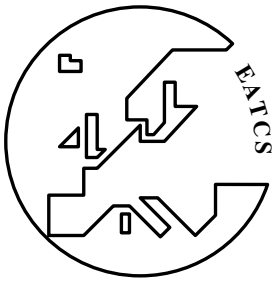
ISSN 0252–9742

Bulletin

of the

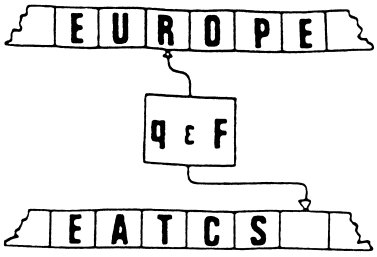
European Association for Theoretical Computer Science

EATCS



Number 135 October 2021

Council of the European Association for Theoretical Computer Science



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

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

Past Presidents:

Maurice Nivat	(1972–1977)	Mike Paterson	(1977–1979)
Arto Salomaa	(1979–1985)	Grzegorz Rozenberg	(1985–1994)
Wilfred Brauer	(1994–1997)	Josep Díaz	(1997–2002)
Mogens Nielsen	(2002–2006)	Giorgio Ausiello	(2006–2009)
Burkhard Monien	(2009–2012)	Luca Aceto	(2012–2016)
Paul Spirakis	(2016–2020)		

Secretary Office:	Efi Chita	Greece
	Emanuela Merelli	Italy

EATCS Council Members

EMAIL ADDRESSES

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

Bulletin Editor: Stefan Schmid, Berlin, Germany
Cartoons: DADARA, Amsterdam, The Netherlands

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

All contributions are to be sent electronically to

bulletin@eatcs.org

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

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

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

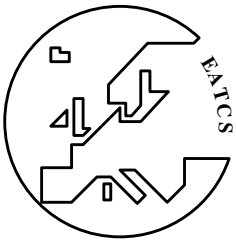
The details can be found at `http://www.eatcs.org/bulletin`, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email `bulletin@eatcs.org`.

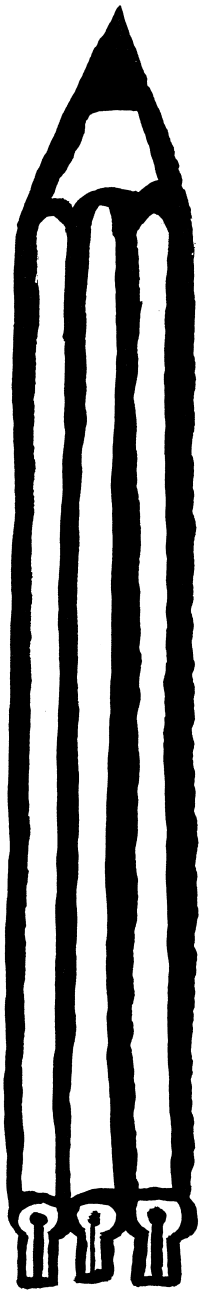
Deadlines for submissions of reports are January, May and September 15th, respectively for the February, June and October issues. Editorial decisions about submitted technical contributions will normally be made in 6/8 weeks. Accepted papers will appear in print as soon as possible thereafter.

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

Table of Contents

EATCS Matters



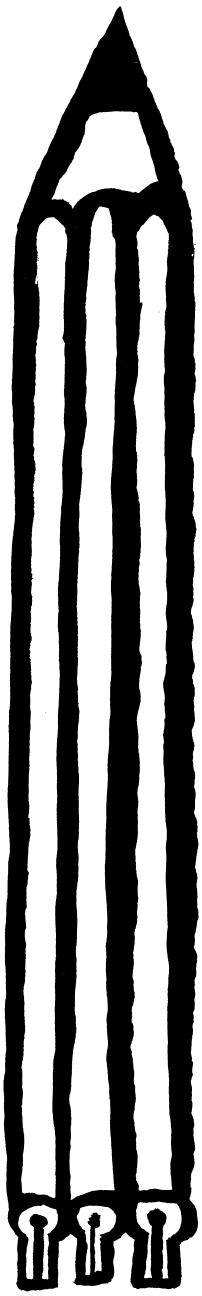


Dear EATCS members,

I hope my letter finds you and your family safe and in good health. While the global coronavirus pandemic still has a major impact on our lives and limits some of our scientific activities, I see many positive signs that there are brighter days ahead of us. Indeed, the pandemic situation seems to be getting normalized in many places of the world and I hope that with the progress in global vaccination we will soon be seeing better times. And so, even though in the last year and a half almost all scientific activities have been run in the online setting, we now see more and more activities taking places in the hybrid setting or even as physical events. I see also active plans for the return of in-person research collaborations and in-place workshops and conferences.

While the current situation is a major challenge for our scientific community, it is great to see that our community has been developing many new and exciting initiatives: we see more online seminars and workshops, a lot of remote collaborative research, and for many of us free or low-cost online conferences provide a great opportunity for broader participation and the increase of visibility of our research through these events. And most importantly, we see some fantastic research done by our community; and so I take the opportunity to wish you all the best and much success for your work.

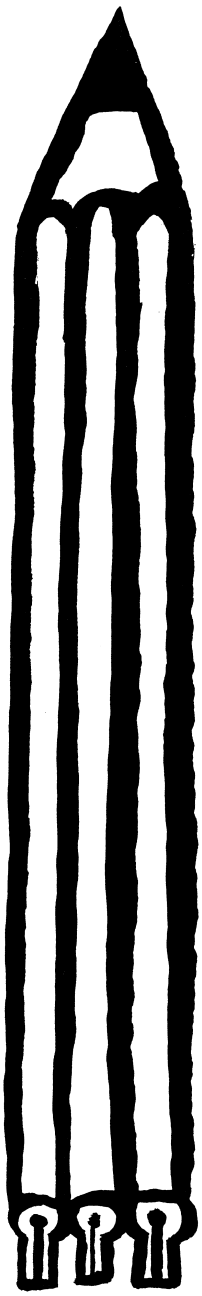
ICALP 2021, the EATCS flagship conference, was run in the online format again. As



always though, the conference had an impressive scientific program highlighting the strength of the research across many areas within theoretical computer science. On behalf of the entire community and the EATCS I would like to thank the Programme Committee led by the chairs Nikhil Bansal and James Worrell, the organizers at the University of Glasgow, led by Simon Gay, and especially - all the participants, for their fantastic efforts that helped to make the ICALP 2021 conference a great success. We plan to have a detailed report of ICALP 2021 in the next issue of the Bulletin.

We also had very successful three EATCS partner conferences: ESA 2021, MFCS 2021, and DISC 2021. While ESA ran fully online, MFCS and DISC had been organized in a hybrid format, with some local participants - a very welcomed feature that we hope to see more and more in the months to come. Running all these four conferences in the online or the hybrid setting allowed to ensure free or low registration fees, and this led to very good numbers of participants and to the increase of visibility of research in these events.

In this issue of the Bulletin, you will find the calls for nominations for the EATCS Award, the Presburger Award, the EATCS Distinguished Dissertation Award, and the EATCS Fellows. As usual, we are lucky to have very strong committees for each of the awards, and I thank all the award committee members in advance for their important service. I strongly encourage you to send nominations for these prestigious awards. I am aware of the fact that we are all very busy and that it takes time and efforts to prepare strong



nominations, but our best researchers and best papers can only win awards if they are nominated. Moreover, awards put areas of, as well as inspirational figures in, theoretical computer science in the spotlight and can serve to inspire young researchers. I look forward to seeing who the award winners will be and to working with all of you to make the EATCS even more influential than it already is.

As usual, the October issue of the Bulletin has the first Call for Papers for ICALP 2022, the flagship conference of the EATCS and an important meeting of the theoretical computer science community world-wide. The 49th International Colloquium on Automata, Languages, and Programming (ICALP 2021), will be held on July 4-8, 2022 in Paris, France, and online. The conference is planned to be in the hybrid format, though we hope the talks to be mostly delivered in person and we hope for a very good in-place audience. We have a great list of invited speaker and expect a fantastic scientific program selected by the PCs led by the chairs David Woodruff and Mikołaj Bojańczyk. Furthermore, ICALP 2022 edition will be the occasion to celebrate the 50th anniversary of both EATCS and the first ICALP, which was first held in 1972 in Rocquencourt, in the Paris area. I will write more details about the planned activities in the next issue of the Bulletin, but as for now: please pencil these dates in your diary and I hope to see many of you joining us in these celebrations.

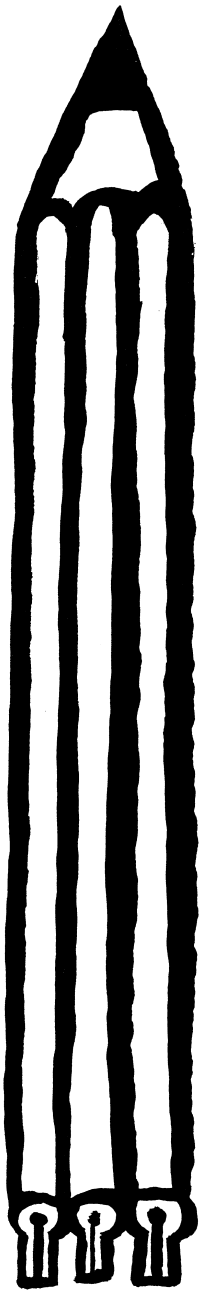
As usual, let me close this letter by reminding you that you are always most welcome to send me your comments,

criticisms and suggestions for improving the impact of the EATCS on the Theoretical Computer Science community at president@eatcs.org. We will consider all your suggestions and criticisms carefully.

I look forward to seeing many of you around, online or maybe even in person (hopefully we will be able to attend conferences and workshops, or to conduct research visits), and to discussing ways of improving the impact of the EATCS within the theoretical computer science community.

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

October 2021



Dear EATCS member,

This issue of the Bulletin of the EATCS comes with several innovations.

First of all, I would like to welcome our new editors: Seth Gilbert, Dennis Komm, Michal Koucky, and Luca Trevisan! I am thrilled to have you on board, and thank you very much for accepting our invitation. Seth will take over the Distributed Computing Column from me. Dennis will join the Educational Column by Juraj Hromkovic as a co-editor. Michal will take over the Computational Complexity Column from Vikraman Arvind, whom I would like to thank very much on this occasion for his great work and contributions as an editor over the last years.

And Luca Trevisan... well, this brings us to our next innovation: in this issue of the Bulletin, we experiment with some new columns. In particular, in order to complement the technical columns, we have four new columns.

The first, is a "Theory Blogs Column" led by Luca Trevisan, which revolves around current trends on the web, including blogs and social media. The idea for such a column was brought up by Sebastian Forster during my recent visit in Salzburg, and I would like to thank Sebastian very much for this suggestion. With the "Viewpoint Column", we introduce a column in which a member of the community shares an opinion or perspective on subjects of interest to the community. Roger Wattenhofer kindly offered to contribute a first such column.

Then, we have an "Interview Column", in which we introduce a "person behind the

papers". Our first interview is with Kurt Mehlhorn, and I highly recommend you to take a look at this column as well. On this occasion, I would like to thank especially Chen Avin, for his great help in creating a set of interview questions; I also thank Klaus-Tycho Foerster and Kathrin Hanauer for feedback on the questions.

Last but not least, we have a "Memorial Column", in which a community member shares some memories from her or his career. The first column is contributed by Gregory Chaitin.

I would like to thank all editors, authors, and other contributors, for their help with this issue. I hope you enjoy some of the new formats, and I welcome any feedback on the issue.

Stefan Schmid, Vienna
October 2021

The EATCS Award 2022

Call for Nominations

Deadline: December 31, 2021

The European Association for Theoretical Computer Science (EATCS) annually honours a respected scientist from our community with the prestigious EATCS Distinguished Achievement Award. The award is given to acknowledge extensive and widely recognized contributions to theoretical computer science over a life long scientific career. For the EATCS Award 2022, candidates may be nominated to the Award Committee consisting of

- Éva Tardos (Chair)
- Johan Håstad and
- Thomas Henzinger

Nominations will be kept strictly confidential. They should include supporting justification and be sent by e-mail to the chair of the EATCS Award Committee:

Éva Tardos
eatcs-award@eatcs.org

Previous recipients of the EATCS Award are:

R.M. Karp (2000)	C. Böhm (2001)	M. Nivat (2002)
G. Rozenberg (2003)	A. Salomaa (2004)	R. Milner (2005)
M. Paterson (2006)	D.S. Scott (2007)	L.G. Valiant (2008)
G. Huet (2009)	K. Mehlhorn (2010)	B. Trakhtenbrot (2011)
M.Y. Vardi (2012)	M.E. Dyer (2013)	G.D. Plotkin (2014)
C. Papadimitriou (2015)	D. Kozen(2016)	É. Tardos(2017)
N. Nisan (2018)	T. Henzinger (2019)	Mihalis Yannakakis(2020)
T. Pitassi (2021)		

The next award will be presented during ICALP 2022 in Paris, France.

The Presburger Award for Young Scientists 2022

Call for Nominations

Deadline: 15 February 2022

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

Nominated scientists can be at most 35 years old on January 1st of the year of the award. Thus, for the 2022 award, the nominee should be born in 1986 or later. Nominations for the Presburger Award can be submitted by any member or group of members of the theoretical computer science community, but not by the nominee themselves nor the advisors for their master's thesis or doctoral dissertation.

The Presburger Award Committee of 2022 consists of

Meena Mahajan (The Institute of Mathematical Sciences, Chennai, Chair), Mikołaj Bojańczyk (University of Warsaw) and Uriel Feige (The Weizmann Institute, Israel).

Nominations, consisting of a two page justification and (links to) the respective papers, as well as additional supporting letters, should be sent by e-mail to:

`presburger-award@eatcs.org`

The subject line of every nomination should start with *Presburger Award 2022*, and the message must be received before **February 15th, 2022**.

The award includes an amount of 1000 Euro and an invitation to ICALP 2022 for a lecture.

Previous Winners:

Mikołaj Bojańczyk, 2010	Patricia Bouyer-Decitre, 2011
Venkatesan Guruswami, 2012	Mihai Pătraşcu, 2012
Erik Demaine, 2013	David Woodruff, 2014
Xi Chen, 2015	Mark Braverman, 2016
Alexandra Silva, 2017	Aleksander Mądry, 2018
Karl Bringmann, 2019	Kasper Green Larsen, 2019
Dmitriy Zhuk, 2020	Shayan Oveis Gharan, 2021

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

EATCS Distinguished Dissertation Award 2021

Call for Nominations

Deadline: 31 December 2021

The EATCS establishes the Distinguished Dissertation Award to promote and recognize outstanding dissertations in the field of Theoretical Computer Science.

Any PhD dissertation in the field of Theoretical Computer Science that has been successfully defended in 2021 is eligible.

Three dissertations will be selected by the committee for year 2021. The dissertations will be evaluated on the basis of originality and potential impact on their respective fields and on Theoretical Computer Science.

Each of the selected dissertations will receive a prize of 1000 Euro. The award receiving dissertations will be published on the EATCS web site, where all the EATCS Distinguished Dissertations will be collected.

The dissertation must be submitted by the author as an attachment to an email message sent to the address `dissertation-award@eatcs.org` with subject `EATCS Distinguished Dissertation Award 2021` by `31 December 2021`. The body of the message must specify:

- Name and email address of the candidate;
- Title of the dissertation;
- Department that has awarded the PhD and denomination of the PhD program;
- Name and email address of the thesis supervisor;
- Date of the successful defense of the thesis.

A five page abstract of the dissertation and a letter by the thesis supervisor certifying that the thesis has been successfully defended must also be included. In addition, the author must include an endorsement letter from the thesis supervisor and can include one more endorsement letter.

The dissertations will be selected by the following committee:

- Susanne Albers
- Nikhil Bansal
- Elvira Mayordomo
- Dale Miller
- Jaroslav Nešetřil
- Damian Niwiński
- David Peleg (chair)
- Vladimiro Sassone
- Alexandra Silva

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

EATCS-Fellows 2022

Call for Nominations

Deadline: December 31, 2021

- VERY IMPORTANT: all nominees and nominators must be EATCS Members
- Proposals for Fellow consideration in 2021 should be submitted by DECEMBER 31st, 2021 by email to the EATCS Secretary (secretary@eatcs.org). The subject line of the email should read "EATCS Fellow Nomination - <surname of candidate>".

The EATCS Fellows Program is established by the Association to recognize outstanding EATCS Members for their scientific achievements in the field of Theoretical Computer Science. The Fellow status is conferred by the EATCS Fellows-Selection Committee upon a person having a track record of intellectual and organizational leadership within the EATCS community. Fellows are expected to be "model citizens" of the TCS community, helping to develop the standing of TCS beyond the frontiers of the community. In order to be considered by the EATCS Fellows-Selection Committee, candidates must be nominated by at least four EATCS Members. Please verify your membership at www.eatcs.org.

The EATCS Fellows-Selection Committee consists of

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

INSTRUCTIONS:

A nomination should consist of details on the items below. It can be co-signed by several EATCS members. Two nomination letters per candidate are recommended. If you are supporting the nomination from within the candidate's field of expertise, it is expected that you will be specific about the individual's technical contributions.
To be considered, nominations for 2021 must be received by **December 31, 2021.**

1. Name of candidate,
Candidate's current affiliation and position,
Candidate's email address, postal address and phone number,
Nominator(s) relationship to the candidate
2. Short summary of candidate's accomplishments (citation – 25 words or less)
3. Candidate's accomplishments: Identify the most important contributions that qualify the candidate for the rank of EATCS Fellow according to the following two categories:
A) Technical achievements

B) Outstanding service to the TCS community
Please limit your comments to at most three pages.
4. Nominator(s):
Name(s)
Affiliation(s), email and postal address(es), phone number(s)
Please note: all nominees and nominators must be EATCS Members

Institutional Sponsors

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

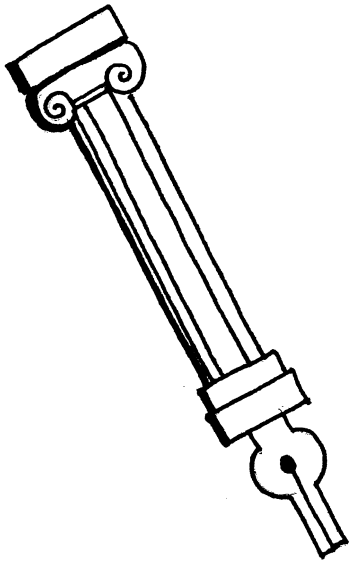
CWI, Centum Wiskunde & Informatica
Amsterdam, The Netherlands

MADALGO, Center for Massive Data Algorithmics
Aarhus, Denmark

Microsoft Research Cambridge
Cambridge, United Kingdom

Springer-Verlag
Heidelberg, Germany

EATCS
Columns



The Interview Column

BY

Chen Avin and Stefan Schmid

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

KNOW THE PERSON BEHIND THE PAPERS

Today: Kurt Mehlhorn

Bio: *Kurt Mehlhorn was a vice president of the Max Planck Society and director of the Max Planck Institute for Computer Science. He graduated from the Technical University of Munichand earned his Ph.D. from Cornell University. Among other, Mehlhorn is known for his contributions to the development of algorithm engineering and he played an important role in the establishment of several research centres for computer science in Germany, including the MPI for Computer Science, the research center for computer science at Dagstuhl and the European Symposium on Algorithms.He won the Gottfried Wilhelm Leibniz Prize, among many other prizes.*

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

Kurt: I share two photos. The first shows me in my office; the picture was taken in 2018. You can see that I enjoy my work, that our institute is located on the border of a forest, and that I am quite effective in producing earth-shaking results. The crumbled papers on the table are testament to this.

The second picture is from 1973. It shows my wife Ena and me at a picnic of the CS department of Cornell University, where I was a graduate student.

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

Kurt: When I started to study computer science in 1968, I had never used a computer. I had even never seen one except on pictures. Programming fascinated me from the beginning; it is an act of creation.

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

Kurt: Let me start with a non-scientific paper. The book Summerhill by A. S. Neill. We (my wife, I, many friends) read it in our twenties and it heavily influenced how we raised our children and how I treat students.

I was a graduate student from 71 to 74. Cook's paper on the NP-completeness of the satisfiability problem appeared in 71 and Karp's paper with the first 21 NP-complete problems appeared in 72. A first draft of the book on Efficient Algorithms by Aho, Hopcroft, and Ullman became available in 73. They were bibles.

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

Kurt: Read the article "Trustworthy Graph Algorithms" by Mohammad Abdulaziz, myself, and Tobias Nipkow (Invited Talk, MFCS 2019). It has a 30-year story behind it. In 1989, Stefan Naeher and I started to develop the library of

efficient data structures and algorithms LEDA. Some of our implementation were incorrect. This experience shaped part of my research agenda until today. The paper discusses what we have learned from this experience and how we have made LEDA a reliable product.

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

Kurt: I am a morning person and try to be on my desk by 8 am. My mornings are reserved for research. I keep my office door closed; I do not accept phone calls and I do not tolerate interruptions. I read, think, and write. After lunch, my office door is open, I interact with students and co-workers, teach and administrate, write reports, and so on. By 5:30, I go home. I sometimes work in the evenings, but never after 10pm.

Of course, the schedule is different when I travel. It was very different when I was vice-president of the Max Planck Society.

My favorite work place is my office, but I can work almost everywhere. Earplugs help a lot.

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

Kurt: Failures are the rule not the exception; see the crumbled paper on the picture. I keep on trying and trying, I discuss the problem with students and colleagues, I put it aside and return to it later, and sometimes I give up on a problem completely. However, I was lucky that most problems that I worked on so far, gave in ultimately.

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

Kurt: What brought me to Saarbruecken? I got my PhD from Cornell in 1974. I wanted to stay in the US and had an offer from Carnegie Mellon, but my wife wanted to return to Germany. Her degree was not valid in the US and she wanted to go back to work. I asked my professors where I should apply. Juris Hartmanis who knew Germany well recommended Bonn, Munich, and Saarbruecken. I applied at all three places and interviewed in the late spring of 1974. My meetings with Rudolf Bayer in Munich and Guenther Hotz in Saarbruecken went well and both offered me orally a position as a researcher. The written offer would follow in a few days. I went back to Cornell, the written offer from Saarbruecken arrived two weeks later, but the written offer from Munich did not come within the next two months. Hence, I accepted the offer from Saarbruecken. I grew up near Munich and we would have decided otherwise, had the offer from Munich arrived in time.

I actually received the offer from Munich after I started my position in Saarbruecken. The letter from Saarbruecken was sent by airmail and took a week to

cross the Atlantic, the offer from Munich was sent by regular mail, crossed the Atlantic on a ship twice (it arrived at our address in Cornell after we had left and then was sent to our new address in Saarbruecken), and hence took months to arrive. Thus, a missing airmail postage stamp had a decisive influence on our lives.

It never occurred to me at the time that I could call Rudolf Bayer to inquire. And email did not exist yet.

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

Kurt: Try to develop depth and breadth. Only depth in a particular area will make you famous, only breadth will allow you to switch topics easily.

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

Kurt: Their intellectual capabilities, their desire to understand, and interests for topics beyond CS.

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

Kurt: Theoretical computer science is striving and new questions and topics arise all the time: fine-grained complexity, the convergence of continuous and discrete optimization, the opening up to questions in economics and learning, to name a few.

When I look back at my career, every 10 years I started to work on topics that I had not known to exist a decade earlier.

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

Kurt: The pandemic did not affect my research much. In fact, I travelled much less and in this way gained additional time for research. I found reasonably efficient ways of interacting with my students and co-workers. What I miss most is the informal interaction over lunch and coffee.

Please complete the following sentences?

- *My favorite movie is...* I do not have one.
- *Being a researcher...* is my dream job. In fact, I do not consider it work. It is pure pleasure.
- *My first research discovery...* turned out to be incorrect.
- *Theoretical computer science in 100 years from now...* will be going strong, although, it will be very different from what it is now.

The Viewpoint Column

BY

Stefan Schmid

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

BEATCS no 135

DO UNIVERSITIES HAVE A FUTURE?

Roger Wattenhofer

1 Introduction

Universities are among the oldest contemporary institutions of humankind. The University of Bologna started teaching non-religious topics in 1088 already, and even the first technical university (The Czech Technical University) is more than 300 years old. So we established that universities certainly have a past – but are they relevant in the future?

The main purpose of universities is debatable. I would argue that fundamental research and higher education are the two core functionalities of a university.[1] However, universities have also been dubbed "knowledge centres" or "competence hierarchies," with research and education being mere side effects.

While universities surely changed a great deal since their beginnings, they are "bipolar" regarding the ongoing digital transformation that is currently shaking up so many aspects of humankind. While a lot of research at universities is often directly studying the digitalization transition, education is generally considered to be one of the least disrupted industries. So, ironically, while we as computer science academics are in the center of the digitalization transition, our own jobs have not changed much.

I started my PhD in 1995. Back then many university and research processes still were "analog." When submitting a paper, we had to send six actual copies of the paper by postal mail[2] to the editor or program committee chair, who then forwarded a copy to each reviewer. Almost every day I headed to the library to find a paper I wanted to read.[3] But already 25 years ago, email started to be used regularly, and then the first web based conference management systems appeared. When I became a professor in 2000, all the obvious university processes were web based already.[4]

One remarkable new development since 2000 are massive open online courses (MOOCs). Coursera's Machine Learning course has had 3 million enrolled stu-

[1] And in this article, I will consequentially focus on education and research.

[2] As today, submissions were often a last minute effort, to the delight of express mail companies.

[3] And like today, often the paper turned out to be less relevant than anticipated.

[4] At Microsoft Research, the library found and scanned any paper you wanted.

dents so far. Yet, universities by and large operate as if MOOCs did not exist. Apart from MOOCs, I am actually surprised how little has changed in the past 20 years. One may claim that one year of Covid-19 has probably affected higher education more than 20 years of digitalization since 2000.

I believe that universities and the academic research environment are in need of change in order to stay relevant. This is the topic of this viewpoint article, first I will talk about education, then about research.

2 Professional Videos

When was the last time you watched a two-hour educational video on Youtube? Assume you want to learn about an unfamiliar subject, and you decide to watch a video.[5] You can choose between a two-hour blackboard lecture by some famous scholar, and a highly animated 5-15 minute video by a random person.[6] What do you choose? Well, I always go for the short video. The short video usually feels like time well spent. So why are universities still following a rigid two-hour lecture schedule?

Sure, actual university lectures are about interaction more than presentation. When I teach, I ask a motivating question right off the bat, and the whole lecture is really a discussion rather than a presentation. The students speak almost as much as I do. However, Covid put a halt to these discussions. I still try to have interactions, but in a video call interactions do not feel natural. So I produced these 90-minute-long videos instead, being incredibly unhappy about them. Since Covid, students watch a lot of these videos, and many students watch these lecture videos at twice the speed. Indeed, a student told to me that he started watching not only lectures at twice the speed, but also movies; now he is so used to watching everything faster, regular conversations feel slow and boring to him.[7]

I also wonder whether academics like us stand a chance against professional video producers. Youtube is full of excellent educational video channels. Here are some of those that I subscribe in alphabetical order: 3Blue1Brown, CGP Grey, Computerphile, Finematics, Kurzgesagt, Map Men, Numberphile, Ordinary Things, Veritasium, Whatifalthist.[8]

While half of these channels are single person hobbyists, the production efforts of the best channels is out of reach for an academic. The most professional of these educational channels is probably Kurzgesagt. So far Kurzgesagt predominantly produces animated sci-fi and life science videos. However, eventually

[5] Personally, I almost always prefer reading to watching a video.

[6] Random as in no obvious academic merits.

[7] At this stage, this is affecting his social life.

[8] Okay, admittedly, many of these are heavily on the entertainment side of education.

there will be a channel producing short videos that explain the fundamental topics in Computer Science in a crisp and intuitive way. These videos will be more appealing than any of my blackboard gymnastics. As a consequence, many students will prefer to watch these professional videos. In order to stay relevant, we need to integrate these videos into our lectures, maybe focusing more on Q&A rather than explaining the basics.[9]

TLDR: Professional 5-15 minute videos will replace 45 minute lectures.

3 MOOC vs. University

Lectures are the core element of education at a university. If lectures are being outsourced to professional videos, universities lose a significant part of their credibility. But lectures come with theoretical and practical exercises, textbooks or scripts,[10] and labs and projects. What about these?

I believe that winning textbooks should be written collaboratively, by a team of academics and illustrators. The cookbook Modernist Cuisine by Nathan Myhrvold was a marvellous example for such a collaboration.[11] Exercises could also be shared between universities. MOOCs are leading the way here.

What about that famous "university spirit"? How do students and staff at a university interact with each other? Do we have an environment where students can be creative? Do students and staff engage in critical thinking? Does the university have a lively discussion culture? Most MOOCs of course have discussion forums, and the best are incredibly lively. I would claim that MOOCs can replicate a lot of this fabulous university spirit[12] by setting up local hubs if there is a large enough audience attending the same MOOC.

Maybe another point to differentiate universities are exams. Some universities simply ask knowledge questions at exams. Excellent universities however ask demanding questions where students must apply the mindset of the lecture, even to a completely fresh model. However, even exams can be standardized, maybe there will be simpler and harder exams for the same kind of content.

But there is hope for universities: There are the advanced classes, for instance on the Master level. This is content that will not be replaced by professional videos anytime soon, since the market for such content is simply too narrow. If a university has professors that are leading in their research area, this will be an advantage for students. So is the future of the university a postgraduate-only

[9] I tried video plus flipped classroom in one of my lectures. The experience was mixed.
[10] To fully understand a topic, a combination of static text/graphics usually beats any video.
[11] For a more recent example, I recommend Philipp Dettmer's book Immune.
[12] Well, Covid kind of killed a great deal of that spirit; let's hope it comes back.

institution a la Weizmann Institute? Ultimately this question will be answered by people like ourselves in the form of this simple question: Would you rather hire a MOOC graduate with a good selection of relevant courses, or one from a foreign university you barely know (maybe you are even unfamiliar with the grading scheme at that university)?[13]

TLDR: MOOCs will do all undergraduate education; universities will be postgrad level only.

4 Organization of Research

Research is remarkably unorganized and chaotic. This may be a sign of our times where Google taught us that they will organize any data for us. On the other hand, Wikipedia also proved that organized data still has a lot of merit. Wikipedia also includes scientific results, however, Wikipedia is quite restrictive regarding content. As a consequence most academic papers are not mentioned in Wikipedia, since these are only relevant to a small audience. I want a Wikipedia for science. Such Science-Wikipedias already exist in certain areas. For instance, there is the Compendium of NP Optimization Problems by Pierluigi Crescenzi and Viggo Kann. The Compendium has not been updated in 20 years, yet it still provides valuable information on various types of approximation algorithms and non-approximability results.[14] If I want to quickly learn about the state of an algorithmic problem, the Compendium is a still a good alternative to "just googling."

We should have such a Science-Wiki. Unlike Wikipedia, this Science-Wiki should not be restrictive, and allow for including a summary of any (published) scientific result.[15] Everybody should be able to add and update information. The Science-Wiki should be highly hierarchical, such that more obscure topics are hidden away in sub-sub-pages. Of course the Science-Wiki would face some of the same problems (wrong information, edit wars, etc.) as Wikipedia. But since Wikipedia is able to contain these problems, we should as well. Being responsible for a domain will come with the same level of recognition and appreciation as being editor in a prestigious journal. The closest thing we have to such a Science-Wiki are some research blogs, where established scientists like Scott Aaronson discuss new results and provide background information about their favorite subject.

[13] Greek theses grades for instance are downright malicious.

[14] Some information is clearly outdated, but weirdly enough still relevant.

[15] I believe that any relevant scientific result can be represented by a simple tagline.

But maybe such organization attempts are outdated.[16][17] Alternatively, machine learning might come to the rescue. The best machine learning language models are now at a level where they might help with literature search. We could have an engine that has access to ArXiv and other repositories, and is able to answer refined text queries like "Did anybody ever improve that result in Y?", "Is there a special case for which Z can be solved?", or even "What are the open problems in area X?"

TLDR: We must organize scientific knowledge with a Science-Wiki – or with machine learning.

5 Conferences and Journals

Conferences came to a full stop during Covid, and we learned a great deal about conferences during this time. We must question whether flying half around the globe several times a year makes sense in light of the progress we made regarding video conferencing tools. Instead we should consider the model of mathematicians that meet in bigger but less frequent events once a year. Oded Goldreich suggested such "festivals" many years ago,[18] and now we have a unique opportunity to implement them.

During Covid, I attended a bunch of virtual conferences. Some were a mere bad copy of the original conference: just linear talks, sometimes in the middle of the night because of an unfortunate time zone scheduling. Other conferences fully embraced being virtual and deviated from the classic one-talk-after-another scheme. They were free of charge, or at least almost-free if you did not present a paper. Some were remarkably inclusive, and simply uploaded all presentation videos to a Youtube channel, free for the whole world to watch. In this case, I started watching every single presentation. I also ended most of them prematurely, often already after only one minute. This may sound terrible, but is actually a blessing compared to the linear organization of the usual conferences where I have to sit through a presentation even though I am not interested anymore. In an old school conference I usually fall into this pattern: I follow a talk, but sometimes I get bored because I do not find the topic relevant, and then I get distracted by the glorious internet.[19] In the worst case, I might even miss the beginning of the next talk! I also asked some colleagues and students about good talks that they attended, and then watched (the beginning of) these videos.

