a fragment of first-order logic with equality, into univariate polynomials. This fragment is expressive, and in particular it is easily powerful enough to express computationally significant things like inductive definitions and combinator reduction. Thus, by arithmetising inductive definitions our translation also arithmetises computation, amongst other things.

Techniques already exist to arithmetise computation by coding it in an abstract machine whose computation has been arithmetised by hand; i.e. by a deep embedding in some preconstructed monolithic cryptographic artefact. What distinguishes our translation is its shallowness and compositionality: it translates logical connectives directly to operations on polynomials, without a deep encoding in an abstract machine.

The translation is, as a piece of pure logical manipulation, deceptively simple and straightforward, but we shall see that the outcome is surprisingly powerful and expressive. Future work is to operationalise it to obtain a useful and practically applicable zero-knowledge or succinct proof-scheme, and we conclude by briefly discussing the barriers to doing this.

Murdoch Gabbay (Heriot-Watt University)
A declarative approach to specifying distributed algorithms using three-valued modal logic

Coalition Logic is a three-valued modal fixed-point logic designed for declaratively specifying and reasoning about distributed algorithms, such as the Paxos consensus algorithm.

Coalition Logic adopts a declarative approach, specifying the logic of computation as an axiomatic theory, without prescribing control flow. Notably, message-passing and (algorithmic) time are not explicitly modeled, distinguishing this from approaches like TLA+. This abstraction emphasises the logical essence of distributed algorithms, offering a novel perspective on their specification and reasoning.

I will demonstrate the applicability of this approach to the Paxos consensus algorithm by presenting it as a logical theory and deriving its standard correctness properties.

I see Coalition Logic as a versatile tool for specifying and reasoning about distributed algorithms, offering a new lens through which distributed algorithms can be specified, studied, and checked.

Nathan Flaherty (University of Liverpool)
Fast and Safe Scheduling of Robots

Mobile robots are becoming an increasingly common part of scientific work within laboratory environments and therefore effectively coordinating them to complete necessary tasks around the laboratory whilst preventing collisions is of great practical interest. We model this theoretically by having a graph $G = (V, E)$ along with some tasks and robots, each task is on a vertex and has a certain duration, the time taken to complete the task, we then have robots which are agents that can move between adjacent vertices on the graph. The problem being to find schedules for the robots which complete all the tasks in G without any two robots sharing the same vertex or edge at any given time. In this talk I reintroduce an algorithm for Robot scheduling on a path with extensions for use on other graph classes, and present an Integer Linear programming formulation of this problem for use on all graphs. This leads onto experimental analysis of the algorithm on paths and lattice graphs. This is work done jointly with Duncan Adamson, Igor Potapov and Paul G. Spirakis.

Marek Jezinski (Swansea University)
Creating Synthetic Test Data for Rail Design Tools

The rail industry uses various software tools, e.g., to design scheme plans. However, how can such tools be tested? The manual design of rail artefacts such as scheme plans is laborious, costly, and an error-prone process. Thus, it seems appropriate to generate synthetic artefacts with a view to reducing the cost and workload for producing test data, and possibly with precise knowledge of where there are faults in these artefacts and where there are not.

In this talk, we present a method to generate artificial scheme plans based on Genetic Algorithms [1]. These algorithms simulate how a population evolves over time. Genetic Algorithms operate on data that is represented in the form of 'chromosomes' (standing for the single members of a population). The chromosomes themselves consist of 'genes' (providing the specific characteristics of a single member of the population). Genetic Algorithms start with an initial, randomly generated population. Evolution is then mimicked by iteratively producing new generations. To this end, offspring are generated using two operations: crossover (splitting and recombining selected genes of different members of the current population) and mutation (randomly modifying already recombined genes). Then, the next generation is selected by applying a fitness function to both, the members of the current generation and the newly generated offspring. This guarantees the survival of the fittest. The Genetic Algorithm terminates when the fitness of the current population exceeds a defined threshold [2, 3].

It is an important feature of the presented method that it includes the possibility of controlled fault injection, where we know exactly where there is a design flaw. In the context of scheme plans, a concrete example of a fault would be the following: placing a balise too close to a point's toe, as this contradicts the design rule that requires a minimum distance between a point's toe and a balise. By utilising genes that represent faults, it is possible to have populations which include faulty members. By choosing a fitness function that requires a certain number of flaws for survival, one can guarantee that all members of the final population will exhibit a pre-determined minimal/maximal number of flaws. On the technical level, our method has been realised in the programming language Python. Using Genetic Algorithms, our Python program generates scheme plans by attaching tiles with several passing loop variants, such as straight parallel tracks, points junctions, and diamond junctions. The user can choose several parameters, e.g., the initial population size or the fitness level for termination. From the final generation, one scheme plan is exported into a bespoke data format from Siemens. This allows for the testing of Siemens' tools. Scheme plan generation and scheme plan export is quick, within the order of seconds only.

We will demonstrate a running tool that produces artificial railway scheme plans using Genetic Algorithms. The scheme plans include faults documented within a log file providing the fault locations. This allows us to thoroughly test if the existing design checkers for scheme plans work correctly. Such tools have been developed by various companies and research groups, including Siemens' data checker [4], the ovado tool described in [5, 6] and the junction tool suite developed by Luteberget documented in [7].

Overall, this work provides a preliminary example demonstrating that the generation of synthetic test data using Genetic Algorithms is a viable approach in testing software tools used in railway design. It is future work to evaluate the quality

of the automatically generated test data, e.g., by developing notions of coverage (whether the fault frequency is 'realistic', all 'normal' situations are generated, or if corner cases are considered).

References

[1] A. E. Eiben and J. E. Smith, Introduction to Evolutionary Computing, 2nd ed., Springer, 2015.
[2] M. Z. R. Khan and A. K. Bajpai, "Genetic Algorithm And Its Application In Mechanical Engineering," International Journal of Engineering Research & Technology, vol. 2, no. 5, 2013.
[3] T. Harrison, Exploring the Feasibility of Using Genetic Algorithms for Generating Railway Map Test Data, Dissertation, Swansea, 2024.
[4] M. Banerjee, V. Cai, S. Lakhsmanappa, A. Lawrence, M. Roggenbach, M. Seisenberger and T. Werner, "A Tool-Chain for the Verification of Geographic Scheme Data," RSSRail 2023, LNCS, vol. 14198, pp. 211–224, 2023.
[5] R. Abo and L. Voisin, "Formal implementation of data validation for railway safety-related systems with OVADO," SEFM'13, LNCS, vol. 8368, pp. 221–236, 2014.
[6] T. Lecomte, L. Burdy and M. Leuschel, "Formally Checking Large Data Sets in the Railways," in Workshop on the experience of and advances in developing dependable systems in Event-B, 2012.
[7] B. Luteberget, Automated Reasoning for Planning Railway Infrastructure, PhD Thesis, University of Oslo, Faculty of Mathematics and Natural Sciences, 2019.

Thomas Karam (University of Oxford)

Adapting some basic matrix rank properties to the ranks of tensors

Tensors are higher-dimensional generalisations of matrices, and likewise the main notion of complexity on matrices — their rank — may be extended to tensors. Unlike in the matrix case however, there is no single canonical notion of rank for tensors, and the most suitable notion often depends on the application that one has in mind. The most frequently used notion so far has been the tensor rank (hence its name), but several other notions and their applications have blossomed in recent years, such as the slice rank, partition rank, analytic rank, subrank, and geometric rank. Unlike their counterparts for the rank of matrices, many of the basic properties of the ranks of tensors are still not well understood. After reviewing the definitions of several of these rank notions, I will present a number of results of a type that arises in many cases when one attempts to generalise a basic property of the rank of matrices to these ranks of tensors: the naive extension of the original property fails, but it admits a rectification which is simultaneously not too complicated to state and in a spirit that is very close to that of the original property

from the matrix case.

Thomas Karam (University of Oxford)
Communication of theoretical computer science notions, and mathematics

This teaching talk will focus on how to transmit notions in theoretical computer science in ways that prepare the recipient well for the input of mathematicians and furthermore do not create a tradeoff with the primary goals of theoretical computer science exposition. For concrete illustration we will discuss the case of Shannon entropy, a notion shared by the TCS and mathematics communities, which has an interesting history inbetween them: it was originally devised for communications engineering purposes, but then led to insights illuminating some distant topics in pure mathematics (and some other topics in physics), while at the same time continuing to have the kinds of applications that it had originally been conceived for.