[16] Some young colleagues probably ignore all proceedings, and simply google when needed.
[17] And in a large enough area such as machine learning, googling might be the only way.
[18] Go and check his essays and opinions on his web page if you have never done so.
[19] Back then conference organizers often turned off Wifi during talks. Very annoying.

Festivals could be different. Video presentations should be available before the festival. The actual festival would be all about connecting the dots, and discussing the results. Not having an audience that must sit through an entire session should also have the additional advantage of improved presentations. Speakers know that they must captivate the audience to continue watching, and they will be more careful with their presentations.

Some conferences might choose a fully virtual future, with or without Covid. Lots of traveling might have made sense in the early days of computer science – to establish the field. But now less travel is better, for the environment but also for ourselves.

An already established trend is to merge conferences and journals. Many conferences started behaving more and more like journals, with multiple deadlines per year, and elaborate discussion rounds. In some communities, reviewing is strictly better at conferences than journals. Some practical conferences send so detailed reviews that all reviews back to back are longer than the actual submitted paper. Machine learning conferences introduced open reviewing systems where everybody could potentially analyze a submission. These are fascinating developments.

Back when I was a student, professors used to tell me that conference publications are not to be trusted: Only journal publication are correct, because they only these are carefully reviewed. Well, journal reviewing is generally not getting better. But more importantly, many publications are probably just not relevant enough, so strangely enough it does not even matter whether their results are correct. If a publication becomes relevant, the author of the textbook who writes about the result and the people citing the result will usually check again in great detail. I would argue that a lot more mistakes in research were caught by interested readers instead of assigned (journal) reviewers. So distinguishing between conferences and journals makes less and less sense. We should simply make all conference journals, and journals conferences, or even better: both festivals.

TLDR: Conferences + Journals $\rightarrow$ Festivals.

The Theory Blogs Column

by

Luca Trevisan

Bocconi University
Via Sarfatti 25, 20136 Milano, Italy
L.Trevisan@UniBocconi.it
https://lucatrevisan.github.io

BEATCS no 135

INTRODUCING THE THEORY BLOGS COLUMN

Luca Trevisan
Bocconi University
lucatrevisan.github.io

Blogs are a format for sharing content online. In a blog, one writes essay-like posts that are shared on the internet and are, typically, freely readable by everybody. The posts are seen in reverse chronological order, and readers can add their comments at the bottom of each post, respond to other comments, and so on. This format, introduced in the late 1990s, quickly became popular in the first decade of the 2000s for a variety of purposes.

Many people used blogs to share diary-like entries (this use peaked around 2010 and then migrated to social media apps like Facebook and Instagram). In addition, enthusiasts and hobbyists used blogging platforms to write in depth about their passions (a type of writing that we now see on Reddit) and professional writers, such as journalists and commentators, used it to share unedited and informal takes on timely issue (this type of writing migrated, in a very different form, to Twitter, and now it is seeing a resurgence in the original long form via newsletters).

Blogs also became popular in academia, including, so that we come to the point of this column, among theoretical computer scientists. Academic blogs, particularly in highly mathematical fields like ours, play a role that has not been subsumed by newer social media apps, and, as far as I can tell, they remain popular.

They allow us to share widely and freely the kind of discussions and ideas that are crucial to the development of our field and that are not fit to be published and disseminated as widely in other formats. They allow the sharing of newly formulated open questions, to discuss informally what is the fundamental new idea of a newly announced technical breakthrough, they allow us to present new and simplified proofs of old results, they allow the discussion and the analysis and quick refutation of a new paper claiming $P \neq NP$, and so on. They also allow the author to share (and collect comments on) course lecture notes, which has been done for several courses, and has even led to published textbooks, such as Ryan O'Donnell's book on analysis of Boolean functions.

This kind of content (insights into new techniques, simplified proofs, new open questions, drafts of lecture notes) has been shared, for a long time, in hallway con-

versations in departments and in conferences, and in private conversations among experts, or in certain graduate seminars. Having these discussion on blogs, however, makes them accessible and open in a completely different way, so that even students and young researchers that do not have access to those hallway conversations and those graduate seminars can share in the "lore" and the "oral tradition" of our field. Anecdotally, I have heard of a number of young theoretical computer scientists that have been learning about our field from blogs since the time they were undergraduates, or even high school students.

I have been thinking a lot about this issue of access to insights and ideas over the past year and a half. Because of Covid-related restrictions, we did not have conferences to go to and have hallway conversations at, and many of us did not even have departments to go to. Most exchanges of ideas happened in scheduled video calls, but one cannot bump into a colleague in a video call and start chatting, or overhear someone else's video call and join the conversation. The Covid restrictions have increased the "privilege," to use a currently fashionable word, of those plugged in to certain academic networks, and they have made it harder for new students to access knowledge, so that the openness of academic blogs is a valuable resource.

Blog posts are also a medium to tell personal stories that relate to research, something that has no place in technical papers and in textbooks, but is of great value. Omer Reingold, for example, has invited a number of people to write on his blog about "research life stories," a project that is, to me, of great interest to see what happens behind the scenes of great research and how other people think about what they do. There is significant concern in our community about our gender imbalance and other issues of representation and inclusion, and significant pressure coming from university administrations to do better. University administrations address this problem with the tools that they have and that they know how to use: top-down mandates and emphasis on process. There are several things that our community can do to change in a bottom-up way, and one thing that I would like to see more of is personal narratives of what it's like to devote one's life to research and to embrace an academic community in which you do not see "people like you," whatever this means for that particular person. This is something that can happen with a variety of media, including academic blogs, and I have personally seen the impact of sharing personal stories concerning LGBT issues.

But enough about theoretical computer science blogs, what does the Bulletin of the EATCS have to do with all this? This is a very good question, as people say during a Q&A when they do not have a good answer. I was asked whether I would like to edit a BEATCS column on theoretical computer science blogs: this sounded like an open ended task with no clearly defined goals, that gave me a good chance to make a fool of myself. So, as I have always done with similar proposals, I said yes.

My plan is to provide BEATCS readers with an access to TCS blog content that adds something that could not be provided by simply linking to posts, and to provide something that is complementary to blog content. For example, blogs thrive in immediacy, and I plan to use some columns to take a broader, retrospective view, and blogs tend to have a distinct, individual voice while this column will feature a variety of points of views and voices. Here is another difference: when a blogger runs out of things to say, the pace of publishing slows down and eventually there is an indefinite hiatus. The regular publication pace of BEATCS will be such that when we run out of interesting things to say to our readers, we will close down this column, perhaps to be revived again in the future.

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 135

Sorting Short Integers: The Exposition

Michal Koucký*
Computer Science Institute
Charles University, Prague
koucky@iuuk.mff.cuni.cz

Abstract

This expository article reviews recent and some not so recent results on sorting integers in various models of computation: from word RAM to Boolean circuits, with the main focus on the latter. We hope that even seasoned researcher will find our perspective refreshing.

1 Sorting using comparisons

Sorting is one of the most ubiquitous and versatile algorithmic primitives. It is taught in introductory computer science courses around the globe. The classical algorithms such as Heapsort and Quicksort take time $O(n \log n)$ [13]. They are comparison based algorithms which means that they rely only on operations that move the input items around and compare two individual items which one is larger. There is a well known $\Omega(n \log n)$ lower bound on the number of comparisons needed by any comparison based algorithm to sort n *distinct* items. The lower bound is derived from the number of different permutations of the items.

It is often overlooked that the lower bound is merely linear in the bit-size of the input. Indeed, any *reasonable* binary encoding of a sequence of n distinct items will use $\Omega(n \log n)$ bits. So in terms of the bit-size of the input the lower bound is linear. This fact can be exemplified in the following scenario which seemingly breaks the lower bound: sorting n items from a domain K of size $2^k < n$. Using ordinary balanced search trees of depth $O(k)$ we can sort such a sequence using $O(nk)$ comparisons. (Each node in the tree contains a list of items of the same value.) For $k \in o(\log n)$, this gives sorting using $o(n \log n)$ comparisons. One can show using the standard argument that $\Omega(nk)$ comparisons are needed. The bit-size of the input is nk, so in this case sorting is linear in the bit-size of the input.

*Partially supported by the Grant Agency of the Czech Republic under the grant agreement no. 19-27871X.

In a more general setting we have n items $x_1, x_2, \ldots, x_n$, each item x_i encoded using w_i bits by some prefix-free code. Encoding of the whole input sequence gives input of bit-size $m = w_1 + w_2 \cdots w_n$. We claim that the sequence can be sorted using $O(m)$ comparisons. If there are K distinct values in the input sequence, each appearing $q_1, q_2, \ldots, q_K$ times, sorting the sequence using Splay trees leads to $O(K + \sum_{j=1}^{K} q_j \log(n/q_j))$ comparisons [34]. It follows from standard information theoretic arguments on the entropy of the sequence that $m \geq \sum_{j=1}^{K} q_j \log(n/q_j)$ [14].

The use of a prefix-free encoding is natural. If the items are encoded individually by a code that can be uniquely decoded then there is a prefix-free encoding of the same efficiency [14]. For example a fixed-size encoding is prefix-free but some sequences might be encoded more efficiently: Consider a sequence consisting of $n - \sqrt{n}$ copies of 0 together with one copy of each number from 1 to $\sqrt{n}$ in arbitrary order. A fixed-size encoding would use $O(n \log n)$ bits but the sequence can be encoded by a prefix-free code using $O(n)$ bits in total, and we can sort the sequence using $O(n)$ comparisons.

Hence, sorting arbitrary sequence of n items can be done using a number of comparisons that is *linear in the bit-size* of the encoding of the sequence. This matches our intuition well — each comparison can extract one bit of entropy from the input and there are at most m bits of entropy to extract. Bit-size of the input will become an important measure later when we consider Boolean circuits.

1.1 Sorting networks

A special class of comparison based sorting algorithms is represented by sorting networks. A *sorting network* is a collection of n horizontal wires which carry the input values from left to right. At various places, two wires can be connected by a vertical *comparator* which switches the values carried along the wires so that the larger value continues on the lower wire and the smaller value on the upper wire. The comparators should be organized so that the values leave the network sorted top to bottom. See Fig. 1 for an example.

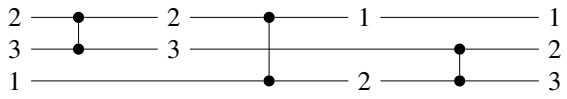


Figure 1: An example of a sorting network with three inputs (the horizontal lines), and three comparators (the vertical lines) [24].

The complexity of the network is the total number of comparators and the depth of the network is the maximum number of comparators on any path from left to right in the network. Comparator networks have been extensively studied

and are used in various practical applications. Most notable constructions are the well-known construction of Batcher [8] of size $O(n \log^2 n)$ and depth $O(\log^2 n)$, and the famous construction of Ajtai, Komlós and Szemerédi [6] of size $O(n \log n)$ and depth $O(\log n)$. Up-to a constant factor the network of Ajtai, Komlós and Szemerédi (*AKS network*) is clearly optimal for sorting n distinct items by the above lower bound. However, it is also optimal for sorting input sequences consisting of only zeros and ones, as shown by the zero-one principle for sorting networks [25]: Any sorting network correctly sorting inputs of zeros and ones correctly sorts arbitrary inputs. Hence sorting items using sorting networks might be less efficient than sorting them by ordinary comparison based sorting algorithms.

Sorting networks are an example of an *oblivious* algorithm in which the sequence of performed operations does not depend on the actual input. Later we will see another example — Boolean circuits.

1.2 Beyond comparison

Beyond comparison based algorithms there is the well known Radixsort. This is an example of an algorithm that runs on the *word RAM*. In the word RAM model the memory is organized into cells of w bits each ($w \geq \log n$). At each time step the program executes some operation with a fixed number of memory cells. Which memory cells are involved in the operation can be determined by the content of special *address* cells of the machine. Usually we place a restriction on how complex each operation can be. Known algorithms usually involve relatively simple operations such as bit-wise Boolean operations, integer addition, multiplication, perhaps division. The input to the machine are n integers each of w bits presented in the first n memory cells. The output is expected to be there as well. When $w \in O(\log n)$, Radixsort can sort n integers in time $O(n)$, i.e., using $O(n)$ operations.

For large w, where $w \in \Omega(\log^2 n \log\log n)$, there are also algorithms for word RAM that run in time $O(n)$ [4, 9]. These algorithms are based on word level parallelism. They compress multiple input values into a single memory cell, and on the compressed input they run a comparison based sorting implemented in parallel. The parallel execution of the sorting algorithm provides the necessary speed-up to obtain a linear-time word RAM algorithm. The compression is usually quite involved and proceeds in several stages as the original input values have w bits.

For w in the range from $\omega(\log n)$ to $o(\log^2 n \log\log n)$ the fastest known algorithm of Han and Thorup [22] is randomized and operates in expected time $O(n\sqrt{\log\log n})$ or more precisely in time $O(n\sqrt{\log n \frac{w}{\log n}})$. The algorithm is a clever combination of several techniques used for sorting algorithms running in time $O(n \log\log n)$ [4, 18].

The first algorithm running in $O(n \log\log n)$ time was given by Andersson et

al. [4]. It is obtained by iterating a linear time reduction of sorting n items of w bits each to sorting n items of $w/2$ bits each [4, 26]. The iteration stops when the integers reach bit-size $O(\log n)$ and we can sort them in linear time using e.g. Radixsort. The reduction splits the items into buckets based on their first $w/2$ most significant bits, and then sorts each bucket based on the $w/2$ least significant bits. The sorting of all the buckets can be done together by annotating each item by its bucket index.

Han [17, 18] developed a different approach and gave a linear time splitting procedure which takes n items of w bits and partitions them into sets $X_1, X_2, \ldots, X_t$ each of size at most $\sqrt{n}$ so that for any $i < j$, items in the set X_i are less or equal to the items in X_j. This procedure can be iterated for $O(\log \log n)$ steps to sort the integers completely. Han's splitting technique is a clever traversal of the trie built from the binary representations of the integers. The traversal is top to bottom and it iteratively splits the formed buckets into smaller ones. The splitting is done using hashing of prefixes into hash values of varying sizes depending on the entropy of the emerging buckets. The splitting is done for many items in parallel using word level parallelism which provides enough speed-up to finish the whole iterative process in linear time.

If there were a technique that would split the integers into sets $X_1, X_2, \ldots, X_t$ of size $2^{\log^{1-\epsilon} n}$ for some $\epsilon > 0$, one could sort in linear time.

Another algorithm running in time $O(n \log \log n)$ is presented in [27]. This algorithm selects a random subset of the input items of size $O(n/\log n)$ and sorts them. For each input item it then finds the proper interval in the sorted sample where the item belongs to. Finding the interval can be done using binary search on the longest common prefix shared with any of the sampled items in time $O(\log \log n)$. This creates small buckets corresponding to different intervals of the sample. Sorting each bucket gives a sorting algorithm running in time $O(n \log \log n)$.

1.3 Turing machines

Perhaps the most fundamental model of computation beside word RAM are Turing machines. Sorting n integers of w bits each can be done using comparison based sorting algorithm such as Quicksort on two-tape Turing machine in time $O(nw \log n)$. In the same time one can also implement Mergesort on three-tape Turing machine. On a single-tape Turing machine one cannot sort faster than $\Omega(n^2/\log^2 n)$ as otherwise one would break the quadratic lower bound on recognizing palindromes by one-tape Turing machines of Hennie [20].

For $w \leq \log n$, one can use binary Radixsort to sort integers on two-tape Turing machines in time $O(nw^2)$. (Splitting a list of integers into two based on a particular bit can be done by passing over the list twice.) It is believed that sorting on two-tape Turing machines requires time $\omega(nw)$. For restricted classes of algorithms this was

proven by Stoss [35] and Paul [31].

2 Sorting by Boolean Circuits

Our main focus for the rest of the article is sorting using Boolean circuits. A *Boolean circuit* is a directed acyclic graph in which each node (*gate*) has in-degree at most two. Each node of in-degree zero is an *input gate* and it is labeled by one of the input bits $y_1, \ldots, y_m$. Each node of in-degree one is labeled by a Boolean negation, and each node of in-degree two is labeled either by AND or OR. (The *in-degree* of a node is the number of its incoming edges and the *out-degree* is the number of its out-going edges.) On input $y \in \{0,1\}^m$ the circuit is evaluated by assigning values to gates and edges as follows: each edge of the circuit receives the value of its starting node, each input gate labeled by an input bit y_i is given the value of y_i, and each node labeled by a Boolean function g is assigned the value of g applied on the values of edges incoming to the node. The output of the circuit is given by the values of designated nodes.

The size of the circuit is the number of its gates and its depth is the length of the longest path from an input gate to some output gate. We want to minimize circuit size and depth. For more background on circuits see e.g. [23].

We will be interested in designing Boolean circuits computing the sorting function $\mathrm{SORT}_{n,w} : \{0,1\}^{nw} \to \{0,1\}^{nw}$ which takes as its input n integers, each encoded in binary using w bits, and outputs the same set of integers but sorted according to the numerical order. A variant of this problem is a partial sorting function according to k bits $\mathrm{SORT}_{n,w,k} : \{0,1\}^{nw} \to \{0,1\}^{nw}$ which takes the same input as $\mathrm{SORT}_{n,w}$ but outputs the set sorted according to the first k bits of the integers. Integers with the same value of the first k bits can appear in arbitrary order. Alternatively, one can think of a partial sort as sorting (key, value) pairs according to k-bit keys with values being $w-k$ bits long. Clearly, $\mathrm{SORT}_{n,w,w} = \mathrm{SORT}_{n,w}$. Our goal is to design circuits computing those functions. *Sorting circuits* will refer to such circuits.

Circuits are an oblivious model of computation as the sequence of performed operations does not depend on the actual input. One can build a sorting circuit from a sorting network by implementing each comparator by a small Boolean circuit. Indeed, it is fairly easy to build a circuit of size $O(w)$ and depth $O(\log w)$ that compares two w-bit integers and outputs them in a sorted order. Replacing each comparator in the AKS sorting network by a copy of such a circuit and connecting the wires appropriately one can get a circuit of size $O(nw \log n)$ and depth $O(\log n \log w)$ that sorts n integers of w bits each. We will refer to this circuit the *AKS sorting circuit*.

It was raised by Asharov et al. [7] whether one can design smaller circuits

when w is small compared to $\log n$. Because of the zero-one law for sorting networks this will necessarily require a different technique than used by sorting networks. For every $\varepsilon > 0$, Asharov et al. [7] give a construction of circuits of size $O(nw^2(1 + \log^* n - \log^* w)^{2+\varepsilon})$ and polynomial depth sorting n integers of w bits each.

Their construction relies on a circuit for $\mathrm{SORT}_{n,w,1}$ which partially sorts the input according to a single bit. Using such circuits one can sort the integers completely bit by bit starting from the most significant bit. This recursive approach requires certain care. We describe next a different approach which gives a circuit of size $O(nw^2)$ and depth $O(\log n + w \log w)$.

2.1 Fast counting

Here we describe an approach to sorting based on Counting Sort from [24] which first counts the number of occurrences (*frequency*) of each possible integer value and then reconstructs the sorted integer sequence corresponding to the frequencies. This will give a sorting circuit of size $O(nw^2)$ and depth $O(\log n + w \log w)$.

Slow Counting. Let $W = 2^w$, where $W < n^{1/20}$. For a given value $y \in \{0, \ldots, W-1\}$ we can determine the frequency of y among $x_1, \ldots, x_n$ using a circuit of size $O(nw)$ and of depth $O(\log n + \log w)$. This is done by comparing y with each x_i for equality, and then summing up the resulting indicator vector. Comparing two w-bit strings can be done using a circuit of size $O(w)$ and of depth $O(\log w)$. Summing up n bits can be done using a circuit of size $O(n)$ and of depth $O(\log n)$. Hence we can calculate the frequency of all the values from 0 to $W - 1$ using a circuit of size $O(Wnw)$ and of depth $O(\log n + \log w)$. However, such a circuit is too big so we proceed differently.

Fast Counting. We divide the integer sequence into blocks of size W^8, and we sort each of the blocks by the AKS sorting circuit of size $O(W^8 w \log W^8)$ and of depth $O(\log W^8 \log w)$. Hence the total size of this stage of the algorithm is $O((n/W^8) \cdot W^8 w \log W^8) = O(nw^2)$ and the depth is $O(w \log w)$. This fits within our budget.

Now we partition each block into *parts* of W^4 integers. There are only W distinct integer values so except for W parts, integers in each part are the same. Any such part is called *monochromatic*. Since each part is sorted we can easily check whether it is monochromatic by comparing its first and last item.

At this stage it is also relatively inexpensive to compute the frequency of all the items in monochromatic parts: For each value, compute the number of monochromatic parts containing it, and multiply the number by W^4. Multiplication by W^4 only requires to pad each count by $4w$ zeros on the right. This will use a circuit of total size $O((n/W^4) \cdot W \cdot w) \subseteq O(n)$ and depth $O(\log(n/W^4)) \subseteq O(\log n)$ which is negligible for us.

It remains to count the number of items of each value in non-monochromatic parts. In each block we replace all the values in monochromatic parts by 0 and we sort each block again using the AKS sorting circuit. Except for the value 0, we can count the frequency of each value in non-monochromatic parts by counting its frequency in the last W^5 items in all the blocks. This can be done using a circuit of size $O((n/W^8) \cdot W^5 \cdot wW) \subseteq O(n)$ and of depth $O(\log n)$. To properly count the frequency of the value 0, we count zeros only in the first non-monochromatic part of each block. Those zeros correspond to the original zeros in the first non-monochromatic part of each originally sorted block. Counting those zeros can be done easily using the same technique as for the other values. Hence, it does not increase the circuit size beyond our budget.

At this point we have two frequency vectors, one counting all the values in monochromatic parts and one counting them in non-monochromatic parts. We can add them point-wise to obtain a vector of overall frequencies. This requires a circuit of a negligible size and depth.

Decompression. Decompression of the frequency vector can be done in a mirror fashion. We decompose the output positions into blocks of size n/W^8. There will be W^8 blocks and at most W of them should become non-monochromatic. For each output block we can determine whether it will be monochromatic and if so what value it will contain. Given that W^8 is substantially smaller than n this can be done by a circuit of small size and depth.

To recreate the non-monochromatic blocks we subtract from the frequencies the counts of items in monochromatic parts, and we use a naïve circuit of size $O((n/W^7) \cdot 2^w \cdot \text{poly}(w))$ to recreate a sequence of (n/W^7) integers corresponding to those frequencies. We split this sequence into blocks of size (n/W^8) and shuffle them with the monochromatic blocks in correct order. This last step is done using the AKS sorting circuit which sorts according to w bits but drags along each integer a string consisting of (n/W^8) integers. This last step requires a circuit of size $O(W^8 \cdot (nw/W^8) \log W^8)$ and depth $O(\log W^8 \log w)$.

Hence, we obtain a circuit of size $O(nw^2)$ and depth $O(\log n + w \log w)$ for sorting n integers w bits each. Lin and Shi [29] give an incomparable result discussed in the next section.

3 Partial Sorting

In this section we will focus on partial sorting of w-bit integers according to k most significant bits. This corresponds to sorting (key, value) pairs where key has k bits and value has $w - k$ bits. Here, w can be large. Comparison based sorting on such an input would use $O(nk)$ comparisons as seen in the first section. Hence, one could hope for partially sorting circuits of size $O(nwk)$. When $k \in \Omega(w)$ the

result of previous section already provides a circuit of size $O(nw^2)$. For small k, one wants smaller circuits.

For arbitrary constant $\varepsilon > 0$, Asharov et al. [7] give a construction of circuits of size $O(nwk(1 + \log^* n - \log^* w)^{2+\varepsilon})$ sorting n integers of w bits each according to their first k bits. First, they design a circuit of size $O(nw(1 + \log^* n - \log^* w)^{2+\varepsilon})$ that sorts the integers according to a single bit. Then they use the circuit to sort the integers by successive bits from the most significant to the least significant. Koucký and Král [24] use the same strategy to first build a circuit for sorting according to one bit (see the next section) and then they apply it iteratively similarly to [7]. This gives a circuit of size $O(nwk(1 + \log^* n - \log^* w))$ and depth $O(\log^3 n)$ that sorts w-bit integers according to their first k bits. In a subsequent work, Lin and Shi [29] get circuits of depth $O(\log n + \log k)$ and size $O(nkw \cdot \text{poly}(\log^* n - \log^* w))$ for $k \in O(\log n)$. Lin and Shi use for their construction the sorting strategy of Ajtai, Komlós and Szemerédi [6] along with the techniques of Asharov et al. [7].

3.1 Sorting according to one bit

In this section we will look at sorting w-bit integers according to a single bit. Sorting a sequence of w-bit integers according to one bit corresponds to the problem of moving designated set of input integers to the beginning or end of the sequence. The set is indicated by the bit according to which we sort. This problem is closely related to routing in graphs and in particular, to routing in superconcentrators.

A *superconcentrator* is a directed acyclic graph with n input nodes and n output nodes that satisfies the property: For any pair of subsets of A and B of input nodes and output nodes, respectively, of the same size ℓ, there are ℓ vertex disjoint paths from vertices in A to vertices of B. Such a superconcentrator can serve as a skeleton of a circuit sorting according to one bit as it can move designated items to their desired positions.

Aho, Hopcroft and Ullman [5] were among the first to observe the connection between routing and various algorithmic problems, and they defined the superconcentrators. Furst, Chandra and Lipton [11] have shown that many natural functions such as addition of two n bit integers require Boolean circuits to have some weak superconcentrator property. It follows from a simple information theoretic argument that the same must be true for sorting circuits. The original motivation to study superconcentrators was to prove lower bounds on their size (number of edges), and hence derive non-linear lower bounds on the size of Boolean circuits for specific functions. The hope was that the superconcentrator property requires graphs to have super-linear number of edges. This turns out not to be the case as shown by Valiant [36]. Indeed, there are superconcentrators with $O(n)$ edges which allow to route any set of items placed on selected inputs to any selected set of outputs (of the same size) along non-intersecting vertex disjoint paths [32, 36]. Moreover those

graphs have in-degree and out-degree bounded by a constant. Known constructions of superconcentrators rely heavily on expander graphs [21].

One could use any off-the-shelf superconcentrator to solve the sorting problem according to one bit by turning the superconcentrator into a Boolean circuit: One could replace each node of the superconcentrator by a *selector* circuit which would select from among the values coming into the node the one which comes along the edge of a routed path, and propagate the value further. Using a linear size superconcentrator and linear size selector circuits would give a circuit of size $O(nw)$. The only issue is who will tell the selector which edge is active so which value should be propagated.

Pippenger [33] resolved this routing issue in a rather efficient way. He gives a construction of linear size superconcentrator together with an efficient algorithm that determines the routing. The algorithm can be implemented by a small size circuit namely, there is a circuit of size $O(n \log n)$ and depth $O(\log^2 n)$ that gets as its input indicator vectors of sets A and B and outputs a vector indicating which edges should be used to connect A to B by $|A|$-many vertex disjoint paths in the accompanying superconcentrator. Put together, this gives a Boolean circuit of size $O(nw + n \log n)$ and depth $O(\log^2 n)$ that sorts w-bit integers according to one bit. We call this *Pippenger's partially sorting circuit*.

To get a smaller circuit Asharov et al. [7] open up the construction of Pippenger [33] and use his technique to build a smaller circuit from scratch. We will use Pippenger's partially sorting circuit as a black-box to build more efficient circuits.

The cost of Pippenger's partially sorting circuit is dominated by the size of the circuitry to calculate the routing. This takes a circuit of size $O(n \log n)$ but we can afford only $O(nw)$. So we use a similar technique as in Section 2.1 and we will apply this circuit only to blocks of inputs of $2^{O(w)}$ items.

Say we divide the input into blocks of integers of size $n' = 2^{2w}$, and we sort each block by Pippenger's partially sorting circuit. This will give a circuit of total size $O((n/n') \cdot [n'w + n' \log n']) = O(nw)$. Now we partition each block into parts of 2^w integers. All but one part in each block will be monochromatic meaning that each part contains items with the same value of the bit according to which we sort. Now we could try to use a similar technique as in Section 2.1 and apply some naïve algorithm on monochromatic and non-monochromatic parts. Unfortunately, that does not work here as even the items in monochromatic parts may vary. So we will proceed iteratively from here.

The iterative procedure will use parameters $W_0, W_1, \ldots, W_{\log^* n - \log^* w - 1}$ where $W_0 = w$ and $W_{i+1} = 2^{W_i}$. At iteration $i > 0$, we divide the current sequence into parts consisting of W_i items, and we form blocks of $2^{2W_i}/W_i$ parts. Hence, each block contains 2^{2W_i} integers. Within each block there will be at most $2 \cdot 2^{2W_i}/W_i^2$ non-monochromatic parts. In each block we will move the non-monochromatic parts to a side using Pippenger's partially sorting circuit that will consider each

part as an item consisting of $W \cdot w$ bits. Then in each block we sort the items in non-monochromatic parts together using Pippenger's partially sorting circuit. The number of those items is at most $2 \cdot 2^{2W_i}/W_i$. We repartition those items into parts of size W_i, and sort all the parts (again as units) within the block using Pippenger's partially sorting circuit according to the appropriate bit of their first item. (We do not care where the single non-monochromatic part ends up.) Notice, if we repartition the block into parts of size $2^{W_i} = W_{i+1}$ then at most two parts in the block out of W_{i+1} parts will be non-monochromatic. At this point we are ready for next iteration $i + 1$.

After the last iteration the total number of parts will be relatively small and the number of items in the non-monochromatic parts will be also small. So we can use few extra Pippenger's partially sorting circuits to finish the sorting. An interested reader might consult [24] for precise details.

The parameters W_i are chosen so that at each iteration the number of items in non-monochromatic parts within a block is $1/W_i$ faction of all the items in the block so we can afford to sort them using Pippenger's partially sorting circuit. Similarly for the number of parts where we are concerned about the logarithmic term in the complexity of Pippenger's partially sorting circuit used to sort the parts. In total, each iteration requires a circuit of size $O(nw)$ and depth $O(\log^2 W_i)$.

Thus we obtain a circuit sorting n w-bit integers according to one bit of total size $O(nw(1 + \log^* n - \log^* w))$ and depth $O(\log^2 n)$. It can be used along the lines of Asharov et al. [7] to build a circuit of size $O(nwk(1 + \log^* n - \log^* w))$ and depth $O(\log^3 n)$ that sorts w-bit integers according to their first k bits.

4 Lower bounds

Sorting is one of the most popular candidate functions for which people try to prove lower bounds. This is rather natural as sorting deals with transfer of information, and information is an aspect of computation that is quantitatively relatively well understood compared to computation itself. Already in 70's researchers tried to prove lower bounds for sorting on Turing machines. Stoss [35] proved an $\Omega(n \log^2 n)$ lower bound for sorting on Turing machines by algorithms that only move items around. This was extended by Paul [31] to algorithms which do not perform any "magic" defined in terms of Kolmogorov complexity. For Turing machines there was only little progress since then.

A modern take on excluding the "magic" is the Network Coding Conjecture of Li and Li [28]. Network coding is a problem in a communication network with ℓ source nodes $s_1, \ldots, s_\ell$ and target nodes $t_1, \ldots, t_\ell$ where each source s_i wants to be transmitting a stream of information to target t_i. The intermediate nodes in the network can combine received messages in arbitrary manner before

transmitting them further. The question is at what rate can the source-target pairs communicate when each link has some restricted capacity, and how this rate relates to the multicommodity flow in the corresponding network.

There are well known examples where the information rate can exceed the multicommodity flow in directed networks [1, 3]. The Network Coding Conjecture postulates that such situation cannot occur in undirected graphs [28]. So far the conjecture was established only in restricted settings. It is a compelling formulation of the "no magic" assumption.

There are multiple recent results establishing conditional lower bounds for sorting under the assumption that the Network Coding Conjecture is true such as lower bound for sorting in external memory [16] or for Boolean circuits [2, 7, 30]. Asharov et al. [7] show that under the Network Coding Conjecture, Boolean circuits sorting n integers w bits each partially according to k bits require size $\Omega(nk(w - \log n))$ even with no restriction on the depth of the circuits. Thus under the Network Coding Conjecture the Boolean circuits described in previous sections are almost optimal.

Sorting lower bound can also be derived from other assumptions. For example, Boyle and Naor [10] show that $\Omega(\log n)$ lower bound on the overhead of *offline oblivious RAM* would imply the super-linear lower bound $\Omega(n \log^2 n)$ on the size of sorting circuits. (It was erroneously thought that such lower bounds for oblivious RAM were already established.) Super-linear lower bounds on the size of sorting circuits of logarithmic depth would also follow from a *non-adaptive* n^ϵ lower bound on the number of queries for the function inversion problem with preprocessing of Hellman [12, 15, 19, 37].

All these partial results witness the centrality of sorting problem. In many ways, sorting seems to be the right candidate for proving super-linear size lower bounds for logarithmic depth circuits. Proving such a bound would constitute a major progress in computational complexity and we invite interested readers to take on this challenge. As Andrew Yao puts it: *You cannot win if you don't buy a lottery ticket.*

References

[1] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung. Network information flow. *IEEE Trans. Inf. Theory*, 46(4):1204–1216, 2000.

[2] P. Afshani, C. B. Freksen, L. Kamma, and K. G. Larsen. Lower Bounds for Multiplication via Network Coding. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, volume 132 of *LIPIcs*, pages 10:1–10:12, 2019.