Alasdair Lambert (University of Strathclyde)
Type Directed Programming for Programming Education

Live programming is a widely used technique in programming education. While it can be an effective tool for introductory classes, expert blindness is a common pitfall. It is also easy for students to assume that the problem solving ability is inherent to the lecturer and that they could never manage this independently.

To ameliorate this we propose Type Directed Programming as a teaching tool. This is where the type signature of a function guides its implementation details. Using type declarations as part of the live programming paradigm allows the lecturer to show that there is no magic, merely someone used to reading types.

Laura Larios-Jones (University of Glasgow)
Structural Parameters for Dense Temporal Graphs

A temporal graph consists of a static graph and a function that dictates when its edges are active. This allows us to model networks where the time at which connections occur is important. Unfortunately, many temporal problems are more computationally complex than their static counterparts. We look for temporal parameters such that, when these are small for the input of a temporal problem, we can efficiently solve the problem at hand. Many temporal parameters currently in use rely on some notion of sparsity in either the graph or function. We introduce three parameters that are small for dense and highly structured temporal graphs. These parameters are temporal cliquewidth, temporal modular-width, and temporal neighbourhood diversity. We also give examples of temporal problems which are tractable with respect to these parameters.

Louis Lemonnier (University of Edinburgh)

A recipe for the semantics of reversible programming

In this presentation, we explore the foundational elements required to interpret reversible programming within a categorical framework. We use the symmetric pattern-matching language introduced by Sabry, Valiron, and Vizzotto as a reference point, and we incorporate several improvements. We show that inductive data types and recursion can also be effectively modelled in this setting. However, these results do not straightforwardly extend to the pure quantum case. We provide insights into the challenges encountered and propose potential directions for addressing these limitations, for example with guarded recursion.

Jessica Newman (University of Southampton)
Alternating-Time Temporal Logic with Dependent Strategies

Alternating-Time Temporal Logic (ATL) is a logical system for multiagent strategic reasoning, in which formulae express temporal properties that coalitions of agents can enforce by concurrently fixing a strategy played across an infinite sequence of normal-form games. We extend this with the ability to specify an order in which agents sequentially fix their strategies, where agents that move later are able to select a strategy dependent on those of the previous agents; this gives a richer way of expressing coalitional power, since agents can enforce different outcomes depending on which of their opponents they are able to respond to. We characterise the sets of outcomes that are possible for a coalition to enforce over different move orderings in a normal-form game, and use this to give a sound and complete axiomatisation of this extended ATL with Dependent Strategies (ATLDS) as well as time complexity bounds for its key decision problems.

Olga Petrovska (Swansea University)
The Art of Teaching Theory Across Diverse Backgrounds

As higher education becomes more accessible, with the emergence of pathways like degree apprenticeships, we are seeing classrooms filled with an increasingly diverse student population. While this diversity is welcomed development, it also presents certain challenges, for instance when introducing highly theoretical concepts to learners with varied backgrounds.

In this talk, I will explore strategies to make abstract ideas accessible for all students. Using the example of a module on Algorithms and Automata, I will discuss storytelling as a pedagogical technique that can bring these abstract concepts to life. Additionally, I will present the ways in which students and educators can benefit from generative AI, while also addressing the potential pitfalls of misuse. Finally, I will present some ideas on assessment designs that encourage authentic student work, ensuring that the integration of AI enhances rather than undermines the educational process.

Benjamin Plummer (University of Southampton)
Coalgebraic Traces by Following Strategies in Two-player Games

In this talk, we introduce the process-theoretic notion of trace into two-player controller-versus-environment games. We demonstrate that the trace semantics at a given game state correspond to the set of subsets of plays that the controller can force, capturing the controller's strategic influence. Building on the coalgebraic framework for traces developed by Hasuo et al., we identify a suitable monad to represent the two layers of branching in these games. This monad, constructed using a weak distributive law to combine two powerset monads, effectively computes what the controller can enforce in a single step. We also present a novel categorical description of strategies, and discuss how our approach easily extends to games with probabilistic environments. This work provides fresh insights into the interaction between strategies and traces, and lays a coalgebraic foundation for game-based program synthesis.

Riu Rodríguez Sakamoto (University of Strathclyde)
Decorated Para for linear quadratic regulators

One of the classical examples of optimal control are regulators of linear dynamical systems with quadratic costs. We employ a characterization of the Para construction as certain Spans with left leg projections due to Capucci and Jaz Myers, together with a dualization of the theory of decorated cospans over vector spaces, to define 'decorated Para' as a category of linear quadratic regulators. We then prove that the algebraic Riccati equation forms a lax functor into a category of cost functions and smooth maps between them.

Yury Savateev (University of Hertfordshire)
Visualising the Hyperbolic Plane

Common visualisation techniques for hyperbolic plane usually are limited to standard models like Poincare disk model. This significantly limits the part of the hyperbolic plane that can be displayed at a workable scale, which diminishes usefulness. We propose a new way to visualise the hyperbolic plane that allows more of the plane to be displayed in a useful manner.

Zephyr Verwimp (University of Oxford)
On hypergraph colouring variants and inapproximability

I will discuss (hyper-)graph colouring problems, and how their computational complexity is captured by a certain notion of symmetry. I will then present recent results on inapproximability of conflict-free and linearly-ordered hypergraph colouring.

Tansholpan Zhanabekova (University of Liverpool)

Semantic Flowers for Good-for-Games and Deterministic Automata

We present an innovative approach for capturing the complexity of ω-regular languages using the concept of flowers. This semantic tool combines two syntax-based definitions, namely the Mostowski hierarchy of word languages and syntactic flowers. The former is based on deterministic parity automata with a limited number of priorities, while the latter simplifies deterministic parity automata by reducing the number of priorities used, without altering their structure. Synthesising these two approaches yields a semantic concept of flowers, which offers a more effective way of dealing with the complexity of ω-regular languages. This letter provides a comprehensive definition of semantic flowers and shows that it captures the complexity of ω-regular languages. We also show that this natural concept yields simple proofs of the expressive power of good-for-games automata.

CONFERENCE SPOTLIGHT: ALGO

by Matthias Bentert (Technische Universität Berlin)

ALGO 2025 took place September 15–19 in Warsaw with the six colocated conferences European Symposium on Algorithms (ESA), International Symposium on Parameterized and Exact Computation (IPEC), Symposium on Algorithmic Approaches for Transportation Modeling, Optimization and Systems (ATMOS), Workshop on Approximation and Online Algorithms (WAOA), International Symposium on Algorithmics of Wireless Networks (ALGOWIN, formally ALGOSENSORS), and International Symposium on Algorithmic Aspects of Cloud Computing (ALGOCLOUD)[1]. I presented a paper at ESA, was in the PC for IPEC, and will share my experience here. I am a postdoc working on (parameterized) algorithms, but this spotlight is written with a broad TCS audience in mind. Before I start, I just quickly want to mention that Warsaw was impressive with a lot of larger and smaller parks around the university campus. These parks were very well maintained and just overall very clean and lovely.

Overall, the organization of ALGO was very good. The local organizers (Jadwiga Czyżewska, Tomáš Masařík, Marcin Pilipczuk, Jakub Radoszewski, Paweł Rzążewski, and Wiktor Zuba) had everything under control and were very helpful in answering questions and helping where needed. The expected bag for participants have an unexpectedly good quality and everyone could choose between a red, green, or blue bag. There were three to four parallel sessions with some plenary sessions for keynote talks. Due to the fact that each session chair followed the schedule very closely and the different rooms were very close to one another, it was easily possible to switch between talks even during a session; though I did not use this possibility myself.

As with any event of this size, there are bound to be a couple of small hiccups in the organization. As far as I can tell, there were only two minor points of critique regarding ALGO. First, the organizers admitted that they underestimated how long it would take to create invoices for almost 300 participants. This led to a situation where some people got it relatively late (during the conference). Second, the organizers decided for a distributed algorithm when it comes to organizing the sessions. In particular, how to distribute the 17 minutes into set-up, talk, and questions was delegated to the session chairs and getting hold of a USB-stick for

[1]The conferences are sorted by the number of talks at ALGO in descending order.

transferring slides to the local computers we were asked to use for the presentation was sometimes a non-trivial task. I would have liked the possibility to upload the slides beforehand to a common repository, but this is more of a suggestion for the future rather than a complaint. Interestingly, the organizers decided to not have a welcoming session on Monday morning. Instead, the conference immediately started with talks of accepted papers. I am not sure whether I liked this or not, but it was definitely an interesting idea and I am curious what other people think of this decision.