[3] M. Adler, N. J. A. Harvey, K. Jain, R. D. Kleinberg, and A. R. Lehman. On the capacity of information networks. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'06)*, pages 241–250, 2006.

[4] A. Andersson, T. Hagerup, S. Nilsson, and R. Raman. Sorting in linear time? *Journal of Computer and System Sciences*, 57(1):74–93, 1998.

[5] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The design and analysis of computer algorithms*. Addison-Wesley, 1974.

[6] M. Ajtai, J. Komlós, and E. Szemerédi. Sorting in $c \log(n)$ parallel steps. *Combinatorica*, 3(1):1–19, 1983.

[7] G. Asharov, W.-K. Lin, and E. Shi. Sorting short keys in circuits of size $o(n \log n)$. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA'21)*, pages 2249–2268. SIAM, 2021.

[8] K. E. Batcher. Sorting networks and their applications. In *American Federation of Information Processing Societies: AFIPS Conference Proceedings: 1968 Spring Joint Computer Conference*, volume 32 of *AFIPS Conference Proceedings*, pages 307–314, 1968.

[9] D. Belazzougui, G. S. Brodal, and J. S. Nielsen. Expected linear time sorting for word size $\Omega(\log^2 n \log \log n)$. In *Proceedings of the 14th International Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2014*, volume 8503 of *Lecture Notes in Computer Science*, pages 26–37. Springer, 2014.

[10] E. Boyle and M. Naor. Is there an oblivious RAM lower bound? In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science (ITCS'2016)*, pages 357–368, 2016.

[11] A. K. Chandra, S. Fortune, and R. J. Lipton. Unbounded fan-in circuits and associative functions. *J. Comput. Syst. Sci.*, 30(2):222–234, 1985.

[12] H. Corrigan-Gibbs and D. Kogan. The function-inversion problem: Barriers and opportunities. In *Proceedings of the 17th International Conference on Theory of Cryptography, TCC 2019*, pages 393–421, 2019.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[14] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, USA, 2006.

[15] P. Dvořák, M. Koucký, K. Král, and V. Slívová. Data Structures Lower Bounds and Popular Conjectures. In *Proceedings of the 29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *LIPIcs*, pages 39:1–39:15, Dagstuhl, Germany, 2021.

[16] A. Farhadi, M. Hajiaghayi, K. G. Larsen, and E. Shi. Lower bounds for external memory integer sorting via network coding. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (STOC'19)*, STOC 2019, pages 997–1008, 2019.

[17] Y. Han. Improved fast integer sorting in linear space. *Inf. Comput.*, 170(1):81–94, 2001.

[18] Y. Han. Deterministic sorting in $o(n \log \log n)$ time and linear space. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing (STOC'02)*, pages 602–608, 2002.

[19] M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, 1980.

[20] F. C. Hennie. One-tape, off-line turing machine computations. *Inf. Control.*, 8(6):553–578, 1965.

[21] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, 43:439–561, 2006.

[22] Y. Han and M. Thorup. Sorting integers in $O(n\sqrt{\log \log n})$ expected time and linear space. In *Proceedings of the IEEE Symposium on Foundations of Computer Science (FOCS'02)*, 2002.

[23] S. Jukna. *Boolean function complexity: advances and frontiers*, volume 27. Springer Science & Business Media, 2012.

[24] M. Koucký and K. Král. Sorting Short Integers. In *Proccedings of the 48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *LIPIcs*, pages 88:1–88:17, Dagstuhl, Germany, 2021.

[25] D. E. Knuth. *The Art of computer programming, Volume 3: Sorting and searching*. Addison-Wesley, Reading, MA, 1973.

[26] D. G. Kirkpatrick and S. Reisch. Upper bounds for sorting integers on random access machines. *Theor. Comput. Sci.*, 28:263–276, 1984.

[27] K. Král. *Complexity of dynamic data structures*. PhD Thesis. Charles University, 2021.

[28] Z. Li and B. Li. Network coding: The case of multiple unicast sessions. *Proceedings of the 42nd Allerton Annual Conference on Communication, Control, and Computing*, 2004.

[29] W.-K. Lin and E. Shi. Optimal sorting circuits for short keys. *arXiv preprint arXiv:2102.11489*, 2021.

[30] W. Lin, E. Shi, and T. Xie. Can we overcome the n log n barrier for oblivious sorting? In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'19)*, pages 2419–2438, 2019.

[31] W. J. Paul. Kolmogorov complexity and lower bounds. In *Proceedings of the Conference on Algebraic, Arithmetic, and Categorial Methods in Computation Theory: Fundamentals of Computation Theory, FCT 1979*, pages 325–334, 1979.

[32] N. Pippenger. Superconcentrators. *SIAM J. Comput.*, 6(2):298–304, 1977.

[33] N. Pippenger. Self-routing superconcentrators. *J. of Comp. Syst. Sci.*, 52(1):53–60, 1996.

[34] D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985.

[35] H. Stoß. Rangierkomplexität von permutationen. *Acta Informatica*, 2:80–96, 1973.

[36] L. G. Valiant. Graph-theoretic properties in computational complexity. *J. Comput. Syst. Sci.*, 13(3):278–285, 1976.

[37] E. Viola. Lower bounds for data structures with space close to maximum imply circuit lower bounds. *Theory of Computing*, 15(18):1–9, 2019.

THE CONCURRENCY COLUMN

BY

NOBUKO YOSHIDA

Department of Computing
Imperial College London
180 Queen's Gate, London, SW7 2AZ
n.yoshida@imperial.ac.uk, http://mrg.doc.ic.ac.uk/

BEATCS no 135

INTERVIEWS WITH THE 2021 CONCUR TEST-OF-TIME AWARD RECIPIENTS

Luca Aceto
ICE-TCS, Department of Computer Science,
Reykjavik University
Gran Sasso Science Institute, L'Aquila

Nathalie Bertrand
Univ Rennes, Inria, CNRS, IRISA

Nobuko Yoshida
Department of Computing, Imperial College London, UK

Abstract

Last year, the CONCUR conference series inaugurated its Test-of-Time Award, purpose of which is to recognise important achievements in Concurrency Theory that were published at the CONCUR conference and that have stood the test of time. This year, the following four papers were chosen to receive the CONCUR Test-of-Time Awards for the periods 1994–1997 and 1996–1999 by a jury consisting of Rob van Glabbeek (chair), Luca de Alfaro, Nathalie Bertrand, Catuscia Palamidessi, and Nobuko Yoshida:

- David Janin and Igor Walukiewicz. On the Expressive Completeness of the Propositional mu-Calculus with respect to Monadic Second Order Logic [3].
- Uwe Nestmann and Benjamin C. Pierce. Decoding Choice Encodings [4].
- Ahmed Bouajjani, Javier Esparza, and the late Oded Maler. Reachability Analysis of Pushdown Automata: Application to Model-checking [2].
- Rajeev Alur, Thomas A. Henzinger, Orna Kupferman, and Moshe Y. Vardi. Alternating Refinement Relations [1].

This year, the second paper was live-interviewed by Nobuko Yoshida; the third paper was interviewed by Nathalie Bertrand and the forth paper was interviewed by Luca Aceto. Adam Barwell and Francisco Ferreira helped making the article from the live interview by Yoshida.

1 Interview with David Janin and Igor Walukiewicz

Luca: You receive the CONCUR ToT Award 2021 for your paper 'On the Expressive Completeness of the Propositional mu-Calculus with respect to Monadic Second Order Logic', which appeared at CONCUR 1996. In that article, you showed what I consider to be a fundamental result, namely that the mu-calculus and the bisimulation-invariant fragment of monadic second-order logic (MSOL) have the same expressiveness over transition systems. Could you tell us how you came to study that question and why do you think that such a natural problem hadn't been answered before your paper?

David and Igor: At that time we were interested in automata characterizations of the expressive power of MSOL and the mu-calculus. In 1988 Damian Niwinski has shown that over binary trees the mu-calculus and MSOL are expressively equivalent. When it appeared, the result was quite surprising, even unexpected. The two logics are not equivalent on unranked trees where the number of children of a node is not fixed. In consequence, the logics are not equivalent over the class of all transition systems.

In 1983 van Benthem showed that modal logic is equivalent to the bisimulation invariant fragment of first-order logic. When we have learned about this result we have realized that our automata models can be used to prove a similar statement for the mu-calculus.

The method of van Benthem could not be applied to MSOL, the automata based method looks like the only way to prove the result.

Luca: I am interested in how research collaborations start, as I like to tell 'research life stories' to PhD students and young researchers. Could you tell us how you started your collaboration?

David: I came to meet Igor when he was visiting Bordeaux in fall 93, presenting his first completeness result on a proof system for the mu-calculus (LICS 93). I was myself starting a PhD considering modal logic for specifying (various forms of) nondeterminism in connection with (power)domain theory (FSTTCS 93).

We eventually met again in Auvergnes (centre of France) for the Logic Colloquium in summer 94. There, spending time in the park nearby the conference venue or walking around the volcanoes, we elaborated a notion of nondeterministic automata for the modal mu-calculus. This result can be seen as extending the notion of 'disjunctive' normal form from propositional logic to the modal mu-calculus. I remember that Igor later used this result for his second completeness result for Kozen's proposed proof system for the mu-calculus (LICS 95). Thanks to Igor, this was for me the occasion to learn in depth the link between mu-calculus and tree automata.

Incidentally, Johan van Benthem was also attending this Logic Colloquium and we both attended his lecture about the bisimulation-invariant fragment of FO. Although we did not yet realize we could generalize his result to MSO, this surely increased our own understanding of logic.

Our first result (Automata for the mu-calculus) was presented at MFCS in 95 in Prague, where Igor and I met again. It could have been there, or a little later, that we eventually postulated our bisimulation-invariance result. However, proof arguments were only found later while exchanging emails. It was a bit amazing for me that we could discuss that way across Europe: email started to be heavily used only in the late 80's. In my head, Poland was far, far away from France.

Yet our collaboration was eventually in the line of an ongoing collaboration between Warsaw and Bordeaux, involving researcher like Arnold (my supervisor) and Niwinski (Igor's mentor), both major specialists in the area of automata and logics. Somehow, as a followup, together with Aachen (Grädel and Thomas) and many other sites, the GAMES European network was later created, and, almost at the same time, Igor came to Bordeaux as a CNRS researcher.

Luca: As you mentioned earlier, van Bentham has shown that modal logic has the same expressive power as the bisimulation-invariant fragment of first-order logic. In some sense, one may consider your main result as the extension of van Bentham's theorem to a setting with fixed-points. Could you briefly describe at a high level the challenges that fixed-points created in your w ork? To your mind, what was the main technical achievement or technique in your paper?

David and Igor: Sure our result is similar to van Benthem's, and, as mentioned above, his own presentation in 1993 was very inspiring. However, his proof relies on compactness of first-order logic and cannot be adapted to monadic second-order logic. In our proof, we have used automata-theoretic techniques.

In our previous works we had developed automata models for MSOL and the mu-calculus on unranked trees. Every transition system is bisimulation invariant to a tree obtained by the unfolding of the transition system. Crucially for our result, it is also equivalent to a tree where every subtree is duplicated infinitely many times. A short pumping argument shows that on such trees the two automata models are equivalent.

Luca: Did any of your subsequent research build explicitly on the results and the techniques you developed in your award-winning paper? Is there any result obtained by other researchers that builds on your work and that you like in particular?

David and Igor: Around that time Marco Hollenberg and Giovana D'Agostino used our automata-theoretic methods to show the uniform interpolation prop-

erty for the mu-calculus.

In collaboration with Giacomo Lenzi (postdoc in Bordeaux from Pisa), David considered the bisimulation-invariant fragment of various levels of the monadic quantifier alternation hierarchy. It turns out that monadic Σ_1 corresponds to reachability properties and monadic Σ_2 corresponds to Büchi properties, respectively the first and second levels of the mu-calculus hierarchy.

It could be the case that all mu-calculus can be translated into monadic Σ_3 formulas – this is true when restricting to deterministic transition systems. However, with nondeterminism, such a result seems difficult to achieve.

Incidentally, Giacomo and David also proved that adding limited counting capacity to modalities yields a fixed-point calculus equivalent to the unravelling invariant fragment of MSO (LICS 2001).

In 1999, in collaboration with Erich Grädel, Igor has used similar techniques to define and study guarded fixed-point logic. Subsequently, several other fixed-point logics of this kind were proposed with the most expressive one probably being guarded negation logic with fixed points of Bárány, ten Cate, and Segoufin from 2015. These works all use automata-theoretic methods to some extent.

Luca: Your paper was written while Igor was at BRICS at the University of Aarhus. Igor, what was it like to be at BRICS@Aarhus at that time? What role do you think centres like BRICS played and can play in the development of theoretical computer science? Do you think that your stay at BRICS had a positive impact on your future career?

Igor: My stay at BRICS had definitively a beneficial impact on me . At that time BRICS was one of the most active centers world wide in theoretical computer science. Being able to see and talk to so many different people, being exposed to so many different ideas, was very enriching. BRICS was a meeting place allowing scientists to have a better and larger vision of our field. BRICS contributed to the development of many people involved, as well as to the excellence of Aarhus, and even Denmark as a whole, in our field.

Luca: I have been brought up in the concurrency-theory tradition and I feel that bisimulation-invariant properties are the interesting ones over transition systems. Do you think that we actually 'need' logics that allow one to specify properties of transition systems that are not bisimulation invariant?

David: That was the idea indeed and the mu-calculus is the good logic for that. From a mathematical perspective, bisimulation is a fairly natural definition. In some sense, bisimulation equivalence is a greatest co-congruence.

However, I always suspected that bisimulation is too fine grained as an equivalence for concurrency. When modelling realistic systems, nondeterminism arises

either as controllable (angelic) or uncontrollable (demonic) choice. In this case, the right equivalence to be considered is simulation equivalence.

The funny thing is that David Park, who 'invented' bisimulation, was actually studying equivalence of J.R. Büchi's deterministic string automata where, because of determinism, bisimulation turns out to be equivalent to simulation equivalence. It is only after Matthew Hennessy and Robin Milner's work in concurrency that bisimulation was applied to nondeterministic behaviors.

Igor: In the context of XML and semistructured data, bisimulation is not relevant. One of the prominent queries in XPATH is whether a value appears twice – that is clearly not a bisimulation-invariant property. So while XPATH looks very close to PDL or the mu-calculus, many techniques and questions are very different. The other context that comes to mind is controller synthesis, where we ask for a transition system of a specific shape, for example, with self-loops on certain actions. Such self loops represent invisibility of the action to the controller.

Luca: What are the research topics that you find most interesting right now? Is there any specific problem in your current field of interest that you'd like to see solved?

David: In Logic for Computer Science, there have always been two kinds of approaches: model theory that eventually led to model checking and proof theory that eventually led to typed programming languages.

Twenty years later, aiming at designing and implementing real concurrent systems, especially for interactive arts, I realized that the latter approach was (at least for me) a lot more effective. Building concurrent systems by synchronizing arbitrary sub-systems sounded for me like unstructured programming; it was essentially unmaintainable. Monads and linear types, among many other approaches in typed functional programming, surely offered interesting alternatives to process calculi approaches.

Igor: In the context of the paper we discuss here, I am surprised by developments around the model-checking problem for the mu-calculus. After some years of relative calm, Calude, Jain, Khoussainov, Li and Stephan have made an important breakthrough in 2017. Yet, despite big activity after this result, the research on the problem seems to have hit one more barrier. Another old prominent problem is decidability of the alternation hierarchy for the mu-calculus. The problem is: given a formula and a number of alternations between the least and the greatest fixed-points, decide if there is an equivalent formula with this number of alternations. Even when the number of alternations is fixed we do not know the answer. Among others, Thomas Colcombet and Christof Loeding have done very interesting work on this subject.

Luca: What advice would you give to a young researcher who is keen to start working on topics related to automata theory and logic in computer science?

David: From the distance, our result sounds to me as a combination of both technique and imagination. Technique for mastering known results and imagination for finding original open problems, especially between research fields that are not (yet) known to be deeply related. As a matter of fact, I felt lucky finding such a fresh open problem that was (probably) a lot easier to solve than many other well-known hard problems.

Technique comes from hard work. It is obviously essential but somehow easy to teach and evaluate in academia. Imagination comes from curiosity. It is still essential but a lot more difficult to teach. So young researchers must develop by themselves their own imagination and curiosity.

In the automata-theory branch of logic in computer science, the remaining open problems seem fairly hard, so I believe that it is the imagination of young researchers that will make the difference for setting up new interesting directions of research, especially those who are ready to look aside, towards other areas of logic in computer science and, because it can be a considerable source of motivation, funny applications.

Igor: Naturally, the field is much broader these days than it was 25 years ago. It is crucial to master some techniques. For this, working on variations of already solved problems is a good method. Yet, I think it is important to escape the cycle of constant modifications of existing problems and their solutions. I would suggest that, at some moment, one should find an important open problem that one is passionate about and should spend a considerable effort on it. I admit that this is a matter of a taste, personality, and having a sufficiently comfortable situation to afford such a risk. Another good option is to look at frontiers with other areas: distributed computing, semantics, control theory, security.

2 Interview with Uwe Nestmann and Benjamin Pierce

Nobuko: You receive the CONCUR 2021 Test-of-Time Award for your paper "Decoding Choice Encodings", which appeared at CONCUR 1996. Could you tell us briefly what lead you to embark on studying the expressiveness of choice in the asynchronous pi-calculus?

Uwe: I did my diploma thesis in '91. I was working on a topic that had to do with communicating functions. I built a lambda calculus with communication in it. It was a typed lambda calculus and it was strong enough to type the Y-combinator. I went to Edinburgh and presented that at the Concurrency Club, and

they asked me, "*who is your supervisor?*". I didn't really have one, because I was mostly self-driven those days. What they told me is that it's better to not work on this topic without a competent supervisor. The second piece of advice was to get a supervisor first and then look for a t opic. So I found B enjamin. Benjamin had this wonderful project at the time on trying to make a programming language out of the pi-calculus and, in that language, choice encodings (or at least choice operators) played a role. He invited me to visit him in Paris, where he was on six-months' leave, I think.

Benjamin: It was actually a "nested postdoc." I did three postdocs after finishing my PhD at CMU: one in Edinburgh, one in Paris, and one in Cambridge. The Paris one was nested inside the Edinburgh one.

Uwe: I was in Paris for one week, and Benjamin told me to try programming in his new programming language, Pict. I tried to write down the dining philosophers problem, in a way such that I could use an abstraction to pick up forks in either order, and I wanted to instantiate it with left and right, and right and left, but the Pict language didn't allow me to do so. Behind was the interplay between choice and abstraction (and instantiation) and that was the start of all of it from my point of view. Then I wrote up an exposé and I ended up actually working on just a third of that for my PhD thesis. And of course, there were technical reasons for Benjamin and Dave [Turner] at the time for being interested in choice constructs.

Benjamin: Of course, Dave Turner is the most important name that needs to be mentioned here, besides obviously Robin Milner. All of this was happening under the umbrella of Robin's wonderful work on pi-calculus and the amazing group that he had assembled at the time. He had this incredible set of students, including Dave Turner and David Sangiorgi and Peter Sewell, doing all sorts of things with pi-calculus. Dave, besides being a first-class hacker, was also a really good theoretician. He truly married the two. He and I started talking at some point about what kind of programming language would you get if you treated the pi-calculus like the Lisp people treated the lambda calculus. What that led to was a lot of different language designs based on different versions of the pi-calculus, but we kept wanting to make it simpler and simpler. Partly because we were thinking of it as possibly even a distributed language, not just a concurrent language, and as everybody knows, the choice operator – in the full-blown pi-calculus or CCS sense – is not a real thing in distributed systems: it's not implementable. So we were trying to make the underlying calculus simpler and simpler, and eventually wound up with this programming language with no choice at all. But as Uwe discovered, there are things that you might want to do where choice is the natural primitive, like dining philosophers, which raises the question of how much of it can you get just by programming on top of plain parallel composition plus messages on channels. We found that programming up a restricted form of choice was a little tricky but not *that* tricky. What *was* really tricky, though, was justifying that it was

correct. The reason why it turned into a whole dissertation for Uwe, was because the well-known notions of correctness that were lying around did not apply to this situation. I remember being totally astonished at the length and technicality of the final proof that Uwe ended up doing.

Nobuko: Did you imagine at the time that your award-winning paper would have so much impact on the area of expressiveness in concurrency theory, and how do you feel now?

Benjamin: Maybe Uwe did; I did not. I think we were just following our noses.

Uwe: Actually, I would say "yes" and "no". The "no" is when it came to the CONCUR acceptance, I got the impression that we just about made it because the competition was so tough and the pi-calculus was really hot at that time. There were six or seven pi-calculus papers in the conference in the end (I don't know how many were in the submission pool). The tiny "yes" that I would like to say is because Kohei [Honda] had foreseen it. When I gave the presentation at the Newton Institute just in autumn '95 – that was the workshop that Benjamin organised on concurrent high-level languages – Kohei came to me after the talk and said something like, "*maybe you don't know yet, but you will be known for this*". I can't remember the exact wording. I think he called it "*Nestmann's Theorem*" or something. Me, a PhD student, the first time in front of this crowd of expert people and then he tells me something like that. I didn't believe him. Of course not.

Benjamin: Kohei was ahead of his time in so many ways.

Nobuko: Could you tell us what the research environment was like in Edinburgh and the UK as a whole at that time and how it has influenced the rest of your career?

Benjamin: I came as a postdoc to Robin's group. I was the last postdoc of Robin's in Edinburgh, and then travelled with him to Cambridge, where Peter Sewell and I were his first postdocs. I would say that both Edinburgh and Cambridge at the time, and still, were just incredible. In Edinburgh you had Milner, you had [Gordon] Plotkin, you had Don Sannella, you had students around you like Martin Hofmann and Philippa Gardner and Marcelo Fiore, and the list goes on and on. You had other postdocs like Randy Pollack. It was just an incredible place. People talking about amazing, deep, mind-bending things all the time. It was particularly an amazing place for thinking about concurrency. There were a lot people breaking new ground.

Nobuko: Benjamin, how did that experience influence your current research?

Benjamin: For one thing, it solidified my interest in language design. The whole Pict experience was so fruitful. It was so much fun working with Dave on

implementing this interesting language, and both the design and the programming that we did in Pict gave rise to so many interesting questions. For example, it led us to do a bunch of thinking about type systems for concurrency, and I can see echoes of those ideas in the work that you, Nobuko, and colleagues have done more recently with session types. Though I don't consider myself a core concurrency researcher any more, the experience gave me an appreciation for the theory of concurrency that has drawn me back to the area over and over.

Nobuko: Uwe, how did it influence your research?

Uwe: I did my PhD in Erlangen (University of Erlangen-Nürnberg), which was a place that was not so much known at that time for theory, and especially not for concurrency theory. I had the opportunity by a bilateral travel exchange programme between these two universities pushed by my other supervisor, Terry Stroup, at that time. And when I came into Edinburgh, not only was there so much competence around, which is mainly what Benjamin summarised, but there was so much openness. There was so much openness for any kind of ideas. So much curiosity and joy. So I was very lucky that I could regularly, every couple of months, visit the LFCS for a few days. There, I was pumped up with content and ideas, and did a presentation in the pi club in Robin's tiny office, with almost ten people sitting around a tiny blackboard, listening to my ideas and my problems. It was just unbelievable at this time. That kind of culture and atmosphere was so great. I traced it back, in May or June '95, since we're talking about this particular paper, it was culminating in the crucial part where I was just before proving choice encodings correct. I only needed two ingredients. One came a week later by Davide Sangiorgi posting, for the first time, a short note on asynchronous bisimilarity. And the other was that we were rediscovering, mostly together in the pi club with Ole-Høgh Jensen and Robin Milner, the notion of coupled similarity. Both Ole and Robin had different ideas and came to the same conclusion. I came back to Erlangen and found the old paper on coupled similarity by [Joachim] Parrow and [Peter] Sjödin and within a week all of the pieces were just about there. I "simply" had to write down the details and convince myself that it went all the way through. That was the crucial moment and without Edinburgh, without this culture, these possibilities, this openness, it would not have happened and maybe I would not even have become a professor in Berlin. Just because of this tiny situation and getting together of bright people.

Nobuko: Studying expressiveness this way was quite new and at the beginning at that time, so you probably cared a lot about presentation and how to communicate your ideas. Do you have any comments about this aspect? I found your paper is still very readable and very clearly written for such a subtle paper. How did you go about writing with this in mind? Apart from technical details.

Uwe: I was a great fan of Benjamin's presentation and communication skills at that time. I was seeing him on stage and reading his papers and I had the possibility to closely interact with this impressive guy and learn from him. Just recently, I learnt about a citation that summarises this approach about writing: *"Do not try to write such that you are understood. Try to write such that you cannot be misunderstood."* I think this expresses precisely what I think I learnt back then in trying to get this paper out, and making these subtle observations, and finding the right notation. It's often underestimated how important the role of good notation is for getting things across. The same goes for graphical presentations. And then, polishing, polishing, polishing, polishing. *"Get simpler sentences,"* Benjamin always said. I'm German, you know, we like complicated constructions which are somewhat nice and deeply nested. I learnt at the time to get it as simple as possible. Presentations were another thing. I found my presentation back at the Newton Institute again and I remember I had this table of contents written with **ABCDE**, which were the initial letters of the concepts that I presented: **A**synchrony, **B**isimulation, **C**oupled similarity, **D**ecoding, and I think E was for **E**nd or something. I obviously like playing with words, and I admire the power and joy of well-chosen language.

Nobuko: I do remember your presentation. You highlighted a coupled simuluation as a part of Rob [van Glabbeek]'s famous diagram at branching bisimilarities' CONCUR paper. I still remember your presentation at Newton Institute.

Benjamin: I have always cared a lot about good writing. Communicating ideas is really one of the most important parts of an academic's job. So it feels important to acknowledge the people I learned about writing from. The first was Don Knuth – his level of attention to writing, among all the other things he did, is totally inspiring to me. The other was John Reynolds, who was one of my two supervisors as a PhD student and who is the most careful writer that I've ever worked closely with. He gave me one time a draft of one of his papers to proofread, and I said to myself, *"Aha, this is my chance to get back at him for all the mistakes and flaws he has found in my writing over the years!"* So I started reading it, and I got more and more frustrated because I couldn't find *anything* to improve. Anything at all! In the whole paper – not a comma, not a notational choice, not the way something was worded. Nothing. That experience was both an inspiration and a humbling lesson to me.

The biggest thing I've learned over the years about writing is that the biggest ingredient of good writing is exactly what Uwe brought to this paper: the willingness to iterate until it's good. Good writers are people that stop polishing later than bad writers.

Nobuko: How much of your later work has built on your award-winning paper? What follow-up result of yours are you most proud of and why?

Uwe: I would like to mention three. Funnily, neither of them was in the decade following the paper. The reason may be because I was dragged into other projects, having to do with security protocols, pi-calculus, and object calculi. (1) By accident, I got back in contact with Ursula Goltz, who was one of my PhD referees: she was working on a project on synchronous and asynchronous systems and she asked me for literature because she knew I was digging deep in the 80s about results on the first CSP implementations. In the course of this project, I got back to actually directly building upon my PhD work, and I found Kirstin Peters, at that time a PhD student, who got interested in that. We found a number of remarkable observations having to do with distributed implementability and notions of distributability and what this may have to do with encodings between calculi. We discovered a hierarchy of calculi where you can very easily see which of them are at the same level of distributed implementability. We found that the asynchronous pi-calculus is actually not fully implementable in a distributed system, like many others. There is the ESOP paper in 2013, which I'm very proud of. Kirstin pushed this reasearch much further. (2) Another follow-up work concerns the notion of correctness that we were applying in the awarded paper, it was a lot about a direct comparison between terms and their translations. Not by plain full abstraction on two different levels and having an if and only if, but a direct translation so you could not distinguish a term from its translation. This kind of observation led to a rerun of, say, the research on what we actually want from an encoding. What is a good criterion for a good encoding? This culminated in the work with Daniele Gorla where we criticised the notion of full abstraction in the sense that it's a very important notation but you can easily misuse it and get to wrong results or useless results. (We also emphasised the importance of operational correspondence, and Daniele went on to establish his, by now, quite standard and established set of criteria for what a good encoding is.) That is a nice highly abstract paper with Daniele in Mathematical Structures in Computer Science in 2016, only, so also well, well after the CONCUR paper in 1996. (3) In just the last two or three years, my PhD student Benjamin Bisping finally studied algorithms and implementations for checking coupled similarity. We found an amazing wealth of new views on these kinds of equivalences that are slightly weaker than weak bisimilarity. So back to the roots, in a sense, to what we were doing 25 years ago. (Like Kirstin Peters and Rob van Glabbeek who further showed that coupled similarity is in fact very closely connected to encodings, in general.) Seeing these developments makes a lot of fun.

Nobuko: This was a TACAS paper, right?

Uwe: Yes, and we also published the survey article "Coupled Similarity – The First 32 Years", for the Festschrift for Robert van Glabbeek. It's basically an advertising paper for this great notion of equivalence, which is highly underestimated. It's, in a sense, much better than weak bisimilarity. Especially if you're

interested in – and this is my favourite domain – distribution, distributability, distributed implementations.

Nobuko: Benjamin, do you have any further comments?

Benjamin: For me, the answer is a little more oblique. I haven't written papers about choice encodings and things like that, besides this one. But what it did for me was to really solidify my interest in the asynchronous pi-calculus as a foundation for programming languages – and as a foundation for thinking about concurrency – because this paper, Uwe's result, teaches us that the asynchronous pi-calculus is more powerful than it looks – powerful enough to do a lot of programming in. You know there's this famous quote attributed to Einstein, *"Make everything as simple as possible, but no simpler."* I felt like the asynchronous pi-calculus was kind of "it" after seeing this result. And that calculus then became the foundation for a whole bunch of my later work on type systems for concurrency and language design.

Uwe: Actually the encodings we did back then went into what is now called the "localised asynchronous pi-calculus", but it simply wasn't yet known back then. The localised asynchronous pi-calculus is at this perfect level of distributed implementability, as we know by now.

Nobuko: This is partly also Massimo Merro did with Davide Sangiorgi, right?

Uwe: Yes, they did this few years later, towards the end of the '90s.

Nobuko: What do you think of the current state and future directions of the study of expressiveness in process calculi, or more generally, concurrency theory as a whole?

Uwe: Back then, in Cambridge, I was having discussions with Peter Sewell. Quite many of them. At the time, we were making fun by saying, *"now we know how to do process calculi, we can do five of them for b reakfast."* We know the techniques, we know how to write down the rules, we know what to look for in order to make it good. And I would say that for studying encodings nowadays it's kind of the same level of maturity; we know what to look for when writing down encodings, pitfalls to avoid, and it's done. So what I found most interesting today, is that often enough, the proximity between encodings and actually doing implementations is very close and that is may be because the maturity of programming languages we can use is much higher. We can use convenient abstractions in order to more-or-less straightforwardly write down encodings.

What's going on? Current state and future directions. The EXPRESS/SOS workshop is still alive and kicking. It attracts great papers and not that many are submitted but typically they're great papers and I think that's good. I think we had an impact on concurrent programming, and for example, if you look at Google Go, the concurrency primitives that you find in there is pretty much a process

calculus. It's message passing, and choice, and even mixed choice, and stuff like that. I cannot say right now that there are deep, deep, deep questions to be solved about encodings except for finding out what Robert van Glabbeek's criteria have to do with Daniele Gorla's criteria. There is an ongoing debate, but the issues are quite technical. What could use more research is typed languages, typed calculi, and typed encodings. It has been done and we have many nice results, but I think there are still some open questions on what the ideal criteria should be on those.

Nobuko: What advice would you give a young researcher interested in working on concurrency theory and process calculus like today?

Benjamin: My best advice for people that want to do theory is: keep one foot in practice. Don't stop building things. Because that's the way you find interesting problems, it's the way you keep yourself grounded, it's the way you make sure that the directions you're looking and the questions that you're asking have something to do with ... something! It's the way to stay connected to reality while also generating great questions.

Nobuko: Uwe, do you have anything to add to that?

Uwe: Having a foot in practice is also good for actually checking and finding mistakes in your reasoning. Building systems not only for finding problems but also for finding out that you have a problem in your thinking. Apart from that, I would not like to push for any particular area for concurrency theory. I mean, concurrency theory is incredibly wide. My advice is: get the best possible supervisor that you can find and then work on his project. I think this is very good advice. Be patient, dig deep. This is very general advice. Never give up. It took me two years until, in one week, the pieces fell together. So be patient, dig deep, and train your communication skills, practice networking. All the general things. Ah, and maybe what I found very useful for my own career: learn the basics and the history of your field. Understand what has already been found and what that means even twenty years after. I learned a lot from the early 80s papers that I was mentioning beforehand on first implementations of the communication primitives of CSP. There is one published supposedly deadlock-free algorithm, which almost twenty years later was discovered to be incorrect. The proof was incorrect; it was not actually deadlock free. So, work on hard problems, dig deep, be patient. And communicate well. This is also the best way to get help.

Nobuko: Wow. Anyone who can satisfy everything would be quite a fantastic student. (Laughs.) Like you, Uwe, you know.

Nobuko: This is the last question: what are the research topics that currently excite you the most? Can I ask Benjamin first?

Benjamin: I will name two. One is machine-checked proofs about real software. Over the past fifteen or twenty years, the capabilities of proof assistants,

and the community around them, have reached the point where you can really use them to verify interesting properties of real software; this is an amazing opportunity that we are just beginning to exploit.