For the social event and the conference dinner, we went to a summer residence of John III Sobieski, the elected (!) king of Poland between 1674 and 1696. There, we learned a bit about King John, his wife Marie Casimire Louise, and the history of Warsaw. King John was a very capable military leader, but also well educated in science and possibly the only Polish King who married the love of his live. The conference dinner took place in a former green house.

Regarding the technical contributions, I cannot possibly write about the many different talks and keynote talks I listened to at ALGO. Fortunately, I do not have to as most (maybe even all) keynote talks will have summaries in the proceedings. These proceedings are expected to be published before the end of the year. I only want to highlight a contribution by Bernhard Haeupler, which is not directly related to computer science, but about the scientific environment overall. He talked about his experience in science having both ADHD and dyslexia. The main point he made was that it is not the people that do not work correctly, but it is the environment that does not let them work correctly. So it is up to all of us to create an environment in which we celebrate our differences and do not stigmatize neuro diversity.

The next part is about the business/community meetings. Since many of them ran in parallel, I can only report on ESA and IPEC here. For ESA, 115 out of 372 submissions were accepted (for comparison: last year, 103 out of 333 submissions were accepted). Due to the larger number of accepted papers, the talks were shortened from 20 minutes slots to 17 (including questions and changing speakers in both). This year, 500 lines + appendix were allowed which replaced the 12 pages + appendix rule from before. This change was overall very well received by the community (according to a vote during the business meeting). Lastly, the idea was brought up to rename track B to track E (for engineering/experiments).

For IPEC, 30 out of 61 submissions were accepted. Unfortunately, not a single student paper was accepted. This led to a discussion whether the definition of a student paper should be softened to allow papers with supervisors as long as the student did most of the work. This discussion is still ongoing. Another notable discussion in the community meeting—not only in the context of IPEC—was the consideration to recognize additional aspects of the research procedure. Ideas of awarding a best figure award, a best talk reward, or a best review

award were brought up. If you have a strong opinion regarding these or other ideas for recognizing different research aspects, please get in touch with the steering committee of IPEC. Finally, as someone working in parameterized algorithmics, I would like to point out two useful tools in the context that were informally presented at IPEC: `https://vaclavblazej.github.io/parameters` and `https://yutookada.com/tools/graph-parameters`. The former is an extensive survey of different graph parameters and their relations. It is managed by Václav Blažej and if you want to have additional parameters included or find a new connection between parameters, get in touch with him, so we can all benefit from a complete overview. The latter is maintained by Yuto Okada and helps you keep track of what an FPT/W[1]-hardness result for a certain parameter tells you about the same problem parameterized by different (related) parameters.

Speaking of graph parameters, the results of this years PACE challenge on Dominating Set and Hitting Set were presented. Most submitted solvers for the exact track used a combination of reduction rules and ILP- or SAT-solvers. In the heuristic track, different combinations of reduction rules, local search, greedy algorithms, and more were tried. Next years challenge was also announced: Agreement Forests, a problem from computational biology. There will be three tracks: an exact/parameterized track, a heuristic track, and a new lower bounds track. For the exact/parameterized track, the organizers invite you to ask for interesting parameters/decompositions you would like to have in addition to the input. Unfortunately, the NETWORKS program ends after 10 years and thus also their financial support for PACE comes to an end. Consequently, there will probably not be any prize money for the winners in the next iteration. You can still win cool medals, though.