On a more pragmatic level, I'm very interested lately in testing. Specifically, specification-based (or property-based) testing in the style popularised by QuickCheck. It's a beautiful compromise between rigor and accessibility. Compared to the effort of fully verifying a mathematically stated property, it is incredibly easier and lower-cost, and yet, you can get tremendous benefit, both from the process of thinking about the specification in the mathematical way that we're used to in this community and from the process of testing against, for example, randomly generated or enumerated examples. It's a sweet spot in the space of approaches to software quality.

Nobuko: These things are still very difficult for concurrency or distributed systems. Do you have any comment because proof assistants for concurrency theory is, I think, still quite difficult compared to hand-written proof.

Benjamin: Yes, in both domains – both verification and testing – concurrency is still hard. I don't have a deep insight into why it is hard in the verification domain, beyond the obvious difficulty that the properties you want are subtle; but in the testing domain, the reason is clear: the properties have too many quantifier alternations, which is hard for testing. Not impossible – not always impossible, anyway – but it raises hard challenges.

Uwe: There's a recurring pattern in what I like doing and that is always to do with looking at different levels of abstractions. You can think of it in terms of encodings or as a distributed system, and I was always wondering about the relation between global (higher-level) properties and local (lower-level) implementation of systems. And throwing formal methods, formal models, and theories at this problem has always been what I liked, and I still do that, nowadays again, more on fault-tolerant distributed algorithms. Maybe also because of the recent hype due to blockchain and the strong interest in practical fault-tolerant Byzantine algorithms, and so on. And, here I meet Benjmain again, at best doing mechanical verification of those. Mechanical verification is still hard and you can easily pull PhD students into a miserable state by dragging them onto a problem that takes an awful lot of time, and then you get out one paper, with the proof in Isabelle, in our case. On the other hand, it's getting more and more a tool that we just use. The more you've done, using a proof assistant, the more you integrate it into your everyday life. Some students, as a standard, test their definitions and their theorems and do their proofs in Isabelle and we now even have bachelor students using that. Good ones, I mean bright ones, of course, but it's becoming more and more an everyday thing. The other idea: Benjamin, you're well-known also for the software foundations series. I don't know whether you've done pedagogical research, learning theory, on top of that in the following sense. What we are inter-

ested in, just recently, is understanding how people learn how to do proofs. It's a long, difficult, mental process and there are a number of theories about this actually works, and whether this works, and there's magic involved, and whatnot. And getting used to, of course. Learning from patterns. But then, what could be the impact of using proof assistants for learning how to do proofs? Does it actually help? Or does it actually hinder?

Benjamin: It turns people into hackers. (Laughs.)

Uwe: Yeah, yeah, yeah. We're talking about computer science students, not maths students, right? Programming is proving, proving is programming. This is of course a slogan from type theory, but one may actually use it as a motivation to write down first proofs, getting feedback from the proof assistant, and go on from there. This is one of the interests that we have, in actually understanding this process of learning how to do proofs.

Nobuko: I now conclude this interview. Thank you both very much for giving us your time.

3 Interview with Ahmed Bouajjani and Javier Esparza

Nathalie: You receive the CONCUR ToT Award 2021 for your paper with Oded Maler *Reachability Analysis of Pushdown Automata: Application to Model-Checking*, which appeared at CONCUR in 1997. In that article, you develop symbolic techniques to represent and manipulate sets of configurations of pushdown automata, or even of the broader class of alternating pushdown systems. The data structure you define to represent potentially infinite sets of configurations is coined alternating multi-automata, and you provide algorithms to compute the set of predecessors (pre*) of a given set of configurations. Could you briefly explain to our readers what alternating multi-automata are?

Ahmed: The paper is based on two ideas. The first one is to use finite automata as a data structure to represent infinite sets of configurations of the pushdown automaton. We called them multi-automata because they have multiple initial states, one per control state of the pushdown automaton, but there is nothing deep there. The second idea is that this representation is closed under the operation of computing predecessors, immediate or not. So, given a multi-automaton representing a set of configurations, we can compute another multi-automaton representing all their predecessors. If you compute first the immediate predecessors, then their immediate predecessors, and so on, you don't converge, because your automata grow bigger and bigger. The surprising fact is that you can compute all predecessors in one go by just adding transitions to the original automaton, without

adding any new states. This guarantees termination. Later we called this process "saturation".

Once you can compute predecessors, it is not too difficult to obtain a model-checking algorithm for LTL model checking. But for about branching-time logics you must also be able to compute intersections of sets of configurations. That's where alternation kicks in, we use it to represent intersections without having to add new states.

Nathalie: Could you also tell us how you came to study the question addressed in your award-winning article? Which of the results in your paper did you find most surprising or challenging?

Javier: In the late 80s and early 90s many people were working on symbolic model-checking, the idea of using data structures to compactly represent sets of configurations. BBDs for finite-state model-checking were a hot topic, and for quite a few years dominated CAV. BDDs can be seen as acyclic automata, and so it was natural to investigate general finite automata as data structure for infinite-state systems. Pierre Wolper and his group also did very good work on that.

About your second question, when I joined the team Ahmed and Oded had already been working on the topic for a while, and they had already developed the saturation algorithm. When they showed it to me I was blown away, it was so beautiful. A big surprise.

Nathalie: In contrast to most previous work, your approach applied to model checking of pushdown systems treats in a uniform way linear-time and branching time logics. Did you apply this objective in other contributions?

Javier: I didn't. The reason is that I was interested in concurrency, and when you bring together concurrency and procedures even tiny fragments of branching-time logics become undecidable. So I kind of stuck to the linear-time case. Did you work on branching-time, Ahmed?

Ahmed: Somehow yes (although it is not precisely about linear vs branching time properties), in the context of Regular Model Checking, a uniform framework for symbolic analysis of infinite-state systems using automata-based data structures. There, we worked on two versions, one based on word automata for systems where configurations can be encoded as words or vectors of words, such as stacks, queues, etc., and another one based on tree automata for configurations of a larger class of systems like heap manipulating programs, parametrised systems with tree-like architectures, etc. The techniques we developed for both cases are based on the same principles.

Nathalie: As it is often the case, the paper leaves some open questions. For instance, I believe, the precise complexity of verification of pushdown systems against CTL specifications is PSPACE-hard and in EXPTIME. Did you or others close this gap since? Did your techniques help to establish the precise complexity?

Ahmed: In our paper we showed that model checking the alternating modal

mu-calculus is EXPTIME-hard. CTL is less expressive, and it was the most popular logic in the verification community at the time, so it was natural to ask if it had lower complexity.

Javier: Yes, as a first step in the paper we showed that a fragment of CTL, called EF, had PSPACE complexity. But I made a mistake in the proof, which was later found by Igor Walukiewicz. Igor cracked the problem in a paper at FSTTCS'00. It turns out that EF is indeed PSPACE-complete (so at least we got the result right!), and full CTL is EXPTIME-complete. I wish Igor had used our technique, but he didn't, he applied the ideas of his beautiful CAV'96 paper on parity pushdown games.

Nathalie: It is often interesting to understand how research collaborations start as it can be inspiring to PhD students or colleagues. Could you tell us how you started your collaboration on the award-winning paper? Did you continue working together (on a similar topic or on something totally different) after 1997?

Ahmed: Javier and I first met in Liege for CAV 9 5. French universities have this program that allows us to bring foreign colleagues to France for a month as invited professors, and I invited Javier to Grenoble in 96.

Javier: It was great fun; Verimag was a fantastic place to do verification, we both liked cinema, Ahmed knew all restaurants, and the Alps were beautiful. Ahmed invited me again to Grenoble in 97. This time I came with my wife, and we again had a great time.

When I arrived in Grenoble in 96 Ahmed and Oded had already written most of the work that went into the paper. My contribution was not big, I only extended the result to the alternation-free mu-calculus, which was easy, and proved a matching lower bound. I think that my main contribution came *after* this paper. Ahmed and Oded were too modest, they thought the result was not so important, but I found it not only beautiful, I thought it'd be great implement the LTL part, and build a model checker for programs with procedures. We could do that thanks to Stefan Schwoon, who started his PhD in Munich around this time–he is now at Paris-Saclay—and was as good a theoretician as a tool builder. Around 2000 he implemented a symbolic version of the algorithms in MOPED, which was quite successful.

Ahmed: In 99 I moved to LIAFA in Paris, and I remember your kids were born.

Javier: Yes, you sent my wife beautiful flowers!

Ahmed: But we kept in touch, and we wrote a paper together in POPL'03 with my PhD student Tayssir Touili, now professor in Paris. We extended the ideas of the CONCUR paper to programs with both procedures and concurrency. Other papers came, the last in 2008.

Javier: And Ahmed is visiting Munich next year, pandemic permitting, so I hope there'll be more.

Nathalie: How would you say this award-winning paper influenced your later work? Did any of your subsequent research build explicitly on it?

Ahmed: This paper was the first of many I have co-authored on verification of infinite-state systems using automata. All of them use various automata classes to represent sets of configurations, and compute reachable configurations by iterative application of automata operations. We call these procedures accelerations; instead of computing a fixpoint of a function by repeated iteration, you "jump" to the fixpoint after finitely many steps, or at least converge faster. Accelerations were implicitly present in the CONCUR'97 paper. They have been also used by many other authors, for example Bernard Boigelot and Pierre Wolper.

My first paper on accelerations was with Peter Habermehl, my PhD student at the time and now at IRIF. We worked on the verification of systems communicating through queues, using finite automata with Presburger constraints as data structure. Then came several works on communicating systems with my student Aurore Annichini and Parosh Abdulla and Bengt Jonsson from Uppsala. As a natural continuation, with the Uppsala group and my student Tayssir Touili we developed the framework of Regular Model Checking. And then, with Peter Habermehl, Tomas Vojnar and Adam Rogalewicz from TU Brno, we extended Regular Model Checking to Abstract Regular Model Checking, which proved suitable and quite effective for the analysis of heap manipulating programs.

We also applied the CONCUR'97 results to the analysis of concurrent programs. The first work was a POPL'03 paper with Javier, Tayssir, and me on an abstraction framework. Two years later, Shaz Qadeer and Jacob Rehof proposed bounded-context switch analysis for bug detection. That paper created a line of research, and we contributed to it in many ways, together with Shaz, Mohamed Faouzi Atig, who was my student then, and is now Professor at Uppsala, and others.

Javier: The CONCUR'97 paper was very important for my career. As I said before, it directly led to MOPED, and later to jMOPED, a version of MOPED for Java programs developed by Stefan Schwoon and Dejvuth Suwimonteerabuth. Then, Tony Kucera, Richard Mayr, and I asked ourselves if it was possible to extend probabilistic verification to pushdown systems, and wrote some papers on the topic, the first in LICS'04. This was just the right moment, because at the same time Kousha Etessami and Mihalis Yannakakis started to write brilliant papers on recursive Markov chains, an equivalent model. The POPL'03 paper with Ahmed and Tayssir also came, and it triggered my work on Newtonian program analysis with two very talented PhD students, Stefan Kiefer, now in Oxford, and Michael Luttenberger, nw my colleague at TUM. So the CONCUR'97 paper was at the root of a large part of my work of the next 15 years.

Nathalie: Is there any result obtained by other researchers that builds on your work and that you like in particular or found surprising?

Javier: After implementing MOPED, Stefan worked with Tom Reps on an extension to weighted pushdown automata, the Weighted Pushdown Library. Tom and Somesh Jha also found beautiful applications to security. This was great work. I was also very impressed by the work of Luke Ong and his student Matthew Hague. In 97 Ahmed and I tried to apply the saturation method to the full mu-calculus but failed, we thought it couldn't be done. But first Thierry Cachat gave a saturation algorithm for Büchi pushdown games, then Luke, Matthew cracked the mu-calculus problem, and then they even extended it to higher-order pushdown automata, together with Arnaud Carayol, Oliver Serre, and others. That was really surprising.

Ahmed: I agree. I'd also mention Qadeer and Rehof's TACAS'05 paper. They built on our results to prove that context-bounded analysis of concurrent programs is decidable. They initiated a whole line of research.

Nathalie: What are the research topics that you find most interesting right now? Is there any specific problem in your current field of interest that you'd like to see solved?

Javier: Ten years ago I've had said the complexity of the reachability problem for Petri nets and of solving parity games, but now the first one is solved and the second almost solved! Now I don't have a specific problem, but in the last years I've been working on parameterised systems with an arbitrary number of agents, and many aspects of the theory are still very unsatisfactory. Automatically proving a mutual exclusion algorithm correct for a few processes was already routine 20 years ago, but proving it for an arbitrary number is still very much an open problem.

Ahmed: I think that invariant and procedure summary synthesis will remain hard and challenging problems that we need to investigate with new approaches and techniques. It is hard to discover the right level of abstraction at which the invariant must be expressed, which parts of the state are involved and how they are related. Of course the problem is unsolvable in general but finding good methodologies on how to tackle it depending on the class of programs is an important issue. I think that the recent emergence of data-driven approaches is promising. The problem is to develop well principled methods combining data-driven techniques and formal analysis that are efficient and that offer well understood guarantees.

Nathalie: Would you have an anecdote or a tip from a well-established researcher to share to PhD students and young researchers?

Javier: Getting this award reminded me of the conference dinner at CAV 12 in St. Petersburg. I ended up at a table with some young people I didn't know. The acoustics was pretty bad. When the CAV Award was being announced, somebody at the table asked "What's going on?", and somebody else answered "Not much, some senior guys getting some award". Never take yourself very seriously ...

Nathalie: Oded Maler passed away almost 3 years ago. Do you have any

memory of him to share with our readers?

Ahmed: Oded was very amused by the number of citations. He used to say "Look at all the damage we've done".

Javier: Yes, Oded had a wonderful sense of humour, very dry and deadpan. When I arrived in Grenoble it took me a few days to learn how to handle it! I miss it very much.

4 Interview with Rajeev Alur, Thomas A. Henzinger, Orna Kupferman and Moshe Y. Vardi

Luca: You receive the CONCUR 2021 Test-of-Time Award for your paper "Alternating Refinement Relations", which appeared at CONCUR 1998. In that article, you gave what I consider to be a fundamental contribution, namely the introduction of refinement relations for alternating transition systems. Could you briefly explain to our readers what alternating transition systems are? Could you also tell us how you came to study the question addressed in your award-winning article and why you focused on simulation- and trace-based refinement relations? Which of the results in your paper did you find most surprising or challenging?

AHKV: When we model a system by a graph, our model abstracts away some details of the system. In particular, even when systems are deterministic, states in the model may have several successors. The nondeterminism introduced in the model often corresponds to different actions taken by the system when it responds to different inputs from its environment. Indeed, a transition in a graph that models a composite system corresponds to a step of the system that may involve some components. Alternating transition systems (ATSs) enable us to model composite systems in more detail. In an ATS, each transition corresponds to a possible move in a game between the components, which are called agents. In each move of the game, all agents choose actions, and the successor state is deterministically determined by all actions. Consequently, ATSs can distinguish between collaborative and adversarial relationships among components in a composite system. For example, the environment is typically viewed adversarially, meaning that a component may be required to meet its specification no matter how the environment behaves.

In an earlier paper[1], some of us introduced ATSs and Alternating Temporal Logics, which can specify properties of agents in a composite system. The CONCUR 1998 paper provided refinement relations between ATSs which correspond to alternating temporal logics. Refinement is a central issue in a formal approach

[1] See `https://www.cis.upenn.edu/~alur/Jacm02.pdf`.

to the design and analysis of reactive systems. The relation "I refines S" intuitively means that system S has more behaviors than system I. It is useful to think about S being a specification and I an implementation. Now, if we consider a composite implementation $I||E$ and specification $S||E$ and we want to check that the component I refines the component S, then the traditional refinement preorders are inappropriate, as they allow I to achieve refinement of $I||E$ with respect to $S||E$ by constraining its environment E. Alternating refinement relations are defined with respect to ATSs that model the interaction among the underlying components, and they enable us to check, for example, that component I has fewer behaviors than component S no matter how component E behaves. They are called "alternating" because refinement may restrict implementation actions but must not restrict environment actions. In other words, refinement may admit fewer system actions but, at the same time, more environment actions.

It was nice to see how theoretical properties of preorders in the traditional setting are carried over to the game setting, and so are the results known then about the computational price of moving to a game setting. First, the efficiency of the local preorder of simulation with respect to the global preorder of trace containment is maintained. As in the traditional setting, alternating simulation can be checked in polynomial time, whereas alternating trace-containment is much more complex. Second, the branching vs. linear characterizations of the two preorders is preserved: alternating simulation implies alternating trace containment, and the logical characterization of simulation and trace-containment by CTL and LTL, respectively, is carried over to their alternating temporal logics counterparts. The doubly-exponential complexity of alternating trace containment, as opposed to the PSPACE complexity of trace containment, is nicely related to the doubly-exponential complexity of LTL synthesis, as opposed to its PSPACE model-checking complexity.

Luca: In your paper, you give logical characterisations of your alternating refinement relations in terms of fragments of alternating temporal logic. Logical characterisations of refinement relations are classic results in our field and I find them very satisfying. Since I teach a number of those results in my courses, I'd be interested in hearing how you would motivate their interest and usefulness to a student or a colleague. What would your "sales pitch" be?

AHKV: There is extensive research on the expressive power of different formalisms. Logical characterization of refinement relations tells us something about the distinguishing power of formalisms. For example, while the temporal logic CTL* is more expressive than the temporal logic CTL, the two logics have the same distinguishing power: if you have two systems and can distinguish between them with a CTL* formula (that is, your formula is satisfied only in one of the sys-

tems), then you should be able to distinguish between the two systems also with a CTL formula. Moreover, while CTL is not more expressive than LTL, we know that CTL is "more distinguishing" than LTL. These results have to do with the logical characterizations of trace containment and simulation. The distinguishing power of a specification formalism is useful when we compare systems, in particular an implementation and its abstraction: if we know that the properties we care about are specified in some formalism L, and our system refines the abstraction according to a refinement relation in which the satisfaction of specifications in L is preserved, then we can perform verification on the abstraction.

Luca: I am interested in how research collaborations start, as I like to recount "research-life stories" to PhD students and young researchers of all ages. Could you tell us how you started your collaboration on the award-winning paper?

AHKV: Subsets of us were already collaborating on other topics related to reactive models and model checking, and all of us shared a common belief that the field was in need to move from the limited setting of closed systems to a more general setting of open systems, that is, systems that interact with an environment. Open systems occur not only when the environment is fully or partly unknown, but also when a closed system is decomposed into multiple components, each of them representing an open system. To build "openness" into models and specifications as first-class citizens quickly leads to the game-theoretic (or "alternating") setting. It was this realization and the joint wish to provide a principled and systematic foundation for the modeling and verification of open systems which naturally led to this collaboration.

Luca: Did any of your subsequent research build explicitly on the results and the techniques you developed in your award-winning paper? Which of your subsequent results on alternating transition systems and their refinement relations do you like best? Is there any result obtained by other researchers that builds on your work and that you like in particular or found surprising?

AHKV: Various subsets of us pursued multiple research directions that developed the game-theoretic setting for modeling and verification further, and much remains to be done. Here are two examples. First, the game-theoretic setting and the alternating nature of inputs and outputs are now generally accepted as providing the proper semantic foundation for interface and contract formalisms for component-based design. Second, studying strategic behavior in multi-player games quickly leads to the importance of probabilistic behavior, say in the form of randomised decisions and strategies, of equilibria, when players have non-complementary objectives, and of auctions, when players need to spend resources

for decisions. All of these are still very active topics of research in computer-aided verification, and they also form a bridge to the algorithmic game theory community.

Luca: One can view your work as a bridge between concurrency theory and multi-agent systems. What impact do you think that your work has had on the multi-agent-system community? And what has our community learnt from the work done in the field of multi-agent systems? To your mind, what are the main differences and points of contact in the work done within those communities?

AHKV: Modeling interaction in multi-agent systems is of natural interest to planning problems studied in the AI community. In 2002, the International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS) was formed and the annual International Conference on Autonomous Agents and Multiagent Systems (AAMAS) was launched. The models, logics, and algorithms developed in the concurrency and formal methods communities have had a strong influence on research presented at AAMAS conferences over the past twenty years. Coincidentally, this year our paper on Alternating-Time Temporal Logic was chosen for the IFAAMAS Influential Paper Award.

Luca: What are the research topics that you find most interesting right now? Is there any specific problem in your current field of interest that you'd like to see solved?

AHKV: Research on formal verification and synthesis, including our paper, assumes that the model of the system is known. Over the last few years, reinforcement learning has emerged as a promising approach to the design of policies in scenarios where the model is not known and has to be learned by agents by exploration. This leads to an opportunity for research at the intersection of reactive synthesis and reinforcement learning. A potentially promising direction is to consider reinforcement learning for systems with multiple agents with both cooperative and adversarial interactions.

The realization that reactive systems have to satisfy their specifications in all environments has led to extensive research relating formal methods with game theory. Our paper added alternation to refinement relations. The transition from one to multiple players has been studied in computer science in several other contexts. For the basic problem of reachability in graphs, it amounts to moving from reachability to alternating reachability. We recently studied this shift in other fundamental graph problems, like the generation of weighted spanning trees, flows in networks, vertex covers, and more. In all these extensions, we consider a game between two players that take turns in jointly generating the outcome. One player aims at maximizing the value of the outcome (e.g., maximize the weight of the

spanning tree, the amount of flow that travels in the network, or the size of the vertex cover), whereas the second aims at minimizing the value. It is interesting to see how some fundamental properties of graph algorithms are lost in the alternating setting. For example, following a greedy strategy is not beneficial in alternating spanning trees, optimal strategies in alternating flow networks may use fractional flows, and while the vertex-cover problem is NP-complete, an optimal strategy for the maximizer player can be found in polynomial time. Many more questions in this setting are still open.

Luca: What advice would you give to a young researcher who is keen to start working on topics related to alternating transition systems and logics?

AHKV: One important piece of advice to young researchers is to question the orthodoxy. Sometimes it is necessary to learn everything that is known about a topic but then take a step back, look at the bigger picture, reexamine some of the fundamental assumptions behind the established ways of thinking, change the models that everyone has been using, and go beyond the incremental improvement of previous results. This is particularly true in formal methods, where no single model or approach fits everything. And young researchers stand a much better chance of having a really fresh new thought than those who have been at it for many years.

References

[1] R. Alur, T. A. Henzinger, O. Kupferman, and M. Y. Vardi. Alternating refinement relations. In D. Sangiorgi and R. de Simone, editors, *CONCUR '98: Concurrency Theory, 9th International Conference, Nice, France, September 8-11, 1998, Proceedings*, volume 1466 of *Lecture Notes in Computer Science*, pages 163–178. Springer, 1998.

[2] A. Bouajjani, J. Esparza, and O. Maler. Reachability analysis of pushdown automata: Application to model-checking. In A. W. Mazurkiewicz and J. Winkowski, editors, *CONCUR '97: Concurrency Theory, 8th International Conference, Warsaw, Poland, July 1-4, 1997, Proceedings*, volume 1243 of *Lecture Notes in Computer Science*, pages 135–150. Springer, 1997.

[3] D. Janin and I. Walukiewicz. On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic. In U. Montanari and V. Sassone, editors, *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*, pages 263–277. Springer, 1996.

[4] U. Nestmann and B. C. Pierce. Decoding choice encodings. In U. Montanari and V. Sassone, editors, *CONCUR '96, Concurrency Theory, 7th International Conference, Pisa, Italy, August 26-29, 1996, Proceedings*, volume 1119 of *Lecture Notes in Computer Science*, pages 179–194. Springer, 1996.

The Distributed Computing Column

BY

Seth Gilbert

National University of Singapore

seth.gilbert@comp.nus.edu.sg

BEATCS no 135

THE DISTRIBUTED COMPUTING COLUMN

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

In this issue of the distributed computing column, Marko Vukolić from Protocol Labs considers the future of decentralized blockchain systems. He argues for the importance of "inclusive" systems, like Bitcoin, where any user can participate. And at the same time, he takes the (perhaps contrarian) position that the issue of energy usage in Bitcoin is likely overstated, with the potential benefits outweighing the costs. Together, these points suggest that Bitcoin is well suited to become the base layer of a large-scale distributed distributed, and he lays out a blueprint for just such a vision. Enjoy this (perhaps provocative) new distributed computing column!

On the Future of Decentralized Computing

Marko Vukolić
*Protocol Labs**

Abstract

Decentralized systems (e.g., blockchain systems) have the potential to revolutionize financial and payment systems, as well as the internet — for the good of humankind and planet Earth. This position paper aims at justifying this standpoint and at laying out a vision for the future of decentralized computing.

We start by revisiting the definition of decentralized systems, briefly surveying the literature on the taxonomy and different facets of decentralization. We complement existing definitions by proposing *Inclusiveness* as a critical facet. We argue that our notion of Inclusiveness rules out some popular candidate technologies for a "base-level" (or L1) blockchain consensus, namely Proof-of-Stake, from replacing Nakamoto's Proof-of-Work (PoW) as the base consensus technology of decentralized systems.

We further discuss why the high energy consumption of Bitcoin's PoW consensus is not wasteful and why Bitcoin should be embraced as the money of the future. We then argue that future decentralized systems should aim at leveraging the "slow-but-very-secure" PoW consensus of Bitcoin, building systems on top of it rather than trying to replace it. Finally, we propose some open problems for decentralized cloud computing research.

1 Introduction

Decentralized systems are a subset of distributed systems. While a distributed system can loosely be defined as "a collection of independent computers that appears to its users as a single coherent system" [51], a basic definition of a decentralized system requires these independent computers to be controlled by multiple authorities, such that no authority is fully trusted by all [52]. In other words, authorities — or operators controlling computers of a system — are assumed to be potentially malicious, or *Byzantine* [36].

*The author is an independent computer scientist and the ConsensusLab Lead at Protocol Labs. Email: *marko@protocol.ai*.

This basic definition is, however, insufficient to capture all nuances of decentralization. For example, this definition is satisfied both by a global-scale open membership (i.e., permissionless) system, such as Bitcoin [43], and by a closed membership (i.e., permissioned) system comprising 4 companies implementing a Byzantine fault-tolerant protocol which tolerates one Byzantine fault. While it is intuitive that the latter system in our example is less decentralized than Bitcoin, it should also be obvious that we need a more fine grained methodology for evaluating the *level of decentralization*. To this end, the first contribution of this paper (Sec. 2) is a brief survey of the literature on taxonomy and different flavors of decentralization. We refine the basic definition of a decentralized system of [52] to identify four main decentralization facets of a distributed system: Resilience, Openness, In-Protocol Incentives and Governance.

This methodology will hopefully give the reader a tool to discern genuinely decentralized projects from others that only give an illusion of decentralization. This seems particularly important given an explosion in the number of "decentralized" cryptocurrency projects. For instance, `coinmarketcap.com` lists more than 12'000 cryptocurrencies, many of which lack a concrete use case and have their token supply and network governance controlled by their respective development teams. These projects piggyback on the rising popularity of tokens that have a genuine use case (e.g., Bitcoin) and can sow confusion and create speculative bubbles.

A minority of these projects are actually motivated by improving the state of the art in decentralized computing. For instance, some projects aim to address performance limitations of Bitcoin (in particular its transaction throughput, currently capped at about 7 transactions per second [53]) or reduce its power consumption. These projects are complicated by fundamental tradeoffs that underlie the design of decentralized systems, in particular the tradeoff between scalability and performance and its very level of decentralization. In other words, it is not easy to scale Bitcoin while retaining its security and decentralization.

In these attempts to improve Bitcoin, there seems to be a confusion about its actual use case. If Bitcoin is seen as a simple payment system, its performance indeed could not answer the demands of the slightly less than 8 billion people on planet Earth. However, Bitcoin network could be used as a final settlement layer while the scaling could happen in a hierarchical approach, in so-called layer 2 [29] and higher-layer protocols.

This leaves a seemingly insurmountable issue related to Bitcoin's power consumption, which today already uses roughly 0.1% of the world's energy production [4]. Proposals aiming at addressing this issue involve changing the base consensus protocol of Bitcoin from Nakamoto's Proof-of-Work [43] to an alternative one. A number of top-ranked cryptocurrencies use or plan to use the approach called *Proof-of-Stake* [2] to be more "green" and energy efficient than Bitcoin.

Unfortunately, Proof-of-Stake is not as open and decentralized as Proof-of-Work and is actually more akin, in its essence, to a closed membership (permissioned) system. While this is intuited in open discussions since the proposal of the Proof-of-Stake idea on the bitcointalk forum in 2011 [2], existing definitions of decentralized systems fail to capture this key difference between Proof-of-Work and Proof-of-Stake. To this end, we propose a new property called *Inclusiveness* (Sec. 3), which refines Openness as previously considered in the literature (and which we cover in Sec. 2). In short, an inclusive system designates a decentralized system which provides *equal opportunities* to its participants. Inclusive systems are a subset of open (permissionless) systems: we show that Nakamoto's Proof-of-Work is inclusive and that systems based on Proof-of-Stake are not. Therefore, Proof-of-Stake cannot be used in Layer 1 in inclusive decentralized systems.

While Bitcoin is inclusive, if it is seen as a simple payment system its energy consumption would indeed be too high a price to pay for this property. However, energy usage or, more generally, the cost of a certain technology, should always be evaluated in the context. We propose to re-evaluate Bitcoin's energy consumption considering a different use case for Bitcoin: that of inclusive decentralized *money* (i.e., "peer-to-peer cash" [43]), of the present and, especially, the future.

If one performs a thought experiment of what would happen if (when) Bitcoin becomes the dominant form of money on Earth, the fears of "excessive", "wasteful" and "useless" power consumption of Bitcoin fade away. We will perform exactly such a thought experiment later in this paper (Sec. 4), thus explaining why Bitcoin's power consumption is not wasteful or excessive, and why it is actually good for planet Earth and mankind.

In short, in Section 4 we make an argument that human behavior in the Bitcoin monetary system is incentivized towards savings and rational spending on things we *need*, with low time preference, encouraging long-term planning, preservation of natural resources and sustainability. We contrast this to the current inflationary fiat monetary system, which inherently promotes spending on things we (think we) *want* and consumerism, high time preference (i.e., focus on short-term profits), and where the entire economy is oriented to "growth", which results in producing and consuming things we often do not need, wasting resources. We will also touch upon the equal opportunities in the Bitcoin monetary system and their impact on human freedom and liberties, contrasting them to inequalities in the current monetary system.

Once we agree that it is good for humankind that the Bitcoin network acts as a backbone for future money and once we embrace this thought, it is interesting to explore how we can use such a very secure decentralized network in use cases beyond money. Some interesting projects pursue use cases different from the use case Bitcoin pursues, e.g., offering decentralized storage of large volumes of data (e.g., Filecoin [6] network which today has more than 10 exabytes (EB), or 10

million TB, of storage capacity). Other projects work towards enabling arbitrary computation in a decentralized network. These *decentralized cloud computing* projects have the potential to, one day, challenge the centralized cloud computing operators that dominate today's internet [27]. Since these systems need to eventually scale to the workloads of today's centralized cloud computing and beyond, they clearly need more efficient consensus hierarchies than the one offered solely by Bitcoin's Proof-of-Work. However, this brings back the decentralization and security challenges of consensus protocols other than Proof-of-Work. The key question is: *can we build more efficient decentralized systems that would benefit from the security of Bitcoin*?

To this end, we propose (in Section 5) to approach the design of the future decentralized internet by leveraging Nakamoto's Proof-of-Work as a secure anchor for critical information needed for secure operation of other, more scalable networks. In a sense, we propose to use Bitcoin network as the backbone of the decentralized internet, helping to secure the rest of it. As an example of such use, we discuss a possible approach in which weights of validators in a Proof-of-Stake network, potentially along with its state checkpoints, are anchored into the Bitcoin blockchain, addressing the critical family of so-called *long-range* attacks on Proof-of-Stake [21]. In this design, to complement Bitcoin as a secure store of state/membership anchors, i.e., hashes of the critical state, we propose optionally using a decentralized content addressable storage system, such as Filecoin/IPFS [15] to resolve those hashes. Finally, we outline some open problems motivating future work.

2 Methodology for Evaluating Decentralization in Distributed Systems

In general, *decentralized systems* can be defined as *a subset of distributed systems where multiple authorities control different components and no authority is fully trusted by all [52].*

For instance, popular cloud and social networks like Google, Facebook or Twitter, are examples of distributed systems. However, these systems are not decentralized, as each one is controlled by a single authority (company). Note that it is not sufficient for a system to simply have its components controlled by multiple authorities in order to be classified as decentralized — the absence of a single trusted authority is needed, meaning that *any component in a decentralized system can be potentially adversarial [52]*, or *Byzantine* [36].

Beyond the above broad definition of a decentralized system, computer science literature considers multiple *facets* of decentralization in an attempt to char-

acterize its nuances, as well as the differences among decentralized systems (see e.g., [44] for a recent survey). We summarize these into the following *decentralization facets*:

1. **Resilience** of the system to adversarial (Byzantine) behavior of its components, i.e., the authorities that control them, as well as the simple disappearance (also called unavailability) of individual components.

   Resilience itself may apply to different properties of the system, namely *safety* and *liveness* [35, 10]. Informally, a safety property of a system stipulates that "bad things" do not happen and a liveness property stipulates that "good things" do eventually happen (i.e., that the system does not stop making progress).

   For instance, an important liveness property of a blockchain system is *censorship* resistance [26], whereas an important safety property of a blockchain system is *double-spend* resistance [43]. We define these properties later, in the context of Bitcoin, in Section 4.1.

   To quantify Resilience, the scientific literature and engineering practice is typically interested in the minimum number of authorities that the adversary needs to compromise to subvert a key property of the system, such as safety or liveness. In the context of blockchains this number is sometimes referred to as the *Nakamoto coefficient*[1] [48]. Intuitively, the higher the Nakamoto coefficient, the higher the level of decentralization. Per the definition of a decentralized system we adopted [52], the system cannot be deemed decentralized if this number is 1 — i.e., if a single participating authority can compromise a key property of the system. Finally, when evaluating the Nakamoto coefficient, it is important to consider possible business relations or shared control structures among otherwise seemingly independent authorities.

2. **Openness** of the system to new participants. In this sense, a widely-used classification of blockchain systems into *permissioned* and *permissionless* systems (see e.g., [40]) reflects this property. Permissionless systems allow participants to self-elect into the system, whereas permissioned systems rely on an external selection process to be admitted into the system — with the authority to choose [participants] typically residing with an institutional or organizational process [40]. In other words, permissionless systems are *open* to any new participant, whereas permissioned systems are not. Some authors define decentralized systems as only those *in which anyone is able*

[1] Honoring Bitcoin's pseudonymous inventor, Satoshi Nakamoto.

to participate [12], effectively restricting the notion of decentralized systems only to open, permissionless systems. As a general principle, even if we accept permissioned systems as decentralized ones, permissionless systems are to be considered more decentralized than permissioned systems.

Some authors further refine the notion of open, permissionless systems focusing on equality of participants within the system. Karakostas et al. [31] define *egalitarianism* in a rather technically involved way aiming at capturing the proportionality of rewards of participants in blockchains compared to their investment. In a related approach, Fanti et al. [24] define *equitability*, which quantifies how much a participant can amplify her token holdings compared to her initial investment. In the next section (Section 3), we argue that these refinements of Openness are not general enough and define a new refinement of Openness, using the notion of *Inclusiveness*.

Finally, some authors recognize *operational decentralization* as a facet of decentralization related to Openness [44]. Intuitively, operational decentralization aims at capturing hardware requirements for participation in the system — the smaller the hardware requirements, the higher the possibility for anyone to participate in the system and, hence, the higher the level of decentralization. For instance, a system which requires large amounts of storage (e.g., hard disk space) to participate in blockchain A would be deemed more centralized than blockchain B which requires less storage space [44].

3. **In-protocol Incentives** refer to the existence of rewards for protocol participants, paid out to protocol participants in the protocol's *native token*. Incentives are an important facet of decentralized systems [44]: Troncoso et al. [52] argue that the development of adequate incentives is necessary to build a successful decentralized system.

   In general, In-protocol Incentives test the Openness of the system. On the one hand, an open system that provides incentives for participants will attract new participants. On the other hand, a seemingly open system that does not provide In-Protocol Incentives effectively limits its Openness, as new participants have less economic rationale to join the system. Such a system may resort to out-of-protocol incentives, in which case incentives are not governed by system software but by people. Out-of-protocol incentives may involve existing participants establishing business and contractual relations with new participants to motivate them to join the system. This approach resembles and is more common in permissioned networks [11], which, as we discussed, do not satisfy Openness.

   In the context of incentives, wealth distribution across token stakeholders is also considered as an aspect of decentralization [44].

4. **Governance** of the system, focusing on power of human stakeholders to influence and change key rules in the system, e.g., through software updates.

   Several parameters for evaluating the decentralization of governance power have been proposed or discussed in the literature. These include:

   (a) *governance of the infrastructure* [25], or *improvement control* [44], often involving the number of developers contributing to systems' codebase and the number of people contributing to the discussion around the system design [12],

   (b) *existence of a public face* [25], which can be defined as a personality and/or institution that is widely recognized as a spokesperson or a representative of the system.

   (c) *owner control*, measured by examining the total tokens accumulated by the stakeholders in the early adoption period. Depending on the consensus mechanism used, such early tokens may give more power to their owners, causing inequalities and centralization — this is particularly relevant in Proof-of-Stake systems [44].

Finally, some authors [44] consider additional facets of decentralization, including the decentralization at the *network layer*, i.e., pertaining to the decentralization of the network that underlies a distributed system, and the decentralization at the *application layer*, which includes, e.g., the diversity of wallets and exchanges. We acknowledge these decentralization facets that go beyond the core of the system itself, opting to focus on systems proper in this position paper.

3 Inclusiveness in Decentralized Systems

In this section, we define *Inclusiveness*, which refines the notion of Openness defined in the previous section. We argue for Inclusiveness as a key property of decentralized systems and show that Proof-of-Stake systems are not inclusive, in contrast to Proof-of-Work systems.

Inclusiveness is somewhat similar to the notions of *egalitarianism* [31] and *equitability* [24]. Compared to Inclusiveness, these notions are less general as they are defined only for protocols with incentives, practically quantifying the linearity of reward distribution compared to the investment made.

Towards defining *Inclusive* systems, we first define the notion of *Equal Opportunities*.

Definition 1 (Equal Opportunities)**.** *A decentralized system provides* Equal Opportunities *if it satisfies both of the following conditions:*

- *(Resource Symmetry) The system allows any (new or existing) participant Bob to have an equal role in the system as any other existing participant Alice, provided Bob makes the same investment in system resources as Alice. Specifically, this means that if we swap the identities (private/public key pairs) of participants Alice and Bob, the resulting system should be indistinguishable from the original system.*

- *(Genuine Openness) The system cannot reach a state in which it prevents Bob from making such an investment. Specifically, Bob's ability to make this investment must never depend on the permission or actions of either Alice or other participants in the system.*

The first condition of the Equal Opportunities property aims at capturing *resource symmetry*, intuitively capturing equality among new and existing network participants. The motivation behind resource symmetry is to measure if the system gives participants equal power in the system (given that their investment is the same), or if it makes some participants "more equal" than the others, e.g., based on discriminating their identities.

For example, two miners in Bitcoin's Proof-of-Work have an equal role and expected rewards in the system if they contribute the same computing (hashing) power to the system (i.e., if they make the same investment in system resources). On the other hand, swapping identities of a participant Alice, who is part of a permissioned system, and Bob, who is not, yields a system which can be distinguished from the original one. In other words, permissioned systems are not resource symmetric. Moreover, not all permissionless systems are resource symmetric.

The second property of Equal Opportunities aims at refining the Openness property. In principle, an open (permissionless) system could be resource symmetric but prevent, in some state, new participants from making an investment in system resources that would allow them to match the investment of existing participants. Arguably, such a system could not be deemed as *genuinely open*.

Finally, we define *Inclusive* decentralized systems.

Definition 2 (Inclusiveness)**.** *A decentralized system is called* Inclusive *if and only if it satisfies Equal Opportunities.*

It is easy to see that inclusive systems are a subset of open (i.e., permissionless) systems. However, not all open systems are inclusive.

Proof-of-Stake and Proof-of-Work permissionless consensus protocols have fundamentally different implications on the decentralization of the network, which are captured by Genuine Openness. In the following, we show that Proof-of-Stake systems do not satisfy this aspect of Equal Opportunities and, hence, are

not Inclusive. We also provide a high-level argument that Proof-of-Work systems satisfy Inclusiveness.

In short, in Proof-of-Stake, "miners" do not expend electrical energy for mining but vote with power proportional to the size of their stake, i.e., holdings in the native token dedicated to voting. This not only implies considerably different economical dynamics compared to Proof-of-Work [24], but may outright lead to violation of Equal Opportunities.

To see this, consider the following simple example of a non-inflationary Proof-of-Stake system, i.e., the one with the non-increasing total supply of a token. If existing miners control more than 50% of the stake in the network and are unwilling to sell their stake to new participants, new participants can never reach the stake of old miners, regardless of the size of their investment. This violates the Genuine Openness aspect of Equal Opportunities and, consequently, Inclusiveness. In this sense, the fact that in Proof-of-Stake existing participants can possibly prevent new participants from meaningfully joining the system evokes similarities between permissioned and Proof-of-Stake permissionless systems.

On the other hand, Bitcoin's Proof-of-Work satisfies Genuine Openness. Namely, the nature of Proof-of-Work consensus (see Sec. 4.1 for details) does not prevent any participant from making an investment into system resources. In particular, and assuming a free market for computing power, as well as absence of scarcity of computing power and energy, existing participants cannot prevent new participants from entering the system. With innovation in computing (Moore's law), the computing power of the existing participants actually decays in time compared to the computing power available outside the system, which is free to join the network.

Therefore, we conclude that Bitcoin is an Inclusive system, whereas (non-inflationary) Proof-of-Stake based systems are not. We leave as open the following crypto-economics problem: are there variants of Proof-of-Stake which provably satisfy Equal Opportunities?

4 Why Bitcoin Does Not Waste Energy

In the previous section, we argued that the systems based on Proof-of-Stake are not inclusive, as opposed to those based on Proof-of-Work. Therefore, assuming that we accept Inclusiveness as a necessary aspect of decentralization, it follows that Proof-of-Stake blockchain systems cannot be used as the basis for decentralized systems (i.e., for the so-called layer 1 (L1)).

While Bitcoin and its Proof-of-Work could, technically, be used as the layer 1 of decentralized systems, the seemingly insurmountable issue of its "excessive" energy consumption remains. We propose an argument as to why this is a non-

issue and support the claim that the energy consumption of Bitcoin is actually good for humankind and planet Earth, arguing that it is neither wasteful nor excessive. We leave formal modeling and proofs of this argument to future work and the future itself.

Towards making such an argument we need to depart from the narrow domains of computer science and engineering.[2] To help see the "big picture", we reason about Bitcoin from the angle of other sciences such as sociology, economics and philosophy. We believe that this is, in itself, thought-provoking, as it brings to the spotlight the multi-disciplinary implications of Bitcoin, revealing its intrinsic beauty and ingeniousness. The main impact of the novelty of Bitcoin is to be measured in the spheres of socio-economics and metaphysics, not in computer science.

To this end, later in this section (Sec. 4.3), we will perform a thought experiment in which we will discuss the properties of a prospective world in which Bitcoin becomes the pre-dominant money for mankind, or *unit of account*. For the sake of simplicity, we will perform the thought experiment assuming that Bitcoin becomes the *only* currency in use — we believe that most of our conclusions would hold even if alternative currencies continue to exist, so long as Bitcoin becomes the unit of account. Then, we come back to discussing energy expenditure of Bitcoin (Sec. 4.4), provoking the reader to reconsider if this energy expenditure is a fair price to pay for living in such a world.

In the course of the thought experiment, our economics arguments will mostly be made using simple logical thinking, based on infinite vs. capped money supply, in an attempt to address a large audience. However, for readers who prefer a more structured scholarly approach, where appropriate we will refer to and echo the economics views of the so-called Austrian School of economics, and in particular the thoughts of Friedrich A. Hayek, the 1974 Nobel Memorial Prize laureate.[3]

Before this, in order to make this paper self-contained, we briefly present, in Section 4.1, the background behind Bitcoin and briefly evaluate its decentralization (Sec. 4.2) using the methodology of Sections 2 and 3. A reader familiar with Bitcoin may skip the next section, whereas a reader unfamiliar with Bitcoin is encouraged to also read Nakamoto's original whitepaper [43].

[2] Provided we do not invoke the Simulation Argument [19] to remain in the realm of computer science.

[3] Specifically, Hayek's "The Constitution of Liberty: The Definitive Edition" [30], first published in 1960.

4.1 Background on Bitcoin

4.1.1 Bitcoin Basics: Use Case and Monetary Policy

Bitcoin [43] is an open-source peer-to-peer computer network for generating and transferring Bitcoin's native token (bitcoin or "BTC") among the users (peers) of the network. Bitcoin was conceived as an electronic cash network to allow online payments to be sent directly from one party to another without going through a financial institution or any other trusted middleman. This was not possible prior to Bitcoin as all electronic payments required trusted intermediaries, unlike physical, in-person, cash or barter transactions.

On a high-level, in Bitcoin, user Alice wishing to send 1 BTC to another user Bob, digitally signs, using her private cryptographic key, a transaction to transfer 1 BTC from an *address A*, that Alice controls, to *address B* supplied to Alice by user Bob. Alice's private key is cryptographically tied to *address A* (which is basically a cryptographic hash of a corresponding public key). Knowledge of the private key allows Alice to have control over her BTC. As a fundamental principle, whoever controls the private keys corresponding to a given address controls bitcoin pertaining to that address.

The main challenge in such a system is that users do not trust each other. Namely, Alice could attempt to *double-spend* her bitcoin. Consider the following example of a double-spend attempt. Alice signs transaction $tx_{Alice-to-Bob}$ in which she transfers 1 BTC from address A she controls, to Bob's address B. However, she also signs a conflicting transaction $tx_{Alice-to-Alice}$ in which she sends 1 BTC from address A to another address A' that Alice also controls.

Which of these conflicting transactions should be actually taken into account? This is the main technical problem that Bitcoin solves. In a process called *consensus*, peers in the Bitcoin network, without trusting each other, agree on the global, totally ordered *ledger* of all transactions in the system.

In our example, all peers in the Bitcoin network would agree on the relative order between the two conflicting transactions $tx_{Alice-to-Bob}$ and $tx_{Alice-to-Alice}$. The first transaction in that order would be considered valid, whereas the other would be discarded.

Besides preventing double-spends (as a safety property), another important property Bitcoin provides is censorship resistance (as a liveness property). In short, censorship-resistance guarantees that a correctly-behaving user Alice will have her transactions eventually included in the ledger (while possibly having Alice pay a *transaction fee* for this service). In other words, censorship-resilience guarantees that transactions will not be excluded from the Bitcoin ledger due to actions of the Byzantine adversary or peers disappearing from the system.

For efficiency reasons, Bitcoin processes transactions in blocks, grouping a

number of transactions together, up to a certain maximum block size. Effectively, the Bitcoin consensus mechanism establishes a global order on those blocks forming a *chain* of blocks (i.e., a "blockchain"). Consequently, Bitcoin establishes global order on the transactions contained in those blocks.

Bitcoin software defines a so-called *genesis* block, the first block in the chain, to which the latter blocks are appended. The Bitcoin genesis block contains a link to the "real" (physical) world, by embedding the headline of the cover page of *The Times* (British daily national newspaper) from January 3rd, 2009 reading *"Chancellor on Brink of Second Bailout for Banks"*. This link to the real world, beyond possibly conveying a motivation for the existence of Bitcoin, is important for proving that the creator of the Bitcoin network could not have ran the code prior to January 3, 2009.

At the beginning of the Bitcoin blockchain history, there weren't any bitcoin to transact, as none had been brought into existence (i.e, *minted* or *mined*) yet. To bring bitcoin into existence, Bitcoin software defines a *block reward*, which is an incentive for network participants to take part in Bitcoin consensus. Bitcoin rewards every participant who successfully adds a block to the blockchain with a fixed reward, which halves every 210,000 blocks. The period of 210,000 blocks corresponds roughly to 4 years, as Bitcoin block production time is set to self-adjust to 10 minutes per block on expectation. For the first 210,000 blocks, the block reward was 50 BTC per block. With maximum supply, as stipulated by Bitcoin code, being 21 million BTC, 50% of all bitcoin have been mined in the first 210,000 blocks. With block reward halving to 25 BTC, from block 210,001 to block 420,000, an additional 25% of the total supply have been minted in that period, and so on, with the current Bitcoin block reward conveniently conveying which percentage of the total supply has been minted within the current 4-year window. Currently, more than 12 years after the genesis block, the Bitcoin network has produced over 700,000 blocks with the current block reward being 6.25 BTC.[4] The last 10% of Bitcoin's total supply is to be mined between today and the year 2140.

A participant in the Bitcoin network is an entity that runs a *full node*. Each Bitcoin full node keeps the entire history of the blockchain, validates new blocks and (optionally) participates in creating new blocks. Bitcoin's maximum block size and a relatively conservative time period interval of 10 minutes between blocks imply that the blockchain does not grow too fast compared to advances in computer hardware. Users can opt-out from running full nodes, by maintaining only *client* wallets, which protect their private keys and send Bitcoin transactions to others' (full) nodes.

In the following, we explain how blocks are generated and validated in the

[4] Each bitcoin is divisible into 100 million smaller units, usually called satoshis.

Bitcoin consensus.

4.1.2 Bitcoin Consensus

Bitcoin consensus proceeds as follows [43]:

1. New transactions are broadcast to all nodes.

2. Each node collects new transactions into a block. A node cryptographically links the new block to its predecessor (parent) block. These parent links define the position of the new block in the blockchain, all the way to the genesis block. In short, a node chooses the predecessor block for the new block to be the one which has the longest chain[5] to the genesis block, out of all blocks known to a node. In principle, nodes consider the transaction history defined by the longest chain as the only valid one.

3. In the process often called *mining*, or *Proof-of-Work* [43], each node works on finding a final piece of information, called a *nonce* that, when embedded into the new block, will make other nodes accept and declare the new block as *valid*.

   This is the key point in the otherwise relatively straightforward Bitcoin consensus. Namely, Bitcoin requires a cryptographic hash of a valid block to start with a number of zeros (0s) when represented as a bit string. Since the output of a cryptographic hash function cannot effectively be predicted, a hash of a block with one specific nonce appears basically as a random string of 0s and 1s. Therefore, nodes need to try many nonces in order to be lucky and construct the required final data for the block such that the hash of the block will start with many 0s, as required by the validation code. The number of zeros required is self-adjusted by the network during its lifetime, based on code, to maintain an expected block time of 10 minutes between the blocks.

   Finding a nonce which makes the block valid is effectively a very simple but computationally intensive guessing game in which a node repeatedly tries different nonces, applies them to the rest of the block, applies the hash function and sees if the output hash has the required number of leading zeros.

4. When a node finds a nonce and completes the Proof-of-Work, it broadcasts the block to all nodes.

[5] In fact, it is the chain which requires most work, which is most often the longest chain. For simplicity of narrative, we talk about "longest chain".

5. Other nodes accept the block only if: *(i)* all transactions in it are valid and do not contain already spent bitcoin, and *(ii)* the hash of the block starts with the required number of 0s. Unlike the mining step (step 3), this validation step (step 5) is very simple and cheap to compute.

In the recent months, the Bitcoin network as a whole is estimated to have performed anywhere between 68 EH/s (exahashes per second) on June 28, 2021 and 190 EH/s (on May 9, 2021). An exahash per second is one quintillion (a billion billion) hashes per second, a very large number of operations. With each hash operation costing actual physical-world energy to compute, this results in large power consumption of the Bitcoin network, which is sometimes frowned upon and considered as wasted.

4.2 Bitcoin's Decentralization

In evaluating Bitcoin's Resilience, we look at two major possible issues: the double-spending (safety) issue and the censorship of transactions (liveness) issue. To mount these attacks effectively on the Bitcoin network, the attacker needs to control more than 50% of the network computing power. This would allow the attacker to simply ignore blocks produced by the rest of the network and produce the dominant longest chain, which would then, by Step 2 of the Bitcoin consensus protocol (Sec. 4.1.2) be the effective history of transactions. In the case of censorship attacks - this new history could simply be empty of transactions. This is known as a 51% attack for Bitcoin and requires a majority of the hash power of the network.

Whereas it is difficult to precisely calculate the Nakomoto coefficient (number of different authorities required to mount the attack) for Bitcoin, this resilience can be (very) conservatively estimated. Namely, to spread out more evenly, in time, their earnings from block rewards, Bitcoin nodes often group into so-called *mining pools*. While individual nodes are often not directly under the control of a mining pool's operator authority and could leave the mining pool if they detected that they were participating in an attack, for a *very conservative* estimate of Resilience one could theoretically assume that a mining pool fully controls all the nodes within. With this in mind, at the time of writing, more than 50% of Bitcoin mining power is controlled by 4 mining pools.

However, in practice, and as indicated in the Bitcoin whitepaper [43], the economic incentives of Bitcoin make safety attacks towards compromising Resilience less likely than if the In-Protocol Incentives did not exist. If certain nodes control a large amount of computing power, they have an economic dilemma between using that power to attack the system or using that power to behave correctly and earn block rewards and transaction fees. This intuitively contributes to effectively

increasing the Nakamoto coefficient (Resilience measure) and consequently increasing the decentralization level of the network, in presence of economically rational participants.

As for other facets of decentralization we defined earlier in the paper, Bitcoin is Inclusive (Sec. 3) and has in-Protocol Incentives (Sec. 4.1).

It also has excellent operational decentralization, i.e., has low hardware requirements for running a full node. Today, the size of the Bitcoin blockchain is about 400GB of data, which means that a full node can be easily run on low-cost hardware, with a mid-sized hard-disk (or a larger microSD card) and internet connection, basically by anyone.[6] This last feature is an important decentralization aspect, and has many benefits. We will illustrate one such benefit later, in Section 5.2, outlining a design that requires a user of a Proof-of-Stake blockchain to run a Bitcoin node to prevent a number of critical attacks on Proof-of-Stake. If running a Bitcoin full node would have had high hardware requirements, this approach would be unrealistic.

Low hardware requirements for running a full node do not imply that everyone can be successful in mining. Full nodes are incentivized to invest more into hardware and computing equipment if they wish to have a higher probability of obtaining block rewards in the Bitcoin consensus. It is well known that economically viable Bitcoin mining requires larger investments, with large miners even running datacenter-size operations. Here, we should not confuse equal opportunities and inclusiveness with linear payouts, i.e., rewards proportional to an investment. Like practically any other economic undertaking, Bitcoin mining benefits from economies of scale. This does not undermine its inclusiveness.

Finally, Bitcoin's Governance benefits from the absence of any single individual or company acting as its public face (as Satoshi Nakamoto disappeared from the public discourse more than ten years ago). Regarding owner control, Bitcoin did not have a hidden owner accumulation phase. The first transaction in the Bitcoin network happened in block #170, seemingly between Satoshi Nakamoto and a cryptographer Hal Finney, on January 12, 2009, nine days after The Times newspaper timestamp contained in the genesis block. The first block following the genesis block was mined, probably by Satoshi Nakamoto, six days after the genesis block, on January 9, 2009.[7]

Concerning code improvement proposals, anyone can propose a change to

[6] Bitcoin full node can be run on hardware which today costs about $200 USD, see `https://getumbrel.com`.

[7] As it is widely believed, Satoshi Nakamoto may have mined a sizeable number of bitcoin in the early days of the network following the genesis, as an early participant. The exact number is practically impossible to support with hard evidence. However, we do have hard evidence, in the very Bitcoin transaction history, that an overwhelming majority of those early bitcoin that could be attributed to Satoshi Nakamoto were never transacted on the network.

the Bitcoin open-source software via Bitcoin Improvement Proposals (BIPs).[8] In practice, relatively few "core" developers (developers of the Bitcoin Core reference node software) propose and implement changes [12]. Major changes to software are relatively rare, with no BIP containing a backwards incompatible change to Bitcoin consensus (also known as a hard fork) ever having been deployed in the software. For changes that restrict consensus validation rules even further and are backwards compatible (soft forks), consensus among core developers is required, together with approval of miners through on-chain voting. That said, as Bitcoin is open-source software, anyone can make any change to the software. Whoever makes such a change, changing e.g., Bitcoin parameters (block size, or frequency, or token supply), has "only" to convince other users to migrate to using such a network. A number of such backwards incompatible changes to Bitcoin code have resulted in Bitcoin network forks (examples include Bitcoin Cash and Bitcoin Gold), all considerably less popular than Bitcoin.

4.3 Life on Earth with Bitcoin as Money and Unit-of-Account: a Thought Experiment

Towards our thought experiment, we first review the incentives for participants (people and businesses) in the current monetary system (Sec. 4.3.1) and then move to incentives for people in the Bitcoin system (Sec. 4.3.2). We then consider the impact of Bitcoin on economic inequalities in Section 4.3.3.

4.3.1 Human Incentives under the Current Fiat Monetary System

As Hayek put it in 1960, *"With government in control of monetary policy, the chief threat in this field has become inflation."* [30]. Hayek made this statement even before we completely abandoned the gold standard in 1971 and transitioned to so-called *fiat* money (not backed by anything), and well before progressive reduction and subsequent elimination (in some jurisdictions) of required reserves for commercial banks when making loans in fiat money.

In such an inflationary economic environment, with continuous increase in monetary supply there are several issues, out of which we outline just a few:

- Savings are silently taxed by inflation, therefore incentivizing people *to spend* fiat money quickly, or to invest it.
- Investments are done through e.g., stocks, by entrusting funds to other economical players, i.e., businesses. The success of these businesses is measured by their economic "growth", where this "growth" is measured again

[8] `https://github.com/bitcoin/bips`

in the same inflationary fiat currency. This incentivizes businesses to promote spending, which business predominantly do by relying on marketing and ads to push products to people in an attempt to make them spend more.

- Consequently, the *time preference* of both individuals and businesses is high (i.e., they are *short-term* oriented). Individuals are incentivized to spend money quickly, whereas businesses are incentivized to "grow" by pushing products to people, even if these are not *needed* by people. Marketing divisions exist to *create the demand* for products.

Therefore, current inflationary fiat economy encourages and incentivizes spending of resources on otherwise unneeded products, for which demand is simply created to fulfill the goal of selling more goods. It is not difficult to see that, this being the predominant economic model for 8 billion people, is unsustainable and will lead, in the long term, to overexploitation and pollution of the planet, without necessarily improving quality of life. The current economic model lacks incentives for long-term considerations.

This system has other profound issues related to the structure of the monetary system and positioning of the preferred players (e.g., banks) in such a system.[9] As inflation creates more money supply, these preferred players are close to the source of money (i.e., to central banks), creating the so-called Cantillon effect[10] and inequalities in money distribution. These inequalities have profound implications on the very freedom and liberty of people who are at the opposite side of this inequality.

4.3.2 Human Incentives under the Bitcoin Monetary System

The idea of using *"rules versus authorities in monetary policy"* has been argued since, at least, Henry Simons [47] in 1936. As Hayek writes, *"arguments advanced [by Simons] in favor of strict rules are so strong that the issue is now largely on of how far it is practically possible to tie down monetary authority by appropriate rules"* [30].

Bitcoin, for the first time, offers a monetary network with such "strict rules", which cannot be changed or manipulated even by the strongest adversaries. Prior to Bitcoin we, as a humanity, did not have such a technology. Therefore, it is pardonable that we resorted to inferior solutions, including the current fiat monetary

[9] Stretching our decentralization terminology of Sections 2 and 3, the fiat monetary system does not provide Openness and, consequently, is not Inclusive.

[10] Richard Cantillon, an 18th century economist, suggested that inflation occurs gradually, originating the concept of non-neutral money, positing that the original recipients of new money enjoy higher standards of living at the expense of later recipients. [8]

system. However, since we now have this technology, let's explore what kind of the world it promises.

With the total money supply of 2.1 quadrillion monetary units (21M BTC $\times$ 100M satoshis/BTC), Bitcoin is a monetary system with hard-capped money supply. This cap cannot be changed in an undetectable way: a single independent full node running Bitcoin software needs not to trust anyone to be able to verify that the supply did not change. The same goes for the Bitcoin's rate of increase in money supply.

In the current adoption period, Bitcoin encourages *long-term savings*. This argument can easily be made without resorting to the historical exchange rate with respect to legacy fiat currencies (although this historical exchange rate can be used to verify the argument). Namely, as adoption grows and more people accept Bitcoin, its value grows, along with its network effect.

While people save and hold (colloquially, *hodl* [3]) Bitcoin, especially for the long term, they change their time-preference and move away from the spending economy and its incentives (Sec. 4.3.1) towards the mindset and an economy which encourages savings and, consequently, preservation of natural resources. While doing this, people have an option to retain financial sovereignty (which can be directly related to personal freedom and liberties [30]) by securing their private keys themselves. It is worth noting that this aspect of Bitcoin still requires a considerable level of technological literacy.

Fast forward to the future in which Bitcoin is the only money for humankind. People do spend their bitcoin,[11] but primarily on things they *need*, as they know the Bitcoin are scarce. It is certainly possible to earn more Bitcoin by working, yet this will entail providing genuine value to other people, in order for the latter to be ready to spend their scarce monetary asset.

This marks a shift from a spending economy in which depreciating money is spent on things we (think we) want but often do not need, to a saving economy of appreciating money which is spent on goods we need (regardless of an individual's definition of a need — this is purely a *local* definition). Business models dramatically change. Classical economic "growth", along with short-term orientation to revenues and profits, becomes largely meaningless under the Bitcoin monetary standard, as one cannot infinitely grow their business when measured in Bitcoin, as it is hard-capped. Instead, businesses need to focus on providing only true value to people and can plan in long-term, since their monetary power does not depreciate in time.

Under the Bitcoin monetary standard, humans as a species are incentivized towards a savings economy and mindset, putting focus on their materialistic needs

[11] This of course occurs already today, yet people who can support themselves on fiat income do not yet need to do this.

instead on their materialistic wants, which were often not even their genuine wants, but were pushed to them in the fiat system. As roughly 8-10 billion people (of the present and future) transition from the fiat monetary system to Bitcoin, the effects on the environment become profound, with resources saved, helping sustainability, *even if this was not necessarily the original goal of every single individual using Bitcoin*.

As their Bitcoin savings allows people to accumulate and preserve the monetary power that they gained by their past work, without fear of an external authority who could depreciate its worth by external decisions, people are now free to decide: do they *want to work* while providing the actual value to others, or do they want to dedicate their time to art, poetry, science, new inventions, charity, or spiritual development, etc? Here, we assume that the reader accepts that we live in an era in which most jobs needed for human basic needs, can be done by the machines and that we are steadily approaching a post-scarcity society, in which all people can be ubiquitously provided basic needs (per Maslow's hierarchy of human needs). Technology already started to free people from *the need to work* and will continue to do so, but only if the distribution of the benefits of such a technology are shared among people.

The success of organizations will primarily be measured not by their revenue and profits, but by their network effect and the number of human lives that they qualitatively improve. An example is the Bitcoin network itself. It does not have a classical business model in the context of the existing monetary system, nor does it need one. It changes lives by pure adoption, giving people back their *freedom* and their *time*. We can only imagine the potential of a society of free people, who are not coerced to trade their time, the only scarce resource we humans actually have, for ever-inflating money, and who are incentivized by the economic system to save and mind their spending. In this sense, it is well probable that Bitcoin will help us to evolve as a species.

4.3.3 On Economic Inequalities and Bitcoin

While Bitcoin is Inclusive (Sec. 3), i.e., it provides Equal Opportunities, it does not guarantee economic equality. However, economic inequalities, in particular the very glaring ones, are much easier to address with Bitcoin as planetary money than in the current system. We provide several arguments towards supporting this claim:

1. Bitcoin allows transfer of value over the internet in a decentralized, permissionless and inclusive way. Note that this is not possible with the current banking system, as large populations of the world are unbanked and authorities can stop and censor transactions. It is unprecedentedly easy to use this

feature of Bitcoin to "equalize" wealth. A straightforward example, which applies to the world of today, are money remittances, i.e., funds sent by migrant workers (typically in wealthier countries) to friends and family in their homeland. This particular use case has already been a motivating use case in Bitcoin's adoption in El Salvador, which became the first country in the world to adopt Bitcoin as a legal tender on 7 September 2021.

2. Under the Bitcoin monetary standard, it is very easy for individuals and businesses to understand if they have a disproportionately high fraction of Bitcoin compared to others. They simply need to divide the total supply by their holdings and compare to the number of people on Earth. Note that this is impossible, in general and for the long-term, in the current monetary system regardless of ones' balance in the bank, as the supply is long-term uncapped (at any given point in time this is possible to estimate, but requires cumbersome computation of the entire fiat money supply and wealth in the world).

   With such an understanding of their own financial security, people could more easily decide to *give and donate their money* thereby reducing economic inequalities.

3. Bitcoin mining can be done practically by anyone, anywhere. While Bitcoin mining consumes energy, energy is very democratically and rather equally distributed across the entire world. In practically every corner of the planet, the sun shines, winds blow, rivers flow and/or geothermal energy exists. Bitcoin's reliance on energy expenditure therefore incentivizes research on utilizing energy which is available in a respective corner of the planet. Across the planet, renewable resources (as an aggregate) are the most equally distributed ones.

4.4 Conclusions on the Bitcoin Energy Expenditure

In the previous section, we outlined arguments which aim at justifying the energy expenditure of Bitcoin. We painted a reality which, hopefully, provokes the reader to reconsider whether Bitcoin's energy expenditure is genuinely "useless", "wasteful" and "excessive". We did this in an unapologetic way, refraining from even calculating what fraction of world's energy Bitcoin uses or will use. We maintain that, if Bitcoin brings us closer to a more sustainable world of free people, any fraction of the energy we produce as humans towards this end is justified. A very large energy expenditure on Bitcoin might, perhaps, even help catalyze our evolution to a Type I civilization on the Kardashev scale.[12] The widely-used

[12] `https://en.wikipedia.org/wiki/Kardashev_scale`

comparison of Bitcoin's energy expenditure to today's nation-states[13] should perhaps only be used to understand which nation-states would still have the power to break the security of Bitcoin (by mounting the 51% attack) if they would somehow engage all the power available in their country.

That said, we should continue investing in research on technologies that would reduce energy expenditure of Bitcoin, yet that would at the same time provide the same or stronger security and decentralization guarantees. So far, such a technology has been elusive. It seems that the second law of thermodynamics might be a challenge here, as Bitcoin reliance on repeated computation of an irreversible hash function, which is the key to its security, results in an irreversible heat production [37].