To conclude this small report, I want to thank all the chairs of the conferences for organizing wonderful conferences:

- Anne Benoit, Haim Kaplan, and Sebastian Wild (ESA)
- Akanksha Agrawal and Erik Jan van Leeuwen (IPEC)
- Jonas Sauer and Marie Schmidt (ATMOS)
- Jannik Matuschke and José Verschae (WAOA)
- Othon Michail and Giuseppe Prencipe (ALGOWIN)
- Domenico Garlisi and Dimitris Chatzopoulos (ALGOCLOUD)

Hopefully, next years ALGO will be similarly great, so see you all in l'Auqila (100km east of Rome in Italy). And by the way, the best way to get there is apparently to take a bus from Rome taking about 2 to 2.5 hours. Thus, we might all be forced (or tempted?) to take a stop at Rome. On top of that, we might have to spend some time there so we might feel pressured to visit the Colosseum, the Pantheon, the Vatican, or ... Oh my, oh my, the things we all collectively endure for science.

Report on STACS'2025

42nd International Symposium on Theoretical Aspects of Computer Science Jena, Germany, March 4 — 7, 2025

Florian Chudigiewitsch
University of Lübeck

The 42nd International Symposium on Theoretical Aspects of Computer Science (STACS) took place from March 4 to March 7, 2025, in Jena, Germany. As a pre-conference workshop, the Theory Day of the German Informatics Society took place on March 3 and 4. In this report, I want to share my personal experience attending the conference as a fourth-year PhD student.

The conference covered a wide variety of topics in theoretical computer science, organized in two parallel tracks. Track A, chaired by Nguyen Kim Thang and Michał Pilipczuk, focused on algorithms, data structures, and complexity, while track B, chaired by Olaf Beyersdorff and Elaine Pimentel, focused on automata, logic, semantics, and theory of programming. All talks had a length of 20 minutes, with sessions having four to five talks.

With the two tracks running in parallel in adjacent rooms, I was able to attend about half of the talks. From this sample, I want to highlight three presentations that stood out for their clarity, depth, and engaging delivery:

- "How to play the Accordion: Uniformity and the (non-)conservativity of the linear approximation of the λ-calculus" by Rémy Cerda
- "Residue Domination in Bounded-Treewidth Graphs" by Jakob Greilhuber
- "Toward Better Depth Lower Bounds: Strong Composition of XOR and a Random Function" by Ivan Mihajlin

In addition to the regular talks, the invited and tutorial presentations offered a broader perspective on current trends in theoretical computer science. There was a two-hour tutorial given by Albert Atserias on "Proof complexity and its relations to SAT solving" on Tuesday at the start of the conference. On Wednesday, Daniel Dadush gave the first of the invited talks with his algorithmically focused talk "A Strongly Polynomial Algorithm for Linear Programs with at most Two Non-zero Entries per Row or Column". On Thursday, Anupam Das' talk "Algebras for automata: reasoning with regularity" represented the logical aspects of the conference; and on Friday, Susanna F. de Rezende gave us insights into current complexity-theoretic research with her talk "Some Recent Advancements in Monotone Circuit Complexity".

Overall, the scientific programme struck a very pleasant balance between breadth and depth, offering something valuable for every participant.

Some interesting statistics on the conference were presented in the business meeting on Wednesday: Submission numbers were very high, track A received 202 submissions of which 55 were accepted, while track B received 57 submissions of which 17 were accepted. In total, there were 259 submissions, of which 72 were accepted, resulting in an acceptance rate of 28%. Submissions came from across the globe, with Germany and France slightly leading in numbers.

The co-location of the conference with the Theory Day also resulted in a high number of attendees for both venues, with 67 for Theory Day and 136 for STACS. Five papers were presented online in a live talk. Selected papers will be published in full on invitation to ACM Transactions on Computation Theory, Logical Methods in Computer Science, and, in outstanding cases, TheoretiCS. Finally, the next conference location was revealed to be Grenoble in France, as the host location of STACS alternates between France and Germany.

I was thoroughly impressed by the local arrangements. Jena is a smaller but very charming town in the east of Germany, and quite worth visiting. Thanks to the local organization committee chaired by Marlene Gründel, the conference was running very smoothly, to the point that for me, the fact that there were no noticeable issues became noticeable itself. The website was organized, which made it easy to get all the relevant information, and choose between the thoughtfully themed and titled sessions.

There were also plenty of occasions to get to know fellow researchers besides the scientific programme, such as the reception on Tuesday (which also featured a short theatrical performance given by the organizers), and the conference dinner on Thursday, both being warm and welcoming events that encouraged social interaction.

Besides this, the conference benefitted from an array of well-thought-out and playful ideas from the local organizers. Besides the usual flyers, we were gifted small puzzles at registration. A particularly delightful social game involved each participant receiving, along with their name tag, the name of a historical figure connected to Jena. The goal was to find your match – another attendee whose historical alter ego had some meaningful connection to yours. This creative activity helped break the ice and sparked many conversations. Background information and local context were provided on personalized flyers. Finally, the poster contained a "hidden message", with the first person to decipher it being awarded a bottle of champagne at the conference dinner.

For me, all these activities (which I can only imagine entailing a tremendous amount of work to organize) elevated the conference from a place of valuable scientific exchange to a fun, social, and memorable event.

Overall, I enjoyed STACS 2025 very much, with ample opportunity to form

Figure 1: The logo of STACS 2025. Can you see the hidden message?

new connections, get new ideas for my research, and gain insight into other adjacent areas within theoretical computer science. For those still reading, I want to leave you with a small taste of the brain teasers we were enjoying: In Figure 1, you find the logo of the conference. In it, there is a hidden message. Can you solve the riddle?

In Memoriam
Christos Levcopoulos
*05.07.1956 - †10.06.2025

Our dear colleague and friend Christos Levcopoulos passed away on June 10 2025 at the age 68, way too early. He was suffering from skin cancer that was rapidly spreading into other organs the last months of his life.

Christos was born in 1956 in Thessaloniki, Greece, and moved to Sweden in 1976. In 1981, he obtained the Bachelor degree in Administration and System Sciences with specialization in Computer Science, at Linköping University.

I first met Christos in 1984, when he became my first PhD student at the Department of Computer Science of Linköping University. Soon we became not only colleagues but also friends for the rest of our lives. During that time, I was researching on decompositions of geometric figures and that topic proved to be perfect for Christos. It soon became clear that he had great talent for algorithmic thinking, especially in computational geometry. Soon he improved several of my results on geometric decompositions. In 1987, he defended his strong PhD thesis on heuristics for minimal divisions of polygons. Among other things, it included tight bounds on minimum number covering of polygons with rectangles [1] and close approximation algorithms for minimum length partitions of isothetic polygons into rectangles, based on a novel thickest-first approach (e.g., see [2]). Christos kept returning to the problems of optimal decompositions of geometric figures in several later papers. In 1989/90 we moved to Lund University together. Shortly after, Christos published several interesting papers on adaptive sorting with his first PhD student Ola Petersson (e.g., see [6, 7]). Then came Christos' pioneering geometric results on the first O(1)-approximation of minimum-weight triangulation in polynomial time [9] and on geometric spanners showing that nearly optimal cheap geometric spanners can be constructed from the Delaunay triangulation in linear time [4] (the first result was established jointly with his PhD student Drago Krznaric). Both results are highly appreciated today. Christos successfully pursued his research on minimum weight triangulation (e.g., [10]) and geometric spanners, jointly with his PhD students (among other things with Joachim Gudmundsson [14]), and later also with Giri Narasimhan and Michiel Smid [13]. He also obtained interesting results on geometric clustering (e.g., see

[11]) and geometric minimum spanning trees (e.g., see [8]), mostly with his PhD students. In 1997 Christos became an associate professor. Soon he had enough strong publications to apply for a promotion to full professor but we could not convince him to do this. He had other values and an academic career was not so important to him. Between 2000 and 2007 Christos was head of our section for theoretical computer science and also head of the entire department of computer science during a substantial part of this period. In the 2000s, Christos extended his research topics by strong contributions among other things on distance oracles [15] and distributed computing [16].

Summarizing, Christos published +120 peer reviewed conference articles (including STOC, SODA, ICALP, ESA, SoCG) and journal articles (e.g., SIAM J. Computing, J. Algorithms, ACM Transactions on Algorithms, Discrete Computational Geometry, Information and Computation, JCSS, TCS, Algorithmica etc.). He was an editor of the Journal of Discrete Algorithms and served on program committees of several TCS conferences including SoCG and SWAT.