In a different approach, one might ask whether this expended energy might be used for something "actually useful" and not for simple repetitive, "useless" hashing; such efforts exists, but have so far not been successful. For instance, one proposal made up on-the-fly would be to use this energy towards, say, machine-learning and/or data science. Today, these are sometimes used for good purposes but more often simply to better place ads and "improve" businesses performance in the technologically and humanly inferior monetary fiat system we live with. We have to be very careful in what we define as "useful" and how we define value. What is useful at one level of abstraction and in one value system can be seen as a completely useless activity in another value system of reference, which might actually be considerably better for humanity.

Some readers may be motivated to precisely calculate the net effect of Bitcoin's energy consumption and might feel uneasy to accept the high-level arguments we presented here towards showing that *Bitcoin is good for humankind*. Such a calculation would certainly be interesting to see, yet challenging to perform as Bitcoin's energy expenditure will need to be contrasted to the sum of energy expenditure on activities that Bitcoin renders obsolete or reduces considerably. These include but are not limited to: legacy banking system, data centers powering ad-based spending, and the sum of all activities people undertake under coercion from the current monetary system. It would also need to take into account the human time wastes under the current system and its opportunity cost. Intuitively, we conjecture that the net result will be orders of magnitude in favor of Bitcoin.

[13] See e.g., `https://digiconomist.net/bitcoin-energy-consumption`.

5 Towards Decentralized Cloud Computing

5.1 Challenges

In the previous section, we argued for the Bitcoin network as the backbone for the money of the future. Once we embrace this thought and accept that the future will include a very secure and decentralized, albeit low-throughput consensus network, an interesting question arises: *can we leverage Bitcoin consensus in use cases other than money?*

One possibly interesting use case is decentralized cloud computing and, more generally, the internet. Today's internet is becoming increasingly more centralized, with half of the internet traffic in 2019 coming from only 5 internet companies (well-known social network, cloud and content providers), which is to be contrasted to thousands of autonomous systems (ASs) needed to reach this fraction in 2007 [27].

In the light of this galloping centralization of the internet, powered by the current ad-based economy, it is rightful to ask: *how can we decentralize the internet again?* We clearly need economic models and payment networks to power this, e.g., allowing paying content creators directly, instead through ads. While this can be done on Bitcoin layers 2 and higher [29], as already being implemented by Twitter, the question remains: how do we decentralize the rest of the cloud computing infrastructure, namely computation and storage, while not compromising their security?

First steps have been already made towards using blockchain to support decentralized arbitrary computation and data storage. For instance, Filecoin [6] currently allows decentralized storage of immutable data with rather large capacity (over 10 EB). A number of other projects (e.g., Ethereum [1]), make first steps towards allowing development of general-purpose applications ("smart-contracts") and running them on the blockchain. Even though these are currently used mostly for very limited use cases of debatable value, especially under the prospective Bitcoin monetary standard (such as decentralized finance), the first steps have been made.

Evolving these decentralized systems further entails two main problems: security and scaling. In an attempt to scale their system, currently based on Proof-of-Work, the Ethereum community decided to move to Proof-of-Stake as its (local) backbone, proposing a Proof-of-Stake *beacon chain* [23], citing environmental concerns around Proof-of-Work.

However, this poses issues with security and decentralization. In Section 3, we argued why a network based on Proof-of-Stake cannot be used as a Layer 1 of decentralized systems, due to lack of Inclusiveness. Beyond this fundamental issue, Proof-of-Stake protocols suffer from many attack vectors, including

"nothing-at-stake" [20], "long range" [21] and other attacks. The situation seems unfortunate, as Proof-of-Stake (PoS) and, more generally, identity-based Byzantine fault-tolerant (BFT) protocols [53] remain one of the most promising avenues for improving performance of decentralized systems, in addition to peer-to-peer payment channels [29].

On the other hand, it is unrealistic to expect Bitcoin to include smart-contracts on its main network, nor would this make sense given Bitcoin's low-throughput. Bitcoin is, rightfully so, conservatively evolving, with major upgrades being deployed rather rarely and never as a hard-fork, i.e., in a backwards-incompatible manner. This is critical to the security of the main network, and it's the correct approach. When it comes to security, less code is more, as more code practically always introduces more vulnerabilities. For the backbone of the world's monetary network, introduction of potential vulnerabilities is a clear no-go.

This situation is, fortunately, addressable. In the rest of this section:

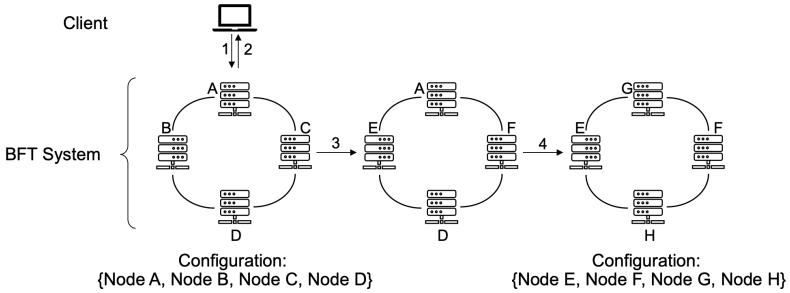
- We first discuss, in Section 5.2, how to help secure more scalable consensus protocols, leveraging the Bitcoin blockchain. More specifically, we discuss a promising approach to "saving" Proof-of-Stake blockchains by anchoring (checkpointing) their state and weighted memberhip (stake) into the Bitcoin blockchain, protecting them from long-range and nothing-at-stake attacks. This approach will be possible in a scalable manner only after the Bitcoin Taproot (supported by Bitcoin Core v0.21.1+) upgrade takes effect, at block height 709632, expected in mid-November 2021.

- Then, in Section 5.3, we discuss some of the additional challenges in extending this emerging architecture towards a genuine decentralized cloud.

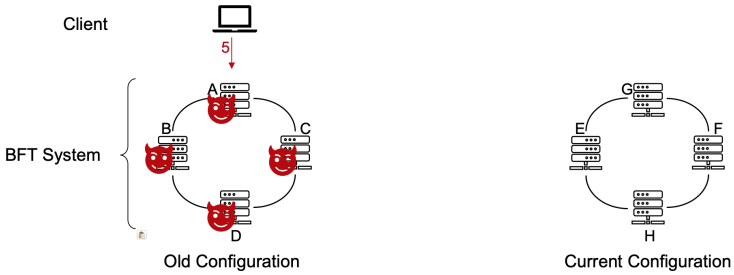
5.2 Bitcoin as a Backbone for PoS/BFT Protocols

Long-range attacks on Proof-of-Stake [21] rely on the inability of a client or other participant in the system, Alice, who disconnects from the system at time t_1 and reconnects at a later time $t_2 > t_1$, to know that validators (i.e., stakeholders or "miners" in a PoS network) who have been legitimate validators at time t_1 and who leave the system (or transfer their stake) at time t' ($t_2 > t' > t_1$), are not to be trusted anymore. These validators can fork Alice, without her being able to recognize the attack even if she is presented with a "valid" chain fork. We illustrate this attack in Figure 1 for the special case of equal weights among validators (for simplicity). This illustration also demonstrates the related "I still work here" attack [9] in permissioned BFT-based blockchains subject to membership changes (reconfiguration).

Steinhoff et al. recently proposed an approach to deal with the long-range / "I still work here" attack by anchoring BFT/PoS membership into Ethereum's



(a) Client communicates with initial configuration of members/stakeholders $\{A, B, C, D\}$, which then transition, perhaps gradually, to configuration $\{E, F, G, H\}$ (steps 1-4).



(b) When client reconnects, the initial configuration may be entirely corrupted by the adversary (step 5).

Figure 1: Illustration of the long-range ("I still work here") family of attacks on PoS (BFT) systems.

Proof-of-Work (PoW) blockchain (Eth 1.0) using Ethereum smart contracts [50]. As a Proof-of-Concept, the approach uses rather unscalable multi-signatures. This approach will no longer be viable once Ethereum abandons Proof-of-Work: the PoS of Eth 2.0 cannot be used instead PoW for anchoring as it too is vulnerable to long-range attacks.

With Bitcoin Taproot[14], one can extend this design and make it scale to thousands of PoS/BFT validators. In short, Bitcoin Taproot's enables Schnorr threshold signatures [45], which permit a public key to be constructed from multiple participant public keys and require cooperation between all participants to sign a transaction. These multi-party keys and signatures are indistinguishable from their single-party equivalents and enable arbitrary t-out-of-n signing policies, very amenable to BFT and PoS protocols.

[14] `https://en.bitcoin.it/wiki/BIP_0341`

An idea very similar to porting the approach of Steinhoff et al. [50] to Bitcoin Taproot has recently been proposed by Matt Bell [14], along with an initial design, in the context of enabling Proof-of-Stake sidechains for Bitcoin. Indeed, the approach could be used for Bitcoin PoS sidechains which stake in BTC, as well as for "external" PoS blockchains which provide staking in other tokens.

Below, we sketch how this approach might work (different implementations and nuances in the design of this approach are currently being considered — see also [14]):

1. Members of the current configuration of a PoS network A (C_0^A) jointly control, using the Schnorr threshold signature scheme enabled by Bitcoin Taproot, the funds on the Bitcoin blockchain pertaining to network A on address A_0.

2. When the configuration of PoS network A changes sufficiently[15], say to configuration C_1^A, a Bitcoin transaction is constructed, outside the Bitcoin blockchain, as follows:

   (a) Members of the old configuration C_0^A jointly sign a single Bitcoin transaction, which transfers BTC belonging to network A from address A_0 to new address A_1. Here, address A_1 is jointly controlled by members of the new configuration C_1^A.

   (b) The *OP_RETURN* opcode in the Bitcoin transaction can be used to store a hash of any additional information validators or users of network A need, such as the Merkle root of the most recent weighted stakeholders or even the Merkle root of the entire state of network A.[16]

   (c) Whatever information is referred to by its hash and stored in the *OP_RETURN* opcode of the Bitcoin transaction, could then be stored in its entirety into an external content-addressable decentralized storage. This could be for example, Filecoin decentralized storage, or the Interplanetary File System (IPFS) [15, 7].

   In this case, the Filecoin/IPFS content ID (CID)[17] of new membership C_1^A, and the latest state checkpoint of network A, could be stored in

[15] Here, the notion of a "sufficient membership change" is intentionally underspecified and may depend on the security policy of network A.

[16] Per `https://en.bitcoin.it/wiki/OP_RETURN`, "Many members of the Bitcoin community believe that the use of *OP_RETURN* is irresponsible... for arbitrary data". Fortunately, Bitcoin is inclusive and decentralized and one does not need to ask permission from other members of the community for using Bitcoin in a certain way.

[17] See `https://docs.ipfs.io/concepts/content-addressing/`.

the Bitcoin transaction OP_RETURN opcode, whereas the data corresponding to said CID could be stored in Filecoin/IPFS.

3. The "account" of network A on the Bitcoin blockchain needs to be periodically refunded with BTC to account for transaction fees on the Bitcoin network. This could be done by having new nodes joining or staking on network A deposit funds to the latest address used by network A on the Bitcoin blockchain (above, address A_1).

4. With such an approach to reconfiguration in place, Alice, the user of PoS network A from the beginning of our long-range attack example, can resolve, by running a Bitcoin full-node and a client of content-addressable decentralized storage (e.g., Filecoin/IPFS), all the information pertaining to PoS network A.

   It is worth noting that running a Bitcoin full node is not an issue here for Alice, due to Bitcoin's high operational decentralization (Sec. 4.2), as a Bitcoin node could be run by an IoT device (e.g., Raspberry Pi) and a standard hard disk. In this context, the fact that Bitcoin blockchain size grows "slowly" due to the small block size and 10-minute block time is an important advantage.

Clearly, in the design outlined above, in addition to using Bitcoin as a stake (membership) anchor, PoS network A could use its own content addressable storage implementation instead of Filecoin/IPFS. However, such implementations are rather involved and it is considerably easier for developers of new PoS networks to leverage existing building blocks for the decentralized internet, i.e., Bitcoin and Filecoin/IPFS.

5.3 Next Steps for Decentralized Computation and Storage

In order for decentralized cloud computing and the decentralized internet (so-called Web3) to become pervasive and supersede current Web2 internet and centralized cloud computing, it is reasonable to require Web3 to run Web2 workloads, with billions of transactions per second, low latencies, high security, etc. All this should be ideally done with higher privacy, censorship resilience (freedom of information) and availability, to provide benefits for humanity over Web2.

This tasks is very challenging, in particular in the context of scalability. However, we firmly believe it is achievable. Towards this goal, we highlight some of the decentralized systems areas which require more research. In a non-exhaustive approach, focusing on problems related to scaling distributed computing for Web3, we divided the problems to be tackled into the following research areas: 1) hierarchical scaling, 2) scaling consensus proper, and 3) scalable execution. These

are supported by research considerations that are pervasive to all of these work areas: security (including privacy), decentralization, and correctness of design and implementation.

Hierarchical scaling. Assuming a future Web3 handling Web2-sized workloads, one cannot rely on a single L2 blockchain network (assuming Bitcoin at L1) to "rule them all", much like today's web workloads are not executed on any single centralised machine.

This immediately brings sharding to the picture, i.e. the horizontal scaling of decentralized systems. This requires extending the work at the intersection of classical distributed computing and database problems related to concurrency — such as cross-shard transaction semantics and shard state management — to Web3-specific security challenges. There is already considerable work in this area (see, e.g., [38, 32, 56] and the surveys [57, 55]), which is picking up in pace in recent years, with more interesting and creative proposals appearing recently (e.g., [34, 39]). Sharding challenges related to scalability and security of shards are complemented by low-level interoperability [13] challenges among consensus protocols of different families.

Not precluding alternative designs, we envision, as an extension of our designed outlined in Section 5.2, a world of *hierarchical consensus protocols* in which children shards, powered by faster and more performant consensus protocols, leverage stronger security of their parent shards, which might in turn have weaker performance. In such a hierarchical approach, checkpointing into a higher security context will be important.

Scaling consensus proper. In a possible hierarchical consensus approach we mentioned above, different consensus protocols will be applicable to different shards, layers and use cases.

We propose identifying the "best" consensus protocol within a given security and scalability context, applicable to a single shard. Different consensus protocols may be relevant, e.g., targeting distinct sybil attack protection mechanisms (e.g., based on verifiable resource commitments [42, 41] or Proof-of-Stake), participating population size (e.g., whether we are dealing with thousands [28] or hundreds [49] of nodes), performance, and security guarantees. Here, we should pay particular attention to the tradeoffs between ease of design, maintainability, and system guarantees.

Here, it is very important to understand that scalable consensus protocol implementations need to evolve into full-fledged battle-tested production systems. There is work ahead of us not only in protocol design but in robust testing and verification tools.

Scaling execution. Advances in scaling through sharding and scaling consensus are insufficient without considering the scalability of application execution. Existing blockchain systems largely follow, within a single shard, a classical order-execute architecture in which sequential execution of applications (e.g. smart contracts or payment scripts) follows prior ordering of transactions [54]. This introduces severe performance bottlenecks, which have been the target of decades of research in databases, multi-core processors, and distributed systems.

Here, we will need to revisit the parallel execution problem through the prism of existing decentralised applications (based on dedicated smart-contract virtual machines [22]) but also aim at accommodating larger-scale general-purpose scale computations, including federated machine learning workloads or computations over large data [33, 16], e.g., stored on IPFS. This may involve porting alternative execution models, such as execute-order-validate, tested in the domain of permissioned blockchains [11] to the world of permissionless systems. Alternative application programming models which support better efficiency of parallel execution (e.g., CRDTs [46]) are also of particular interest.

Finally, very relevant to all these research areas in particular hierarchical scaling and consensus proper, will be further research on improving distributed randomness used in actual decentralized systems (e.g., drand [5]), using perhaps Bitcoin itself [18], or approaches based, e.g., on verifiable delay functions [17].

6 Conclusions and Future Work

In this paper we argued for Bitcoin as a backbone of future decentralized money and the internet, making a case for it to be good for humankind.

The main novelty of this paper is not necessarily technical — the main contribution is a proposal to give existing systems, notably Bitcoin, a different look. Author's own view on Bitcoin changed only relatively recently from "generally positive" to the one depicted in this paper, and this was equal to an enlightenment. Arguably, one cannot change other people, but can only change oneself. The goal of this paper is simply to share the message.

Future work in building decentralized systems, briefly tackled in Section 5.3, will, by definition, not be a task of any single individual or organization. An interesting question here is how do we, decentralized systems researchers and developers, organize better towards building coherent and interconnected systems, while remaining in a decentralized working mode and outside the control of any single organization?

In an attempt to facilitate this, we recently established ConsensusLab as a research and development hub and part of Protocol Labs. One of the main goals of ConsensusLab is to serve as a weak synchronization point for researchers across

academia, open-source blockchain projects and industry, and to help foster discussion and open collaboration pertaining to decentralized consensus and related technologies that are at the heart of decentralized systems. The collaboration here is, in part, related to openly discussing, criticizing and, perhaps, gently steering technical innovations in consensus, but is also very much related to meta-collaboration, discussing how do we build tools, incentives and funding schemes towards working together in a decentralized way. We wholeheartedly look forward to these collaborations and future innovations.

Acknowledgments

I am indebted to Seth Gilbert and Stefan Schmid, editors of the BEATCS Distributed Computing column, for providing me with an invaluable opportunity, inviting my contribution, catalyzing the writeup of this position paper and giving valuable feedback. I thank Evan Miyazono for very useful comments, including questioning my arguments. I am deeply grateful to Jorge Soares, whose very detailed feedback helped considerably improve the paper. I thank Juan Benet for inspiring discussions around hierarchical consensus and for proposing ambitious goals for Web3.

Finally, I wish to express my deep gratitude to the people who helped me come to the viewpoint expressed in this paper. I am indebted to Andreja Čobeljić for helping me better understand the world around me and many discussions on cryptocurrencies and beyond. I thank Bojana and Nenad Mijailović for referring me to the works of the economists of the Austrian School. Finally, I heartily thank Ana Vukolić for endless discussions and invaluable feedback on the argument presented here.

References

[1] Ethereum. `http://ethereum.org`.

[2] Proof of stake instead of proof of work. `https://bitcointalk.org/index.php?topic=27787.0`, 2011.

[3] I AM HODLING. `https://bitcointalk.org/index.php?topic=375643.0`, 2013.

[4] Cambridge Bitcoin electricity consumption index. `https://cbeci.org/cbeci/comparisons`, 2021.

[5] drand: Distributed randomness beacon. `https://drand.love/`, 2021.

[6] Filecoin. `https://filecoin.io/`, 2021.

[7] Interplanetary file system (ipfs). `https://ipfs.io/`, 2021.

[8] Wikipedia: Richard Cantillon. `https://en.wikipedia.org/wiki/Richard_Cantillon`, 2021.

[9] M. K. Aguilera, I. Keidar, D. Malkhi, J. Martin, and A. Shraer. Reconfiguring replicated atomic storage: A tutorial. *Bull. EATCS*, 102:84–108, 2010.

[10] B. Alpern and F. B. Schneider. Recognizing safety and liveness. *Distributed Comput.*, 2(3):117–126, 1987.

[11] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolic, S. W. Cocco, and J. Yellick. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference, EuroSys 2018, Porto, Portugal, April 23-26, 2018*, pages 30:1–30:15, 2018.

[12] S. Azouvi, M. Maller, and S. Meiklejohn. Egalitarian society or benevolent dictatorship: The state of cryptocurrency governance. In A. Zohar, I. Eyal, V. Teague, J. Clark, A. Bracciali, F. Pintore, and M. Sala, editors, *Financial Cryptography and Data Security - FC 2018 International Workshops, BITCOIN, VOTING, and WTSC, Nieuwpoort, Curaçao, March 2, 2018, Revised Selected Papers*, volume 10958 of *Lecture Notes in Computer Science*, pages 127–143. Springer, 2018.

[13] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia. A survey on blockchain interoperability: Past, present, and future trends, 2021.

[14] M. Bell. Proof-of-stake bitcoin sidechains. `https://gist.github.com/mappum/da11e37f4e90891642a52621594d03f6`, 2021.

[15] J. Benet. IPFS - content addressed, versioned, P2P file system. *CoRR*, abs/1407.3561, 2014.

[16] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečný, S. Mazzocchi, H. B. McMahan, T. V. Overveldt, D. Petrou, D. Ramage, and J. Roselander. Towards federated learning at scale: System design, 2019.

[17] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In *Advances in Cryptology – CRYPTO 2018*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018.

[18] J. Bonneau, J. Clark, and S. Goldfeder. On Bitcoin as a public randomness source. *IACR Cryptology ePrint Archive*, 2015:1015, 2015.

[19] N. Bostrom. Are we living in a computer simulation? *Philosophical Quarterly*, 53(211), 2003.

[20] J. Brown-Cohen, A. Narayanan, A. Psomas, and S. M. Weinberg. Formal barriers to longest-chain proof-of-stake protocols. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, EC '19, page 459–473, New York, NY, USA, 2019. Association for Computing Machinery.

[21] E. Deirmentzoglou, G. Papakyriakopoulos, and C. Patsakis. A survey on long-range attacks for proof of stake protocols. *IEEE Access*, 7:28712–28725, 2019.

[22] T. D. Dickerson, P. Gazzillo, M. Herlihy, and E. Koskinen. Adding concurrency to smart contracts. *Distributed Comput.*, 33(3-4):209–225, 2020.

[23] Ethereum Foundation. Proof-of-stake (PoS). `https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/`, July 2021.

[24] G. C. Fanti, L. Kogan, S. Oh, K. Ruan, P. Viswanath, and G. Wang. Compounding of wealth in proof-of-stake cryptocurrencies. In I. Goldberg and T. Moore, editors, *Financial Cryptography and Data Security - 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18-22, 2019, Revised Selected Papers*, volume 11598 of *Lecture Notes in Computer Science*, pages 42–61. Springer, 2019.

[25] P. D. Filippi and B. Loveluck. The invisible politics of Bitcoin: Governance crisis of a decentralized infrastructure. *Internet Policy Review*, 5(4), 2016.

[26] A. E. Gencer, S. Basu, I. Eyal, R. van Renesse, and E. G. Sirer. Decentralization in Bitcoin and Ethereum networks. In S. Meiklejohn and K. Sako, editors, *Financial Cryptography and Data Security - 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26 - March 2, 2018, Revised Selected Papers*, volume 10957 of *Lecture Notes in Computer Science*, pages 439–457. Springer, 2018.

[27] P. Gigis, M. Calder, L. Manassakis, G. Nomikos, V. Kotronis, X. Dimitropoulos, E. Katz-Bassett, and G. Smaragdakis. Seven years in the life of hypergiants' off-nets. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference*, SIGCOMM '21, page 516–533, New York, NY, USA, 2021. Association for Computing Machinery.

[28] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.

[29] L. Gudgeon, P. Moreno-Sanchez, S. Roos, P. McCorry, and A. Gervais. Sok: Layer-two blockchain protocols. In J. Bonneau and N. Heninger, editors, *Financial Cryptography and Data Security - 24th International Conference, FC 2020, Kota Kinabalu, Malaysia, February 10-14, 2020 Revised Selected Papers*, volume 12059 of *Lecture Notes in Computer Science*, pages 201–226. Springer, 2020.

[30] F. A. Hayek. *The Constitution of Liberty: The Definitive Edition*. The University of Chicago Press, 1960.

[31] D. Karakostas, A. Kiayias, C. Nasikas, and D. Zidros. Cryptocurrency egalitarianism:a quantitative approach. In *International Conference on Blockchain Economics, Security and Protocols (Tokenomics 2019)*, 2019.

[32] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 583–598. IEEE Computer Society, 2018.

[33] J. Konečny, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon. Federated learning: Strategies for improving communication efficiency, 2017.

[34] M. Król, O. Ascigil, S. Rene, A. Sonnino, M. Al-Bassam, and E. Rivière. Shard scheduler: object placement and migration in sharded account-based blockchains, 2021.

[35] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Trans. Software Eng.*, 3(2):125–143, 1977.

[36] L. Lamport, R. E. Shostak, and M. C. Pease. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[37] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[38] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, page 17–30, New York, NY, USA, 2016. Association for Computing Machinery.

[39] G. A. Marson, S. Andreina, L. Alluminio, K. Munichev, and G. Karame. Mitosis: Practically scaling permissioned blockchains, 2021.

[40] A. Miller. *Permissioned and Permissionless Blockchains*, pages 193–204. 2019.

[41] T. Moran and I. Orlov. Simple proofs of space-time and rational proofs of storage. Cryptology ePrint Archive, Report 2016/035, 2016. `https://ia.cr/2016/035`.

[42] Protocol Labs. Filecoin: a decentralized storage network. `https://filecoin.io/filecoin.pdf`, 2017.

[43] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.

[44] A. R. Sai, J. Buckley, B. Fitzgerald, and A. LeGear. Taxonomy of centralization in public blockchain systems: A systematic literature review. *Information Processing and Management*, 58(4), July 2021.

[45] C. P. Schnorr. Efficient identification and signatures for smart cards. In G. Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.

[46] M. Shapiro, N. M. Preguiça, C. Baquero, and M. Zawirski. Conflict-free replicated data types. In X. Défago, F. Petit, and V. Villain, editors, *Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings*, volume 6976 of *Lecture Notes in Computer Science*, pages 386–400. Springer, 2011.

[47] H. C. Simons. Rules versus authorities in monetary policy. *Journal of Political Economy*, 44:1–30, 1936.

[48] B. Srinivasan. Quantifying decentralization. Blockstack Summit 2017, `https://www.youtube.com/watch?v=4UXT5YVJwB4`, 2017.

[49] C. Stathakopoulou, T. David, M. Pavlovic, and M. Vukolić. Mir-BFT: High-throughput robust BFT for decentralized networks, 2021.

[50] S. Steinhoff, C. Stathakopoulou, M. Pavlovic, and M. Vukolić. BMS: Secure Decentralized Reconfiguration for Blockchain and BFT Systems, 2021.

[51] A. S. Tanenbaum and M. van Steen. *Distributed systems - principles and paradigms, 2nd Edition*. Pearson Education, 2007.

[52] C. Troncoso, M. Isaakidis, G. Danezis, and H. Halpin. Systematizing decentralization and privacy: Lessons from 15 years of research and deployments. *Proc. Priv. Enhancing Technol.*, 2017(4):404–426, 2017.

[53] M. Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *International Workshop on Open Problems in Network Security (iNetSec)*, pages 112–125, 2015.

[54] M. Vukolić. Rethinking permissioned blockchains. In *Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts*, BCC '17, page 3–7, New York, NY, USA, 2017. Association for Computing Machinery.

[55] G. Wang, Z. J. Shi, M. Nixon, and S. Han. Sok: Sharding on blockchain. In *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, AFT '19, page 41–61, New York, NY, USA, 2019. Association for Computing Machinery.

[56] J. Wang and H. Wang. Monoxide: Scale out blockchains with asynchronous consensus zones. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 95–112, Boston, MA, Feb. 2019. USENIX Association.

[57] G. Yu, X. Wang, K. Yu, W. Ni, J. A. Zhang, and R. P. Liu. Survey: Sharding in blockchains. *IEEE Access*, 8:14155–14181, 2020.

The Education Column

BY

Juraj Hromkovič and Dennis Komm

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

BEATCS no 135

The Problem with Debugging in Current Block-based Programming Environments

Juraj Hromkovič and Jacqueline Staub
Department of Computer Science, ETH Zürich
Universitätstrasse 6, 8092 Zürich, Switzerland
{juraj.hromkovic, jacqueline.staub}@inf.ethz.ch

Abstract

Programming is a highly creative activity that cultivates problem-solving skills, but also requires a high degree of precision. The Logo philosophy empowers novice programmers to become successful problem solvers who are capable of dealing with mistakes. The widespread emergence of block-based programming languages has lead to an active prevention of certain classes of errors while others still prevail. Rather than providing support, most block-based interfaces, to some extent, abandon learners in the difficult task of troubleshooting. We present a block-based programming environment that supports autonomous troubleshooting. Programming competences are not restricted to writing programs only. Correcting, modifying, and extending the functionality of previously-written programs is equally important and should not be neglected. Learning by productive failure is an unavoidable part of education.

1 Why Debugging is Important

Programming is a creative activity that develops constructive problem-solving skills and which finally has found its way into public school education. By programming, students communicate with a machine and instruct it what to do. For this purpose, they learn a programming language which essentially is a language that the machine "understands" [20]. Unlike humans, however, machines are not able to interpret ambiguities or handle imprecise statements. As a consequence, even the smallest flaws can cause a program to result in unexpected behavior or fail entirely. Making mistakes is natural and troubleshooting is one of the most crucial parts of a programmer's skill set [13].

Under the term *debugging* we understand a systematic process that is aimed at finding the reason why a given program does not work as intended and fixing

the underlying cause [8]. Although the primary goal of debugging is simply to fix bugs, we consider the path of reaching this goal equally important: while analyzing novices' debugging patterns, Perkins et al. [18] found that tinkering is a widespread practice among novice debuggers. Generally, novices' debugging process often seems to lack systematicity [14] and especially locating errors is challenging for inexperienced programmers [13]. Fostering debugging skills explicitly, on the other hand, has shown to have a positive impact on learners' conceptual understanding of programming [2, 9].

The birth of the Logo programming language in 1967 marked the dawn of a profound philosophy of debugging for educational purposes which we will subsume under the term *Logo philosophy* [1, 6]. Papert expressed a meta-cognitive vision for programming education that emphasizes the need for learners to "rethink their thinking and to learn about their own learning" [16]. Debugging is an essential component of this philosophy as the following *Mindstorms* excerpt shows [17]:

> What we see as a good program with a small bug, the child sees as "wrong", "bad", "a mistake". School teaches that errors are bad; the last thing one wants to do is to pore over them, dwell on them, or think about them. [...] The debugging philosophy suggests an opposite attitude. Errors benefit us because they lead us to study what happened, to understand what went wrong, and, through understanding, to fix [the program]. Experience with computer programming leads children more effectively than any other activity to "believe in" debugging.

Confronting students with problems and allowing them to fail is relevant far beyond the scope of programming and computer science only. Van Lehn et al. [12] provide strong evidence for delaying external support during instruction and letting students learn by failure. This form of impasse-driven learning is associated with concept understanding [22]. While students are trying (and failing) to solve unstructured problems, they build relevant mental models that can later be used for more efficient direct instruction. This concept of *productive failure* has shown clear learning benefits for students [7].

Programming provides a good framework for students to learn by means of the productive failure approach. Programs are sequences of commands that are executed one after the other and thus the effect of a program can be anticipated and planned. Programs are typically written with a clear expectation of what exactly will happen during execution. Due to mistakes, however, reality and expectation may diverge unexpectedly. Quite often, this outcome puzzles programmers and leaves them with the challenging task of finding out where an error crept in. Narrowing down the exact location of an error has proven to be especially hard for novice programmers [13].

Scholars have tried different measures (e.g., pedagogical interventions via didactic methods, carefully chosen analogies, and technological tools) to support novices' troubleshooting and error localization throughout the past 50 years. This article highlights some technical debugging methods and contrasts them with today's common practice in block-based programming. We present a block-based programming environment that is specifically designed for novices to acquire debugging skills by handling logical errors. We intend to raise awareness and hope to spark more active research in the domain of debugging and block-based programming.

2 Two Different Perspectives on Debugging

Logo not only revolutionized the domain of programming education but it also brought forth a variety of debugging mechanisms, strategies, and tools that were devised with the intent of fostering novice programmers' debugging skills. With the dawn of block-based programming, debugging practices have shifted away from the deliberate and active exposure to errors towards a more preventive form of error handling. In this section, we present the key attributes of both of these perspectives and hint at their respective implications on novice programmers.

2.1 Debugging in Accordance with the Logo Philosophy

The term "Logo" characterizes not only the name of a family of programming languages but also a philosophical interpretation of how learning is best achieved [1]. This philosophy is deeply anchored in the ideas of *constructivist* learning by Jean Piaget (i.e., "learning by doing") and the *constructionist* visions of Seymour Papert (i.e., "learning by making"). Within the context of these theories, programming simply represents a *tool for learning*. The Logo philosophy is based on the rationale of making students iteratively construct, analyze, and refine their own learning artifacts. While programming, students develop creative solutions to given problems. They learn to formally describe the resulting algorithm as a program whose execution they can delegate to a computer. Errors can creep in at any point during this process and students must learn to deal with unexpected results.

2.1.1 The Logo-Way of Handling Errors

Programming errors can occur in virtually all conceivable situations and the underlying causes are manifold. With the Logo philosophy (or closely connected with it), numerous debugging mechanisms have been invented with the aim of supporting novice programmers in error handling and developing their debugging

skills. Three of these ideas (creating observable models of computation, providing advanced error diagnostics, and integrating process-related debugging mechanisms directly into the language) will be presented separately.

- **Introducing the Turtle: An Observable Model of Computation**

  Programmers need to express their ideas as sequences of commands that are subsequently interpreted and executed. Exactly how this interpretation and execution takes place, however, verges on pure magic for most novices. Numerous models of computation exist and some of them have been created specifically with the intent of helping novices peek inside this magic blackbox of program execution [3]. One of the first such models was the Logo turtle with which programmers can draw geometric shapes onto a screen. Thanks to the turtle's observable behavior, errors become obvious and programmers learn to systematically test their programs.

- **Developing Advanced Error Diagnostics for Novices**

  As many other programming languages, Logo knows at least three different types of errors: (i) Structural syntax errors arise due to grammatical inconsistencies that can be detected by the parser. (ii) Structural semantic errors are typically detected at runtime and lead to early termination of program execution. (iii) Logical semantic errors are neither detected by the parser nor at runtime and their characterizing feature is an unexpected program behavior. Several techniques allow novice programmers to cope with such errors more easily, from custom error messages, to in-line error highlighting, all the way to tailor-made debuggers.

- **Extending the Language with Built-In Debugging Mechanisms**

  Logo originally came with its own debugging vocabulary that was directly incorporated into the language. Via the `pause` command, for example, the execution of Logo programs could be interrupted at any point in time. This allows the existing program variables and intermediate results to be examined using an interactive mode of programming. The `trace` command is useful for tracing the program flow across multiple layers of abstraction. Several of these functionalities add to the underlying concept of *Read-Eval-Print-Loop* (REPL) that Logo inherited from Lisp.

In summary, the Logo philosophy is characterized by its wide range of different debugging mechanisms that supports troubleshooting with novice programmers. The focus of the Logo approach is on helping learners understand and make use of the complex and abstract concepts of automatic information processing.

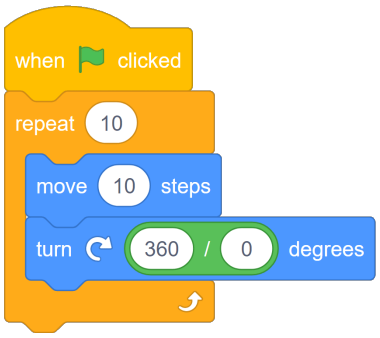


Figure 1: Example of a (silently failing) runtime error in Scratch. The result of a division by zero is mathematically undefined.

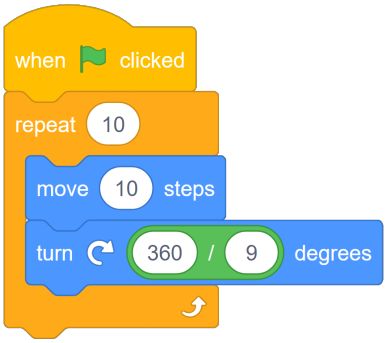


Figure 2: Example of a logical error in Scratch. In a decagon (polygon with 10 corners of equal angle), each angle must be 360/10 degrees.

2.2 Debugging and Block-Based Programming

After more than three decades, Logo had to slowly hand over its undisputed monopoly to the newer type of block-based programming languages; a class of languages that allows programmers to write programs by snapping command blocks in a graphical user interface. Using this feature, it is possible to prevent syntactic errors almost entirely from happening [10].

Although many block-based interfaces still contain traits of the Logo turtle, the approach of how block-based environments handle errors is different from the original Logo philosophy. Although certain classes of errors can be eliminated using a block-based interface, the issue of debugging is far from resolved: besides syntactic errors there are two more classes of errors which must be considered, independently of whether the chosen environment uses a block- or text-based interface, namely (i) logical errors and (ii) runtime errors. Both of these error classes are semantic in nature and cannot be handled considering only the syntactical attributes of a programming language. Figures 1 and 2 show two examples of such errors in Scratch. The example on the left results in a runtime error (i.e., division by zero is mathematically undefined) while the example on the right turns out to be logically incorrect.

One troubleshooting strategy used by many block-based environments consists in using "failsoft commands" which essentially result in runtime errors to be treated like logical errors. Rather than stopping execution, as is usually the case with runtime errors, many block-based programming environments swallow runtime errors and resume execution without informing the user. The concept of failsoft

commands became popular with the Scratch programming environment more than ten years ago [10]. By now, it is common practice among many well-known and broadly-used block-based programming platforms such as Snap, Code.org., and even Blockly. We question the usefulness of failsoft commands. Error localization is known to be one of the hardest parts of debugging. Intentionally hiding relevant information from users thus seems contra-productive from a debugging point of view.

3 Our Approach

Logo was originally designed to provide a broad variety of supporting features to facilitate error handling. This can be observed in a plethora of different mechanisms from syntax (e.g., Logo is particularly light on syntax with a bare minimum of syntactic elements [21]) to the role of the turtle [17] and the question of how to best phrase compiler error messages for beginners to understand [5, 19].

The argument for introducing block-based programming was inherently motivated by error handling, too. Syntax errors were claimed to distract novices from algorithmic aspects and for certain age ranges handling a keyboard was considered an additional challenge by itself. For those factors, block-based programming has proven to be a useful tool. Still, block-based programming cannot help against all errors and dedicated mechanisms for handling runtime errors and logical errors are still required.

XLogoOnline is a tailor-made programming environment for novice programmers from kindergarten to grade 6. Students at the lower end of this age range typically are not able to read and write yet and their formal education in mathematics has only just begun. In order to still allow students to dive into the world of algorithms and programming, XLogoOnline provides a block-based programming interface for students between kindergarten and grade 4. This interface was extended with three forms of support for troubleshooting: (i) a small command set was chosen not to admit any runtime errors, (ii) the environment includes the possibility for physical program execution to reduce struggles with change of perspective, and (ii) an exercise collection tool was developed which verifies student solutions and reports some logical errors. These three forms of troubleshooting will be elaborated individually.

3.1 Runtime Errors not Possible

Young children between 6 and 8 years old can learn to program despite their challenges in reading and writing and limited formal mathematics background. XLogoOnline provides a block-based interface which allows students to navigate a

turtle on a rectangular grid. For this purpose, students are provided with six block commands:

1. **forward** moves the turtle ahead. Each of these movements is unparameterized and uses an internal unit distance of 100 pixels, i.e., one grid cell.
2. **back** moves the turtle backwards. Each of these movements is unparameterized and uses an internal unit distance of 100 pixels, i.e., one grid cell.
3. **left** rotates the turtle 90 degrees to the left. The rotation angle is fixed and aligns with the underlying rectangular grid.
4. **right** rotates the turtle 90 degrees to the right. The rotation angle is fixed and aligns with the underlying rectangular grid.
5. **setpencolor** exchanges the pen with one of six possible colors. The selection involves a drop-down menu.
6. **repeat** allows any sequence of commands to be executed repeatedly. Students choose the number of repetitions by providing a positive integer parameter. Arithmetic operations are disallowed.

None of these six commands has the possibility to throw an unexpected runtime exception. This means that only logical errors can still occur and must be handled by programmers.

3.2 Physical Program Execution

One of the known logical difficulties Logo novices initially struggle with is connected to the concept of perspective. All of the four movement and rotation commands mentioned in Section 3.1 are interpreted from the perspective of the turtle, rather than the perspective of the programmer. This means that programmers have to imagine themselves in the position of the turtle in order to figure out whether to turn to the left or to the right.

This change of perspective is sometimes hardly even noticed by students, especially if the different perspectives align (as in Figure 3). There are, however, also cases which are notoriously hard for young programmers due to a clash of perspective. Figure 4 shows such a case where the ladybug's perspective differs from the reader's perspective. A clash of perspective may result in confusion between left and right.

In order to tackle this problem, we provide not only a virtual turtle but a physical turtle in addition as shown in Figure 8. This approach was originally proposed by Papert in the early days of Logo. Back then, the turtle was usually considered

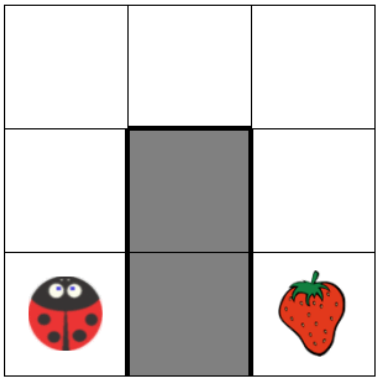


Figure 3: The change in perspective may not be noticed as much because the ladybug's perspective is the same as the programmer's perspective at the beginning.

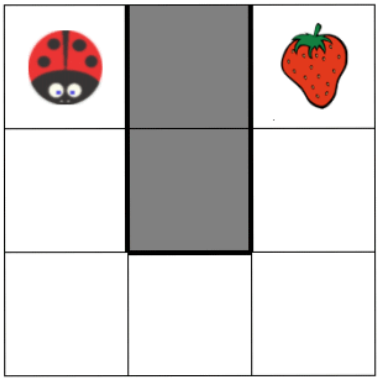


Figure 4: The ladybug's perspective deviates from that of the programmer. As a result, the required change of perspective entails additional cognitive effort.

a physical robot instead of a virtual agent on the screen. Physical robots have the advantage that a change of perspective comes naturally by simply changing one's position relative to the robot. This is something many young novices do unconsciously and which seems to improve their performance.

3.3 Exercise Collection with Automatic Verification

Programming is a precise form of communication; in order to instruct a computer to do something for us, we need to express ourselves in a way that is unambiguous and instructive enough for a mindless machine to "understand". Novice programmers must learn to deal with this required level of precision. One way of reaching this goal involves providing novices with age-appropriate exercises that challenge their conceptual understanding. We have developed an exercise collection with predefined tasks (some examples are shown in Figures 5 to 7). Student solutions to these tasks are automatically verified for correctness and the result is reported back the user.

Tasks differ in terms of what students are allowed to do in their solution. Some tasks contain fields that students are not allowed to cross. Forbidden fields may be marked visually sometimes (e.g., Figure 5) while other times, their status can only be inferred from the exercise text (e.g., Figure 6). In addition to forbidden fields, walls can be used to separate neighboring fields which would otherwise be

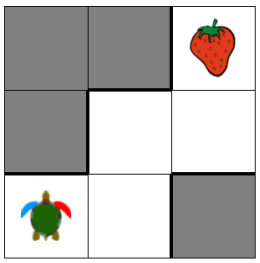


Figure 5: Find the strawberry without passing over a forbidden field.

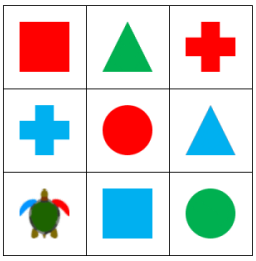


Figure 6: Collect all blue shapes without crossing over a red field.

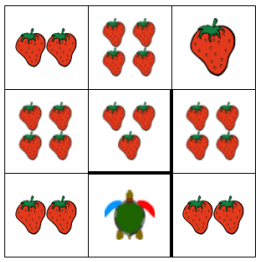


Figure 7: Collect 18 strawberries but without crossing a wall.

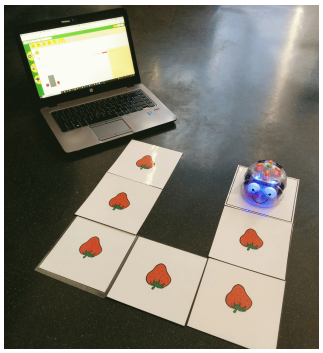


Figure 8: Students who struggle with changing perspectives benefit from executing their programs on a physical robot.

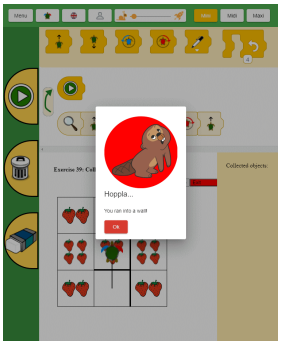


Figure 9: Error dialog that pops up whenever one of the underlying rules is violated.

connected. One such example is shown in Figure 7. Not respecting one of these conditions leads to an error dialog, as shown in Figure 9.

4 Summary and Conclusion

Not only programming itself but also dealing with errors needs to be learned. Block-based programming environments offer one-sided support that focuses mainly on the elimination of syntactic errors. There are several error classes (e.g., runtime errors and logical errors) that are not affected by the use of blocks and that are oftentimes not handled in block-based programming environments. Teaching pupils how to approach problems and making them learn from failure is of utmost

importance for pupils' cognitive development. Whether block- or text-based, it is crucial that learners leave school not as passive consumers of technology but that they learn to create their own solutions and deal with errors – one of the most natural ways of doing so consists in teaching them how to program.

References

[1] Harold Abelson. Logo for the Apple II *Peterborough, NH: BYTE/McGraw-Hill*, 1980

[2] C.-F. Chiu and H.-Y. Huang. Guided Debugging Practices of Game Based Programming for Novice Programmers. *International Journal of Information and Education Technology*, 5:343–347, 01 2015.

[3] B. du Boulay, T. O'Shea, and J. Monk. The Black Box Inside the Glass Box: Presenting Computing Concepts to Novices. *International Journal of Man-Machine Studies*, 14(3):237–249, 1981.

[4] S. Fitzgerald, G. Lewandowski, R. McCauley, L. Murphy, B. Simon, L. Thomas, and C. Zander. Debugging: Finding, Fixing and Flailing, a Multi-Institutional Study of Novice Debuggers. *Computer Science Education*, 18(2):93–116, 2008.

[5] M. Forster, U. Hauser, G. Serafini, and J. Staub. Autonomous Recovery from Programming Errors Made by Primary School Children. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 17–29. Springer, 2018.

[6] Juraj Hromkovic, Dennis Komm, Regula Lacher, and Jacqueline Staub. Teaching with LOGO philosophy. In *Encyclopedia of Education and Information Technologies*, Springer, 2019.

[7] M. Kapur and K. Bielaczyc. Designing for Productive Failure. In *Journal of the Learning Sciences*, 21:45–83, 12 2012.

[8] C. Kim, J. Yuan, L. Vasconcelos, M. Shin, and R. Hill. Debugging During Block-Based Programming. *Instructional Science*, 46, 10 2018.

[9] M. J. Lee and A. J. Ko. Comparing the Effectiveness of Online Learning Approaches on CS1 Learning Outcomes. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research*, ICER '15, page 237–246, New York, NY, USA, 2015. Association for Computing Machinery.

[10] J. Maloney, L. Burd, Y. Kafai, N. Rusk, B. Silverman, and M. Resnick. Scratch: a Sneak Preview [Education]. In *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.*, pages 104–109. IEEE, 2004.

[11] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond. The Scratch Programming Language and Environment. *ACM Transactions on Computing Education (TOCE)*, 10(4):1–15, 2010.

[12] K. VanLehn, Toward a theory of impasse-driven learning. *In Learning issues for intelligent tutoring systems* pages 19-41. Springer, New York, NY, 1988.

[13] R. McCauley, S. Fitzgerald, G. Lewandowski, L. Murphy, B. Simon, L. Thomas, and C. Zander. Debugging: a Review of the Literature from an Educational Perspective. *Computer Science Education*, 18(2):67–92, 2008.

[14] T. Michaeli and R. Romeike. Improving Debugging Skills in the Classroom: The Effects of Teaching a Systematic Debugging Process. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education*, WiPSCE'19, New York, NY, USA, 2019. Association for Computing Machinery.

[15] N. Nusen and A. Sipitakiat. Robo-Blocks: a Tangible Programming System with Debugging for Children. In *Proceedings of the 19th international conference on computers in education. Chiang Mai*, pages 1–5, 2011.

[16] S. Papert. *Teaching Children Thinking (LOGO Memo)*. Massachusetts Institute of Technology, USA, 1971.

[17] S. A. Papert. *Mindstorms: Children, Computers, and Powerful Ideas*. Basic books, 2020.

[18] D. Perkins, C. Hancock, R. Hobbs, F. Martin, and R. Simmons. Conditions of Learning in Novice Programmers. *Journal of Educational Computing Research*, 2:37 – 55, 1986.

[19] C. Solomon, B. Harvey, K. Kahn, H. Lieberman, M. L. Miller, M. Minsky, A. Papert, and B. Silverman. History of Logo. *Proc. ACM Program. Lang.*, 4(HOPL), June 2020.

[20] J. Staub, M. Barnett, and N. Trachsler. Programmierunterricht vom Kindergarten bis zur Matura in einem Spiralcurriculum. *Informatik Spektrum*, 42(2):102 – 111, 2019-04.

[21] Jacqueline Staub Programming in K–6: Understanding Errors and Supporting Autonomous Learning. *ETH Zurich*, Thesis, 2021.

[22] C.-Z. Yen, P.-H. Wu, and C.-F. Lin. Analysis of Experts' and Novices' Thinking Process in Program Debugging. In K. C. Li, F. L. Wang, K. S. Yuen, S. K. S. Cheung, and R. Kwan, editors, *Engaging Learners Through Emerging Technologies*, pages 122–134, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

THE LOGIC IN COMPUTER SCIENCE COLUMN

BY

YURI GUREVICH

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

BEATCS no 135

A TUTORIAL FOR COMPUTER SCIENTISTS ON FINITE EXTENSIVE GAMES WITH PERFECT INFORMATION

Krzysztof R. Apt
CWI, Amsterdam, The Netherlands
MIMUW, University of Warsaw, Poland
apt@cwi.nl

Sunil Simon
Department of CSE, IIT Kanpur, India
simon@cse.iitk.ac.in

Abstract

We provide a self-contained introduction to finite extensive games with perfect information. In these games players proceed in turns having, at each stage, finitely many moves to their disposal, each play always ends, and in each play the players have complete knowledge of the previously made moves. Almost all discussed results are well-known, but often they are not presented in an optimal form. Also, they usually appear in the literature aimed at economists or mathematicians, so the algorithmic or logical aspects are underrepresented.

Contents

1 Introduction

In computer science for a long time the most commonly studied games have been infinite two-player games (see, e.g., [1] for an account of some of the most popular classes). With the advent of algorithmic game theory various classes of games studied by economists became subject of interest of computer scientists, as well. These games usually involve an arbitrary finite number of players. Among them one the most common ones are strategic games, in which the players select their strategies simultaneously. They have been covered in several books and surveys.

However, in our view a systematic account of another most popular class of games, extensive games with perfect information, is missing. It is true that they are extensively discussed in several books, mostly written for theoretical economists. However, in the introductory texts technical results are usually omitted and illustrated by examples (e.g., [5]). In turn, in the advanced textbooks the presentation is often difficult to follow since these games are introduced as a special case of the extensive games with imperfect information, which leads to involved notation (e.g., [9]). An exception is [16] which devotes a separate chapter to extensive games with perfect information.

From the point of view of computer science the main results are usually not presented in an optimal way. For example, the backward induction is often introduced in a verbose way, or formulated in a way that hides its algorithmic aspect. This way the optimal result that relates it to the set of *all* subgame perfect equilibria (Theorem 7) is often missed. We explain here that it is actually a nondeterministic algorithm that can be even presented as a parallel algorithm. We also discuss an important article [3] that formalizes common knowledge of rationality in finite extensive games with perfect information and relates it to backward induction.

Further, Theorem 17 that clarifies the relation between backward induction and the iterated elimination of weakly dominated strategies, is either only illustrated by an example (e.g., [5]) or only a sketch of a proof is provided (e.g., [16]).

Also, some more recent results, Theorem 19, due to [7], and Theorem 20, due to [12], merit in our opinion some attention among computer scientists. Finally, the so-called Zermelo theorem about chess-like games is in our view often proved in a too elaborate way, starting with the exposition in the classic [22].

These considerations motivated us to write a tutorial presentation of finite extensive games with perfect information aimed at computer scientists. Often these games are discussed by introducing strategic games first. We follow this approach, as well, as it allows us to view extensive games as a subclass of strategic games for which some additional notions can be defined and for which additional results hold.

In our presentation we shall often refer to the account given in [16] that comes closest to our ideal. We shall strengthen some of their results, provide more detailed proofs of some of them, and add some new results. Also, we shall recall their natural notion of a *reduced strategy* that in our view merits more attention.

2 Preliminaries on strategic games

To discuss extensive games it is convenient to introduce first strategic games. A ***strategic game*** for $n \geq 1$ players consists of:

- a set of players $\{1, \ldots, n\}$,

and for each player i

- a non-empty (possibly infinite) set S_i of ***strategies***,
- a ***payoff function*** $p_i : S_1 \times \cdots \times S_n \to \mathbb{R}$.

We denote it by $(S_1, \ldots, S_n, p_1, \ldots, p_n)$. So the set of players is implicit in this notation.

Strategic games are studied under the following basic assumptions:

- players choose their strategies *simultaneously*; subsequently each player receives a payoff from the resulting joint strategy,
- each player is ***rational***, which means that his objective is to maximize his payoff,
- players have ***common knowledge*** of the game and of each others' rationality.[1]

[1] Intuitively, common knowledge of some fact means that everybody knows it, everybody knows that everybody knows it, etc. It is discussed in the context of extensive games in Section 6.

Finite two-player games are usually represented in the form called a ***bimatrix***, where one assumes that the players choose independently a row or a column. Each entry represents the resulting payoffs to the row and column players. The following examples of two-player games will be relevant in the subsequent discussion.

Example 1. The following game is called ***Prisoner's Dilemma***. The strategies C and D stand for 'cooperate' and 'defect':

	C	D
C	$2, 2$	$0, 3$
D	$3, 0$	$1, 1$

The following game is called ***Matching Pennies***. The strategies H and T stand for 'head' and 'tail':

	H	T
H	$1, -1$	$-1, \ 1$
T	$-1, \ 1$	$1, -1$

□

Fix a strategic game $H := (S_1, \ldots, S_n, p_1, \ldots, p_n)$. Let $S = S_1 \times \cdots \times S_n$. We call each element $s \in S$ a ***joint strategy (of players*** $1, \ldots, n$***)***, denote the ith element of s by s_i, and abbreviate the sequence $(s_j)_{j \neq i}$ to s_{-i}. We write (s'_i, s_{-i}) to denote the joint strategy in which player's i strategy is s'_i and each other player's j strategy is s_j. Occasionally we write (s_i, s_{-i}) instead of s. Finally, we abbreviate the Cartesian product $\times_{j \neq i} S_j$ to S_{-i}.

Given a joint strategy s, we denote the sequence $(p_1(s), \ldots, p_n(s))$ by $p(s)$ and call it an ***outcome*** of the game. We say that H ***has*** k ***outcomes*** if $|\{p(s) \mid s \in S\}| = k$ and call a game ***trivial*** if it has one outcome. We say that two joint strategies s and t are ***payoff equivalent*** if $p(s) = p(t)$.

We call a strategy s_i of player i a ***best response*** to a joint strategy s_{-i} of the other players if

$$\forall s'_i \in S_i : p_i(s_i, s_{-i}) \geq p_i(s'_i, s_{-i}).$$

Next, we call a joint strategy s a ***(pure) Nash equilibrium*** if for each player i, s_i is a best response to s_{-i}, that is, if

$$\forall i \in \{1, \ldots, n\} : \forall s'_i \in S_i \ p_i(s_i, s_{-i}) \geq p_i(s'_i, s_{-i}).$$

So a joint strategy is a Nash equilibrium if no player can achieve a higher payoff by *unilaterally* switching to another strategy. Intuitively, a Nash equilibrium is a situation in which each player is a posteriori satisfied with his choice. It is often used to predict the outcomes of the strategic games.

It is easy to check that there is a unique Nash equilibrium in Prisoner's Dilemma which is (D, D), while the Matching Pennies game has no Nash equilibria.

A relevant question is whether we can identify natural subclasses of strategic games where a Nash equilibrium is guaranteed to exist. Below, we describe two such classes which are well studied in game theory. Fix till the end of this section a strategic game $H = (S_1, \ldots, S_n, p_1, \ldots, p_n)$.

We say that a pair of joint strategies (s, s') forms a ***profitable deviation*** if there exists a player i such that $s'_{-i} = s_{-i}$ and $p_i(s') > p_i(s)$. If such a pair (s, s') exists, then we say that player i can ***profitably deviate*** from s to s' and denote this by $s \to s'$. An ***improvement path*** is a maximal sequence (i.e., a sequence that cannot be extended to the right) of joint strategies in which each consecutive pair is a profitable deviation. By an ***improvement sequence*** we mean a prefix of an improvement path.

We say that H has the ***finite improvement property*** (***FIP*** in short), if every improvement path is finite. Clearly, if an improvement path is finite, then its last element is a Nash equilibrium. So if H has the FIP, then it is guaranteed to have Nash equilibrium, which explains the interest in this notion. A trivial example of a game that has the FIP is the Prisoner's Dilemma game.[2]

However, the FIP is a very strong property and several natural games with a Nash equilibrium fail to satisfy it. Young in [23] and independently Milchtaich in [14] proposed a weakening of this condition and introduced the following natural class of games. We say that H is ***weakly acyclic*** if for any joint strategy s, there exists a finite improvement path that starts at s. Consequently, every weakly acyclic game has a Nash equilibrium.

We call the function $P : S \to \mathbb{R}$ a ***weak potential*** for H if

> $\forall s$: if s is not a Nash equilibrium, then for some profitable deviation $s \to s'$, $P(s) < P(s')$.

The following natural characterization of finite weakly acyclic games was established in [15].

Theorem 1 (Weakly acyclic)**.** *A finite game is weakly acyclic iff it has a weak potential.*

Sometimes it is convenient to assume that a weak potential is a function to a strict linear ordering (that can be subsequently mapped to $\mathbb{R}$).

One way to find a Nash equilibrium in a strategic game is by using a concept of dominance. In the context of extensive games the most relevant is the notion of weak dominance. By a ***subgame*** of a strategic game H we mean a game obtained from H by removing some strategies.

[2] A more interesting class of games that have the FIP are the congestion games, see [18].

Consider two strategies s_i and s'_i of player i. We say that s_i ***weakly dominates*** s'_i (or equivalently, that s'_i is ***weakly dominated by*** s_i) in H if

$$\forall s_{-i} \in S_{-i} : p_i(s_i, s_{-i}) \geq p_i(s'_i, s_{-i}) \text{ and } \exists s_{-i} \in S_{-i} : p_i(s_i, s_{-i}) > p_i(s'_i, s_{-i}).$$

Denote by H^1 a subgame of H obtained by the elimination of some (not necessarily all) strategies that are weakly dominated in H, and put $H^0 := H$ and $H^{k+1} := (H^k)^1$, where $k \geq 1$. Note that H^k is not uniquely defined, since we do not stipulate which strategies are removed at each stage.

Abbreviate the phrase 'iterated elimination of weakly dominated strategies' to IEWDS. We say that each H^k is obtained from H ***by an IEWDS***. If for some k, some subgame H^k is a trivial game we say that H ***can be solved by an IEWDS***. The relevant result (that we shall not prove) is that if a finite strategic game H can be solved by an IEWDS then every remaining joint strategy is a Nash equilibrium of H. We shall illustrate it in Example 7 in Section 4.

The following lemma will be needed in Section 7.

Lemma 2. *Let $H := (S_1, \ldots, S_n, p_1, \ldots, p_n)$ be a finite strategic game and let $H^k := (S^k_1, \ldots, S^k_n, p_1, \ldots, p_n)$, where $k \geq 1$. Then*

$$\forall i \in \{1, \ldots, n\}\ \forall s_i \in S_i\ \exists t_i \in S^k_i\ \forall s_{-i} \in S^k_{-i} : p_i(t_i, s_{-i}) \geq p_i(s_i, s_{-i}).$$

Proof. We proceed by induction. Take some $i \in \{1, \ldots, n\}$ and $s_i \in S_i$. Suppose $k = 1$. If $s_i \in S^1_i$, then we are done, so assume that $s_i \notin S^1_i$. H is finite and the relation 'weakly dominates' is transitive, so some strategy t_i from H weakly dominates s_i in H and is not weakly dominated in H, and thus is in H^1.

Suppose the claim holds for some $k > 1$. By the induction hypothesis for some $u_i \in S^k_i$ we have $p_i(u_i, s_{-i}) \geq p_i(s_i, s_{-i})$ for all $s_{-i} \in S^k_{-i}$. If $u_i \notin S^{k+1}_i$, then for the same reasons as above some strategy t_i from H^{k+1} weakly dominates u_i in H^k and consequently $p_i(t_i, s_{-i}) \geq p_i(s_i, s_{-i})$ for all $s_{-i} \in S^k_{-i}$. □

Finally, we introduce the following condition defined in [13]. We say that a strategic game $(S_1, \ldots, S_n, p_1, \ldots, p_n)$ satisfies the ***transference of decisionmaker indifference (TDI)*** condition if:

$$\begin{aligned}&\forall i \in \{1, \ldots, n\}\ \forall r_i, t_i \in S_i\ \forall s_{-i} \in S_{-i} :\\ &p_i(r_i, s_{-i}) = p_i(t_i, s_{-i}) \rightarrow p(r_i, s_{-i}) = p(t_i, s_{-i}).\end{aligned}$$

Informally, this condition states that whenever for some player i two of his strategies r_i and t_i are indifferent w.r.t. some joint strategy s_{-i} of the other players then this indifference extends to all players.

In the next section we shall introduce a natural class of strategic games that satisfy the TDI condition.

3 Preliminaries on strictly competitive games

Sections 8 and 10 concern specific extensive games that involve two players. It is convenient to introduce them first as strategic games. A strategic two-player game is called ***strictly competitive*** if

$$\forall i \in \{1,2\}\ \forall s,t \in S\ :\ p_i(s) \geq p_i(t) \text{ iff } p_{-i}(s) \leq p_{-i}(t).$$

(As there are here just two players, $-i$ denotes the opponent of player i, so $p_{-i}(s)$ and $p_{-i}(t)$ are here numbers.) Note that every strictly competitive game satisfies the TDI condition as the definition implies that

$$\forall i \in \{1,2\}\ \forall s,t \in S\ :\ p_i(s) = p_i(t) \text{ iff } p_{-i}(s) = p_{-i}(t). \tag{1}$$

A two-player game is called ***zero-sum*** if

$$\forall s \in S\ :\ p_1(s) + p_2(s) = 0.$$

An example is the Matching Pennies game. Clearly every zero-sum game is strictly competitive.

In Section 10 we shall discuss two classes of zero-sum games. A zero-sum game is called a ***win or lose game*** if the only possible outcomes are $(1,-1)$ and $(-1,1)$, with 1 interpreted as a *win* and -1 as *losing*. Finally, a zero-sum game is called a ***chess-like game*** if the only possible outcomes are $(1,-1)$, $(0,0)$, and $(-1,1)$, with 0 interpreted as a *draw*.

The following results about strictly competitive games will be needed in Section 8.

Lemma 3. *Consider a strictly competitive strategic game H with a Nash equilibrium s. Suppose that for some $i \in \{1,2\}$, t_i weakly dominates s_i. Then (t_i, s_{-i}) is also a Nash equilibrium.*

Proof. Let $H := (S_1, S_2, p_1, p_2)$. Take a strategy s'_i of player i. By the assumptions about s and t_i

$$p_i(t_i, s_{-i}) = p_i(s_i, s_{-i}) \geq p_i(s'_i, s_{-i}).$$

Next, take a strategy s'_{-i} of player $-i$. By (1) and the fact that s is a Nash equilibrium

$$p_{-i}(t_i, s_{-i}) = p_{-i}(s_i, s_{-i}) \geq p_{-i}(s_i, s'_{-i}).$$

This establishes the claim. □

Given a finite strategic game $H := (S_1, \ldots, S_n, p_1, \ldots, p_n)$ we define for each player i

$$maxmin_i(H) := \max_{s_i \in S_i} \min_{s_{-i} \in S_{-i}} p_i(s_i, s_{-i}).$$

We call any strategy s_i^* such that $\min_{s_{-i} \in S_{-i}} p_i(s_i^*, s_{-i}) = maxmin_i(H)$ a ***security strategy*** for player i in H.

The following result goes back to [22], where it was established for zero-sum games. The formulation below is from [16, pages 22-23]).

Theorem 4 (Minimax)**.** *Suppose that s is a Nash equilibrium of a strictly competitive strategic game H. Then each s_i is a security strategy for player i and $p(s) = (maxmin_1(H), maxmin_2(H))$.*

Corollary 5. *Consider a finite strictly competitive strategic game H that has a Nash equilibrium. Then H^1 has a Nash equilibrium, as well, and for all $i \in \{1, 2\}$, $maxmin_i(H) = maxmin_i(H^1)$.*

(The notation H^1 was introduced in Section 2.)

Proof. We first prove that some Nash equilibrium of H is also a joint strategy of H^1. Let s be a Nash equilibrium of H. Suppose first that only one strategy from s, say s_i, is not a strategy in H^1. The game H is finite and the relation 'weakly dominates' is transitive so some strategy t_i weakly dominates s_i and is not weakly dominated. Thus (t_i, s_{-i}) is a joint strategy in H^1, which by Lemma 3 is a Nash equilibrium in H.

Suppose now that none of the strategies from s are strategies in H^1. By the argument just made we conclude that for some joint strategy t in H^1 first (t_i, s_{-i}) is a Nash equilibrium in H and then that t is a Nash equilibrium in H.

We conclude that a joint strategy is both a Nash equilibrium in H and in H^1. The other claim then follows by the Minimax Theorem 4. □

Lemma 6. *Consider a strictly competitive strategic game H that has a Nash equilibrium and has two outcomes. Let H^1 be the result of removing from H all weakly dominated strategies. Then H^1 is a trivial game.*

Proof. Let s^* be a Nash equilibrium of $H = (S_1, S_2, p_1, p_2)$ and s' a joint strategy such that $p(s^*)$ and $p(s')$ are the two outcomes in H. By condition (1) from Section 2 $p_1(s^*) \neq p_1(s')$ and $p_2(s^*) \neq p_2(s')$. H is strictly competitive, so for some i both $p_i(s^*) > p_i(s')$ and $p_{-i}(s') > p_{-i}(s^*)$.