Christos was a talented excellent researcher, dedicated successful supervisor, solid teacher and leader. He was always very kind and balanced, ready to offer constructive help to anybody in his environment. We shall miss him very much.

Selected Christos Levcopoulos' publications

[1] C. Levcopoulos, A. Lingas. Covering Polygons with Minimum Number of Rectangles. *STACS* 1984: 63-72

[2] C. Levcopoulos. Fast Heuristics for Minimum Length Rectangular Partitions of Polygons. *SCG* 1986: 100-108

[3] C. Levcopoulos. An $\Omega(\sqrt(n))$ Lower Bound for the Nonoptimality of the Greedy Triangulation. *Inf. Process. Lett.* 25(4): 247-251 (1987)

[4] C. Levcopoulos, A. Lingas. There Are Planar Graphs Almost as Good as the Complete Graphs and Almost as Cheap as Minimum Spanning Trees. *Algorithmica* 8(3): 251-256 (1992) (Preliminary version *Optimal Algorithms* 1989: 9-13)

[5] C. Levcopoulos, A. Lingas, J.-R. Sack. Heuristics for Optimum Binary Search Trees and Minimum Weight Triangulation Problems. *Theor. Comput. Sci.* 66(2): 181-203 (1989) (Preliminary version *ICALP* 1987: 376-385)

[6] C. Levcopoulos, O. Petersson. Adaptive Heapsort. *J. Algorithms* 14(3): 395-413 (1993) 1992

[7] C. Levcopoulos, O. Petersson. Sorting Shuffled Monotone Sequences. *Inf. Comput.* 112(1): 37-50 (1994)

[8] D. Krznaric, C. Levcopoulos, B.J. Nilsson. Minimum Spanning Trees in d Dimensions. *ESA* 1997: 341-349

[9] C. Levcopoulos, D. Krznaric. Quasi-Greedy Triangulations Approximating the Minimum Weight Triangulation. *J. Algorithms* 27(2): 303-338 (1998) (Preliminary version *SODA* 1996: 392-401)

[10] C. Levcopoulos, D. Krznaric. A Linear-Time Approximation Scheme for Minimum, Weight Triangulation of Convex Polygons. *Algorithmica* 21(3): 285-311 (1998) (Preliminary version *SODA* 1997: 518-527)

[11] D. Krznaric, C. Levcopoulos. Fast Algorithms for Complete Linkage Clustering. *Discret. Comput. Geom.* 19(1): 131-145 (1998) (Preliminary version *ISAAC* 1995: 392-401)

[12] C. Levcopoulos, D. Krznaric. A Linear-Time Approximation Scheme for Minimum, Weight Triangulation of Convex Polygons. *Algorithmica* 21(3): 285-311 (1998) (Preliminary version *SODA* 1997: 518-527)

[13] C. Levcopoulos, G. Narasimhan, M.H. M. Smid. Improved Algorithms for Constructing Fault-Tolerant Spanners. *Algorithmica* 32(1): 144-156 (2002) (Preliminary version *STOC* 1998: 186-195)

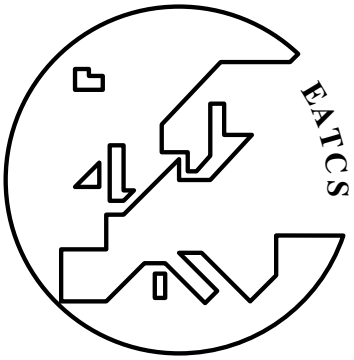
[14] J. Gudmundsson, C. Levcopoulos, G. Narasimhan. Fast Greedy Algorithms for Constructing Sparse Geometric Spanners. *SIAM J. Comput.* 31(5): 1479-1500 (2002)

[15] J. Gudmundsson, C. Levcopoulos, G. Narasimhan, M.H. M. Smid. Approximate distance oracles for geometric spanners. *ACM Trans. Algorithms* 4(1): 10:1-10:34 (2008) (Preliminary version SODA 2002: 828-837)

[16] L. Gasieniec, J. Jansson, C. Levcopoulos, A. Lingas. Efficient assignment of identities in anonymous populations. *Inf. Comput.* 303: 105265 (2025)

BEATCS no 147

European Association for Theoretical Computer Science



E A T C S

204

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`,