First we show that $p_i(s_i^*, s_{-i}) = p_i(s^*)$ for all $s_{-i} \in S_{-i}$. Suppose otherwise. Take s_{-i} such that $p_i(s_i^*, s_{-i}) \neq p_i(s^*)$. Then $p_i(s_i^*, s_{-i}) = p_i(s')$, so by (1) $p_{-i}(s_i^*, s_{-i}) = p_{-i}(s') > p_{-i}(s^*)$, which contradicts the fact that s^* is a Nash equilibrium.

Hence by the choice of i for all $s_{-i} \in S_{-i}$

$$p_i(s_i^*, s_{-i}) = p_i(s^*) \geq p_i(s_i', s_{-i})$$

and

$$p_i(s_i^*, s_{-i}') = p_i(s^*) > p_i(s').$$

So s_i^* weakly dominates s_i'. This implies that H^1 is a trivial game. □

4 Extensive games

After these preliminaries we now focus on the subject of this tutorial.

A ***rooted tree*** (from now on, just a ***tree***) is a connected directed graph (i.e., such that the undirected version is connected) with a unique node with the in-degree 0, called the ***root***, and in which every other node has the in-degree 1. A ***leaf*** is a node with the out-degree 0. We denote a tree by (V, E, v_0), where V is a non-empty set of nodes, E is a possibly empty set of directed edges, and v_0 is the root. In drawings the edges will be directed downwards.

An ***extensive game with perfect information*** (in short, just an ***extensive game***) for $n \geq 1$ players consists of:

- a set of players $\{1, \ldots, n\}$,
- a ***game tree*** $T := (V, E, v_0)$; we denote its set of leaves by Z,
- a ***turn function*** $turn : V \setminus Z \to \{1, \ldots, n\}$,
- the ***outcome functions*** $o_i : Z \to \mathbb{R}$, for each player i.

We denote it by $(T, turn, o_1, \ldots, o_n)$.

As in the case of strategic games we assume that each player is rational (which now means that his objective is to maximize his outcome in the game) and that the players have common knowledge of the game and of each others' rationality.

A node w is called a ***child*** of v in T if $(v, w) \in E$. A node in T is called a ***preleaf*** if all its children are leaves. We say that an extensive game is ***finite*** if its game tree is finite. In what follows we limit our attention to finite extensive games.

The function *turn* determines at each non-leaf node which player should move. The edges of T represent possible ***moves*** in the considered game, while for a node $v \in V \setminus Z$ the set of its children $C(v) := \{w \mid (v, w) \in E\}$ represents possible ***actions*** of player $turn(v)$ at v.

In the figures below we identify the actions with the labels we put on the edges and thus identify each action with the corresponding move. For convenience we do not assume the labels to be unique, but it will not lead to confusion. Further, we annotate the non-leaf nodes with the identity of the player whose turn it is to move and the name of the node. Finally, we annotate each leaf node with the corresponding sequence of the values of the o_i functions.

Example 2. Consider the Prisoner's Dilemma and Matching Pennies games from Example 1. Suppose the players move sequentially with the row player moving first. The game trees of the resulting extensive games are depicted in Figures 1 and 2 below. The thick lines in the second drawing will be explained later. □

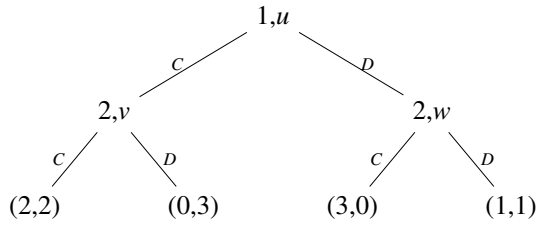


Figure 1: Extensive form of the Prisoner's Dilemma game

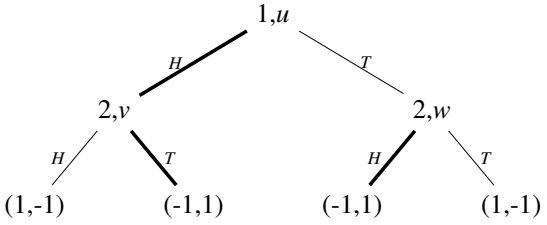


Figure 2: Extensive form of the Matching Pennies game

Example 3. The following two-player game is called the ***Ultimatum game***. Player 1 moves first and selects a number $x \in \{0, 1, \ldots, 100\}$ intepreted as a percentage of some good to be shared, leaving the fraction of $(100 - x)\%$ for the other player. Player 2 either *accepts* this decision, the outcome is then $(x, 100 - x)$, or *rejects* it, the outcome is then $(0, 0)$. The game tree is depicted in Figure 3, where the action of player 1 is a number from the set $\{0, 1, \ldots, 100\}$, and the actions of player 2 are A and R.

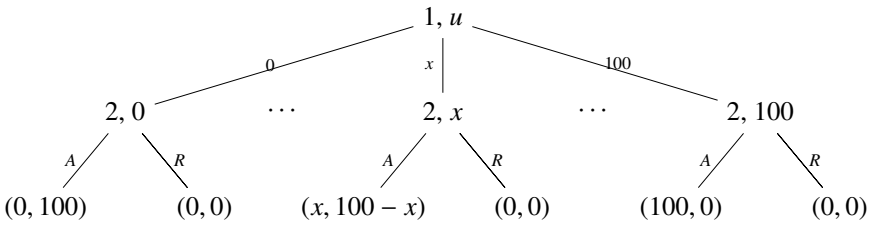


Figure 3: The Ultimatum game

□

Next we introduce strategies in extensive games. Consider a finite extensive game $G := (T, turn, o_1, \ldots, o_n)$. Let $V_i := \{v \in V \setminus Z \mid turn(v) = i\}$. So V_i is the set

of nodes at which player i moves. Its elements are called the ***decision nodes*** of player i. A ***strategy*** for player i is a function $s_i : V_i \to V$, such that $(v, s_i(v)) \in E$ for all $v \in V_i$. ***Joint strategies*** are defined as in strategic games. When the game tree consists of just one node, each strategy is the empty function, denoted by $\emptyset$, and there is only one joint strategy, namely the n-tuple of these functions.

Each joint strategy different from $(\emptyset, \ldots, \emptyset)$ assigns a unique child to every node in $V \setminus Z$. In fact, we can identify joint strategies with such assignments. Each joint strategy $s = (s_1, \ldots, s_n)$ determines a rooted path $play(s) := (v_1, \ldots, v_m)$ in T defined inductively as follows:

- v_1 is the root of T,
- if $v_k \notin Z$, then $v_{k+1} := s_i(v_k)$, where $turn(v_k) = i$.

Informally, given a joint strategy s, we can view $play(s)$ as the resulting *play* of the game.

G is finite, so for each joint strategy s the rooted path $play(s)$ is finite. Denote by $leaf(s)$ the last element of $play(s)$. We call then $(o_1(leaf(s)), \ldots, o_n(leaf(s)))$ the ***outcome*** of the game G when each player i pursues his strategy s_i and abbreviate it to $o(leaf(s))$.

Example 4. Let us return to the extensive form of the Matching Pennies game from Example 2. The strategies for player 1 are: H and T, while the strategies for player 2 are: HH, HT, TH, and TT, where for instance TH stands for a strategy that selects T at the node v and H at the node w. In Figure 2 thick lines correspond with the joint strategy (H, TH). Here $play(H, TH) = (u, v, (-1, 1))$, where we identify each leaf with the corresponding outcome, and $o(leaf(H, TH)) = (-1, 1)$. □

With each finite extensive game $G := (T, turn, o_1, \ldots, o_n)$ we associate a strategic game $\Gamma(G) := (S_1, \ldots, S_n, p_1, \ldots, p_n)$ defined as follows:

- S_i is the set of strategies of player i in G,
- $p_i(s) := o_i(leaf(s))$.

We call $\Gamma(G)$ the ***strategic form*** of G.

All notions introduced in the context of strategic games can now be reused in the context of finite extensive games simply by referring to the corresponding strategic form. This way we obtain the notions of a best response, Nash equilibrium, extensive games that have the FIP, are weakly acyclic, etc.

Example 5. The strategic form of the extensive form of the Matching Pennies game from Example 2 differs from the initial Matching Pennies game from Example 1 and looks as follows:

	HH	HT	TH	TT
H	$1, -1$	$1, -1$	$-1, \ 1$	$-1, \ 1$
T	$-1, \ 1$	$1, -1$	$-1, \ 1$	$1, -1$

Note that this game has two Nash equilibria: (H, TH) and (T, TH). The first one is depicted in Figure 2 by thick lines. By definition, these are the Nash equilibria of the extensive form of the Matching Pennies game.

The intuitive reason that there are two Nash equilibria is that no matter which out of his two strategies the first player selects, the second player can always secure the payoff 1 for himself. Of course, the decision which player moves first affects both the sets of strategies and the sets of Nash equilibria.

One can also easily check that the extensive form of the Prisoner's Dilemma game given in Figure 1 has one Nash equilibrium, (D, DD). □

Example 6. Consider now the Ultimatum game from Example 3. Each strategy for player 1 is a number from $\{0, 1, \ldots, 100\}$, while each strategy for player 2 is a function from $\{0, 1, \ldots, 100\}$ to $\{A, R\}$.

It is easy to check that $(100, s_2)$, where for $y \in \{0, 1, \ldots, 100\}$, $s_2(y) = R$ is a Nash equilibrium with the outcome $(0, 0)$ and that all other Nash equilibria are of the form (x, s_2), where $s_2(x) = A$ and $s_2(y) = R$ for $y > x$, and $x, y \in \{0, 1, \ldots, 100\}$, with the corresponding outcome $(x, 100 - x)$. □

Concepts such as Nash equilibrium can be defined directly, without a detour through the strategic games. However, introducing strategic games first allows us to view finite extensive games as a special class of strategic games and allows us to conclude that some results established for the strategic games, for instance the Weakly Acyclic Theorem 1, also hold for all finite extensive games.

As we shall see, for extensive games, due to their structure, additional results hold. Further, their structure suggests a new equilibrium notion that is meaningful only for these games. Finally, when discussing iterated elimination of weakly dominated strategies in an extensive game, one needs to reason about its strategic form.

All examples in this section are instances of the so-called ***Stackelberg competition***. In such games a *leader* moves first and a *follower*, having observed the resulting action, moves second. Of course, there are other natural extensive games, in particular multi-player games and games with infinite game trees.

For extensive games that are not Stackelberg competition games it is legitimate to question the notion of a strategy. Namely, one would expect that when a player following a strategy makes a move to a node u, then all his subsequent moves should take place in the subtree rooted at u. However, the definition of a strategy does not stipulate it as it is 'overdefined'. A natural revision was introduced in [16, page 94].

Given a node w in the game tree consider the path $v = v_1, \ldots, v_k = w$ to it from the root v. Let

$$[w]_i := \{(v_j, v_{j+1}) \mid j \in \{1, \ldots, k-1\} \text{ and } turn(v_j) = i\}.$$

Informally, $[w]_i$ is the set of the moves of player i that lie on the path from the root to w.

We call rs_i a ***reduced strategy*** of player i if it is a maximal subset of a strategy s_i (recall that each strategy is a function, so a set of pairs of nodes) that satisfies the following property:

$$\text{if } (u, w) \in rs_i, \text{ then } [w]_i \subseteq rs_i.$$

Intuitively, rs_i is a reduced strategy of player i if according to it each of his moves follows his earlier moves in the game.

Just like the joint strategies, each joint reduced strategy determines a play of the game. This allows us to define the outcome of the game when each player pursues his reduced strategy. Associate now with each joint reduced strategy r the following set of original joint strategies:

$$Str(r) := \{s \mid \forall i \in \{1, \ldots n\} : r_i \subseteq s_i\}.$$

One can show that the sets $Str(r)$, where r is a Nash equilibrium in the reduced strategies, form a partition of the set of original Nash equilibria. So each Nash equilibrium in the reduced strategies is a convenient representation of a set of Nash equilibria in the original strategies.

Example 7. Consider the classic centipede game due to [17]. In Figure 4 we present a version of the game from [16, page 107]. C and S represent the actions 'continue' and 'stop'.

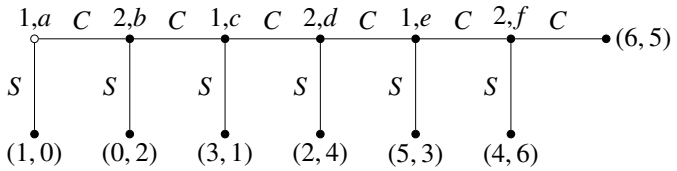


Figure 4: A six-period version of the centipede game

Each player has three decision nodes and two actions at each of them. So each player has eight strategies. In contrast, both players have four reduced strategies. These are

for player 1: aS, $aCcS$, $aCcCeS$, $aCcCeC$, and

for player 2: bS, $bCdS$, $bCdCfS$, $bCdCfC$,

where $aCcCeS$, is a shorthand for $\{(a,C),(c,C),(e,S)\}$, etc. The strategic form corresponding to the joint reduced strategies is given in Figure 5.

	bS	$bCdS$	$bCdCfS$	$bCdCfC$
aS	$1,0$	$1,0$	$1,0$	$1,0$
$aCcS$	$0,2$	$3,1$	$3,1$	$3,1$
$aCcCeS$	$0,2$	$2,4$	$5,3$	$5,3$
$aCcCeC$	$0,2$	$2,4$	$4,6$	$6,5$

Figure 5: The strategic form of the centipede game that uses reduced strategies

This game has a unique Nash equilibrium, namely (aS,bS), with the outcome $(1,0)$. It can be obtained by solving the game by an IEWDS in six steps, by repeatedly eliminating the rightmost column and the lowest row. In contrast, the strategic form corresponding to the original extensive game has several Nash equilibria. According to the notation introduced above, these Nash equilibria form the set $Str((aS,bS))$. □

5 Subgame perfect equilibria

5.1 Definition and examples

Example 6 suggests that the concept of a Nash equilibrium is not informative enough to predict outcomes of extensive games: it results in too many scenarios, some of which are obviously inferior to all players. Another issue is the problem of so-called 'not credible threat' as illustrated in the following example.

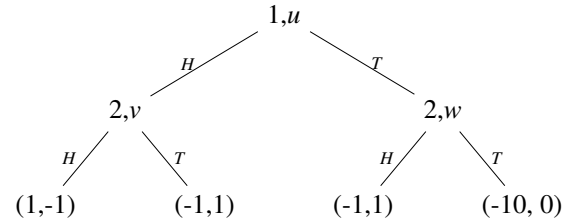


Figure 6: A modification of the Matching Pennies game

Example 8. Consider the extensive game given in Figure 6. This game has three Nash equilibria: (H, TH), (T, TH), and (H, TT). However, the last equilibrium is not plausible: if player 1 chooses T, then player 2 should select H and not T: the 'threat' of player 2 to choose T at the node 2 is not credible. □

Motivated by the issue of non-credible threats Selten introduced in [20] a stronger equilibrium concept. To define it we need to use strategies instead of the restricted strategies.

Consider an extensive game $G := (T, turn, o_1, \ldots, o_n)$ and a non-leaf node w of T. Denote by T^w the subtree of T rooted at w. The ***subgame of G rooted at the node*** w, denoted by G^w, is defined as follows:

- its set of players is $\{1, \ldots, n\}$,
- its tree is T^w,
- its turn and payoff functions are the restrictions of the corresponding functions of G to the nodes of T^w.

So the notion of a subgame has a different meaning for the strategic and for the extensive games. Note that some players may 'drop out' in G^w, in the sense that at no node of T^w it is their turn to move. Still, to keep the notation simple, it is convenient to admit in G^w all original players in G. Each strategy s_i of player i in G uniquely determines his strategy s_i^w in G^w. Given a joint strategy $s = (s_1, \ldots, s_n)$ of G we denote by s^w the joint strategy $(s_1^w, \ldots, s_n^w)$ in G^w.

A joint strategy s of G is called a ***subgame perfect equilibrium*** in G if for every node w of T, the joint strategy s^w of G^w is a Nash equilibrium in G^w. Informally s is subgame perfect equilibrium in G if it induces a Nash equilibrium in every subgame of G.

It is straightforward to check that all Nash equilibria in the extensive forms of the Prisoner's Dilemma and Matching Pennies games are also subgame perfect equilibria. The modified Matching Pennies game given in Example 8 has two subgame perfect equilibria: (H, TH) and (T, TH). Note that the Nash equilibrium (H, TT) that involves a non-credible threat is not a subgame perfect equilibrium.

Example 9. Return now to the Ultimatum game from Example 3. In Example 6 we noticed that this game has several Nash equilibria. However, only two of them are subgame perfect equilibria. They are depicted in Figures 7 and 8. In the first equilibrium player 1 selects 100 and player 2 accepts all offers, while in the second equilibrium player 1 selects 99 and player 2 accepts all offers except 100. These equilibria are more intuitive than the remaining Nash equilibria and provide natural insights into this game. □

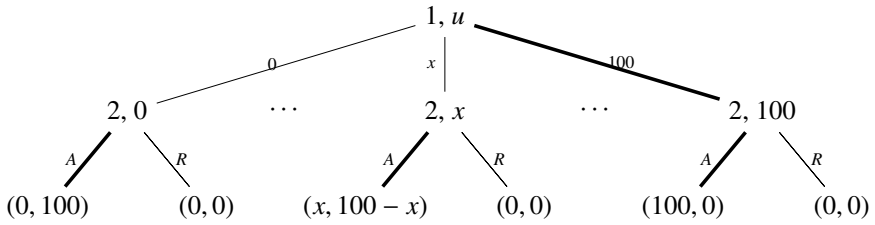


Figure 7: A subgame perfect equilibrium in the Ultimatum game

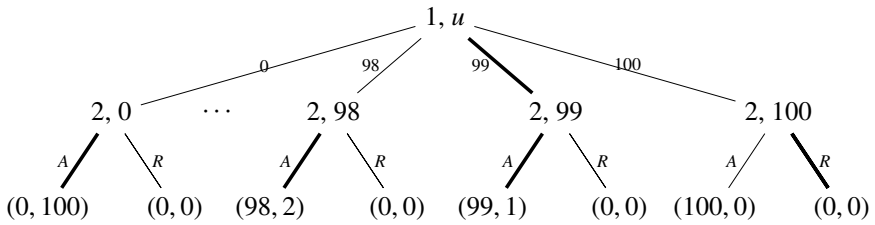


Figure 8: Another subgame perfect equilibrium in the Ultimatum game

It should be noted that the Ultimatum game has been extensively analysed in experimental economics. It has been observed that in practice, people do not often play a Nash equilibrium or a subgame perfect equilibrium.

5.2 Backward induction

We now show that for a finite extensive game G over T, one can construct a subgame perfect equilibrium using the iterative procedure described in Algorithm 1 called the ***backward induction algorithm***. Since each loop iteration of Algorithm 1 modifies the underlying game tree, we use the notation $C(v, T)$ to denote the set of children of node v in the current version of the tree T.

Note that Algorithm 1 always terminates but in general need not have a unique outcome due to the presence of the **choose** statements. Each execution (i.e., each selection of values in the **choose** statements) constructs a unique joint strategy s and an extension of the functions $o_1, \ldots, o_n$ to all nodes of the game tree.

Example 10. Consider the Ultimatum game from Example 3. Algorithm 1 generates two possible outputs that correspond to Figures 7 and 8. For the second outcome the corresponding extensions of the outcome functions to all nodes are given in Figure 9. □

Algorithm 1:

Input: A finite extensive game $G := (T, turn, o_1, \ldots, o_n)$ with $T = (V, E, v_0)$

Output: A subgame perfect equilibrium s in G and extensions of the functions $o_1, \ldots, o_n$ to all nodes of T such that $o(v_0) = o(leaf(s))$

```
1 while |V| > 1 do
2     choose v ∈ V that is a preleaf of T;
3     i := turn(v);
4     choose w ∈ C(v, T) such that o_i(w) is maximal;
5     s_i(v) := w;
6     o(v) := o(w);
7     V := V \ C(v, T);
8     E := E ∩ (V × V);
9     T := (V, E, v_0)
```

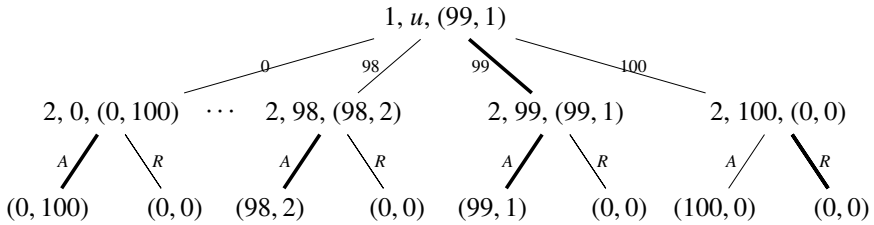


Figure 9: The backward induction algorithm and the Ultimatum game

The following characterisation result makes use of the nondeterminism present in the algorithm.

Theorem 7. *For every finite extensive game all possible executions of the backward induction algorithm generate precisely all subgame perfect equilibria.*

To establish this result we shall need a preparatory lemma, called the 'one deviation property' (see, e.g., [16, page 98]). Recall that for a function $f : X \to Y$ (with $X \neq \emptyset$), $\mathrm{argmax}_{x \in X} f(x) := \{y \in X \mid f(y) = \max_{x \in X} f(x)\}$.

Lemma 8. *Let G be a finite extensive game over the game tree T. A joint strategy s is a subgame perfect equilibrium in G iff for all non-leaf nodes u in T*

$$s_i(u) \in \mathrm{argmax}_{x \in C(u)} o_i(leaf(s^x)), \text{ where } i = turn(u).$$

In words, this condition states that for all non-leaf nodes u in T and $i = turn(u)$, $s_i(u)$ selects a child x of u for which $o_i(leaf(s^x))$ is maximal.

For a proof see, e.g, [16, pages 98-99] or a more detailed presentation in the appendix of [2].

Corollary 9 ([2], Corollary 7)**.** *Let G be a finite extensive game over the game tree T with the root v. A joint strategy s is a subgame perfect equilibrium in G iff for all $u \in C(v)$*

- $s_i(v) \in \mathrm{argmax}_{x \in C(v)} o_i(leaf(s^x))$, *where* $i = turn(v)$,
- s^u *is a subgame perfect equilibrium in the subgame* G^u.

Intuitively, the first condition states that among the subgames rooted at the children of the root v, the one determined by the first move in the game G yields the best outcome for the player who moved.

Proof of Theorem 7. The proof proceeds by induction on $height(T)$, defined as the number of edges in the longest path in the tree T. The base case when $height(T) = 1$ is straightforward. Suppose $height(T) > 1$. Let $C(v) = \{w_1, \ldots w_k\}$.
($\Rightarrow$) Consider a joint strategy s in G together with some extensions of the functions $o_1, \ldots, o_n$ to the nodes of the game tree of G that are generated by an execution of Algorithm 1.

Fix an arbitrary $l \in \{1, \ldots, k\}$. Delete in this execution the **while** loop iterations that do not involve the nodes of the game tree T^{w_l} of the subgame G^{w_l} and use at the beginning the game tree T^{w_l} instead of T. This way we obtain an execution of Algorithm 1 applied to the game G^{w_l} that generates a joint strategy s^{w_l} in G^{w_l} together with some extensions of the functions $o_1, \ldots, o_n$ to the nodes of the game

tree T^{w_l}. By the induction hypothesis s^{w_l} is a subgame perfect equilibrium in G^{w_l} and $o(w_l) = o(leaf(s^{w_l}))$.

Consider now the last iteration of the **while** loop in the original execution of Algorithm 1. At this stage $V = \{v_0, w_1, \ldots, w_k\}$, so before line **2** we have $C(v_0, T) = \{w_1, \ldots, w_k\}$. After line **3** we have $i = turn(v_0)$ and after line **4** w is such that $w \in \{w_1, \ldots, w_k\}$ and $o_i(w) \geq o_i(w_l)$ for all $l \in \{1, \ldots, k\}$.

By the previous conclusion for all $l \in \{1, \ldots, k\}$ we have $o_i(w_l) = o_i(leaf(s^{w_l}))$, so for all $l \in \{1, \ldots, k\}$ we have $o_i(leaf(s^w)) \geq o_i(leaf(s^{w_l}))$. After line **5** we have $s_i(v_0) = w$, so by Lemma 8 s is a subgame perfect equilibrium. Finally, after line **6** we have $o(v_0) = o(leaf(s^w)) = o(leaf(s))$.

($\Leftarrow$) Suppose that s is a subgame perfect equilibrium in G. We show that it can be generated by Algorithm 1 together with the extensions of the functions $o_1, \ldots, o_n$ to all nodes of T such that $o(v_0) = o(leaf(s))$.

Fix an arbitrary $l \in \{1, \ldots, k\}$. By the assumption on s, the joint strategy s^{w_l} is a subgame perfect equilibrium in the subgame G^{w_l}, so by the induction hypothesis some execution of Algorithm 1 applied to the subgame G^{w_l} generates s^{w_l} together with the extensions of the functions $o_1, \ldots, o_n$ to the nodes of the game tree of G^{w_l} such that $o(w_l) = o(leaf(s^{w_l}))$.

Using these k executions of Algorithm 1 applied to the subgames $G^{w_1}, \ldots, G^{w_k}$ we construct the desired execution of Algorithm 1 applied to the game G as follows. First we 'glue' these k executions into one but using at the beginning of the execution the game tree T instead of the game tree of G^{w_1} and using at the beginning of each subsequent execution the current tree T instead of the game tree of the considered subgame.

After these k executions glued together $V = \{v_0, w_1, \ldots, w_k\}$, so before line **2** we have $C(v, T) = \{w_1, \ldots, w_k\}$, in line **2**, v_0 is selected and after line **3** we have $i = turn(v_0)$.

By the induction hypothesis for all $l \in \{1, \ldots, k\}$ we have $o(w_l) = o(leaf(s^{w_l}))$, so by Lemma 8 $w = s_i(v_0)$ is a node from $\{w_1, \ldots, w_k\}$ such that $o_i(w)$ is maximal. So in line **4** we can select this node w, which ensures that the assignment in line **5** is consistent with the original joint strategy s. Further, the assignment in line **6** ensures that $o(v_0) = o(w) = o(leaf(s^w)) = o(leaf(s))$. □

Corollary 10 ([11])**.** *Every finite extensive game (with perfect information) has a subgame perfect equilibrium (and hence a Nash equilibrium).*

We presented backward induction as a nondeterministic algorithm, but one can go even further. Exploiting the fact that the children of each node can be dealt with independently, we can present it as an algorithm that uses nested parallelism. In such an algorithm there is no need to modify the game tree. Given a non-leaf node v we define a nondeterministic program $Seq(v)$ by

$i := turn(v)$;
choose $w \in C(v)$ such that $o_i(w)$ is maximal;
$s_i(v) := w$;
$o(v) := o(w)$

Then for a preleaf node v we define $Comp(v)$ as $Seq(v)$ and for each node v that is neither a preleaf or a leaf we define $Comp(v)$ by

$$[||_{w \in C(v)} Comp(w)]; Seq(w)$$

where $[||_{w \in C(v)} Comp(w)]$ stands for a parallel composition of the programs $Comp(w)$ for $w \in C(v)$. So each such node v is processed with only after its children have been processed and these children are processed in an arbitrary order.

Then $Comp(v_0)$ is the desired parallel version of the backward induction algorithm.

5.3 Special classes of extensive games

It is natural to study conditions under which an extensive game has a unique subgame perfect equilibrium. The following property was introduced in [4]. We say that an extensive game is ***without relevant ties*** if for all non-leaf nodes u in T the function o_i, where $turn(u) = i$, is injective on the leaves of T^u. This is more general than saying that a game is ***generic***, which means that each o_i is an injective function.

Corollary 11. *Every finite extensive game without relevant ties has a unique subgame perfect equilibrium.*

In particular, every finite generic extensive game has a unique subgame perfect equilibrium.

Proof. If a game is without relevant ties, then so is every subgame of it. This allows us to proceed by induction on the *height* of the game tree. Let G be a finite extensive game without relevant ties over a game tree T. If $height(T) = 0$ the claim clearly holds. Suppose that $height(T) > 0$. Let v be the root of T and $i = turn(v)$.

By the induction hypothesis for each $w \in C(v)$ there is exactly one subgame perfect equilibrium t^w in G^w. Let $t = \times_{w \in C(v)} t^w$. Then for different $w, w' \in C(v)$, $leaf(t^w)$ and $leaf(t^{w'})$ are different leaves of the game tree of G. Since G is without relevant ties, $o_i(leaf(t^w)) \neq o_i(leaf(t^{w'}))$.

This means that the function $g : C(v) \to \mathbb{R}$ defined by $g(w) := o_i(leaf(t^w))$ is injective. Consequently the set $\operatorname{argmax}_{w \in C(v)} o_i(leaf(t^w))$ has a unique element and hence by Corollary 9, G has exactly one subgame perfect equilibrium. □

Note that the centipede game from Example 7 is generic, so by Corollary 11 it has exactly one subgame perfect equilibrium. To determine it we can use the observation there established, namely that in every Nash equilibrium both players select S at the nodes a and b, respectively. Indeed, by the structure of the game this observation also holds for every subgame. It follows that in the unique subgame perfect equilibrium both players select S at all non-leaf nodes. This counterintuitive form of the subgame perfect equilibrium in this game is sometimes used to question the adequacy of this solution concept.

It is also natural to study conditions under which the subgame perfect equilibria are payoff equivalent. The following theorem is implicit in [13]. The TDI condition was introduced in Section 2 when discussing strategic games.

Theorem 12. *In every finite extensive game that satisfies the TDI condition all subgame perfect equilibria are payoff equivalent.*

Proof. Consider a finite extensive game $G := (T, turn, o_1, \ldots, o_n)$ that satisfies the TDI condition. We proceed by induction on the number of nodes in the game tree. The claim holds when the game tree has just one node, since there is then only one subgame perfect equilibrium. Suppose the game tree has more than one node.

Consider two subgame perfect equilibria s and t in G. Take a preleaf v in T. Suppose $s_i(v) = w_1$ and $t_i(v) = w_2$, where $i = turn(v)$. By Corollary 9 $w_1, w_2 \in \mathrm{argmax}_{x \in C(v)} o_i(x)$.

Case 1. $leaf(s) = w_1$ and $leaf(t) = w_2$.

Take a strategy s_i' that differs from s_i only for the node v to which it assigns w_2. Then $leaf(s_i', s_{-i}) = w_2$, so $o_i(leaf(s)) = o_i(w_1) = o_i(w_2) = o_i(leaf(s_i', s_{-i}))$. Hence by the TDI property $o(leaf(s)) = o(leaf(s_i', s_{-i}))$, so $o(leaf(s)) = o(leaf(t))$.

Case 2. $leaf(s) \neq w_1$ or $leaf(t) \neq w_2$.

Without loss of generality suppose that $leaf(t) \neq w_2$. Consider the game $G' := (T', turn, o_1, \ldots, o_n)$ obtained from G by setting $o(v)$ to $o(w_1)$ and by removing all the children of v. So in G the node v is a preleaf, while in G' it is a leaf with the outcome $o(w_1)$. G' also satisfies the TDI condition since all its outcomes are also outcomes of G.

Let s' and t' be joint strategies in G' obtained from s and t by dropping v from the domains of s_i and t_i. Then both s' and t' are subgame perfect equilibria in G'. (We leave the proof of this fact to the reader.)

We have $o(leaf(s)) = o(leaf(s'))$ and by assumption the node v does not lie on the path $play(t)$, so $leaf(t) = leaf(t')$. Hence $o(leaf(s)) = o(leaf(t))$ by the induction hypothesis. □

6 Backward induction and common knowledge of rationality

Recall that player's rationality in an extensive game means that his objective is to maximize his outcome in the game. Backward induction is a natural procedure and it is natural to inquire whether it can be justified by appealing to players' rationality.

In this section we discuss Aumann's result [3] that for a natural class of extensive games common knowledge of players' rationality implies that the game reaches the backward induction outcome.

To formulate this result we introduce first Aumann's approach to modeling knowledge in the context of extensive games. Fix a finite extensive game with no relevant ties $G := (T, turn, o_1, \ldots, o_n)$ with $T = (V, E, v_0)$. Let $S_1, \ldots, S_n$ be the respective sets of strategies of players $1, \ldots, n$.

A ***knowledge system*** for G consists of

- a non-empty set Ω of ***states***,
- a function $\mathbf{s} : \Omega \to S_1 \times \cdots \times S_n$,
- for each player i a partition P_i of Ω.

One possible interpretation of a state is that it represents a 'situation', in which complete information about the players' strategies is available. This information is provided by means of the function $\mathbf{s}$.

Given a knowledge system, player i does not know the actual state ω but he knows the element of the partition P_i to which ω belongs. This interpretation suggests the following assumption.

Define for player i the function $\mathbf{s}_i : \Omega \to S_i$ by

$$\mathbf{s}_i(\omega) := s_i, \text{ where } s_i \text{ is the } i\text{th component of } \mathbf{s}(\omega).$$

Then we assume that for each player i the function $\mathbf{s}_i$ is constant on each element of the partition P_i. Intuitively, it means that in the assumed knowledge system each player knows his strategy.

We first introduce concepts that do not rely on the function $\mathbf{s}$. By an ***event*** we mean a subset of Ω. For an event E and player i we define the event K_iE by

$$K_iE := \bigcup \{L \in P_i \mid L \subseteq E\}.$$

Intuitively, K_iE is the event that player i *knows* E.

Next, we define

$$KE := K^1E := \bigcap_{i=1}^{n} K_iE,$$

inductively for $k \geq 1$

$$K^{k+1}E := K^kE,$$

and finally

$$CKE := \bigcap_{k=1}^{\infty} K^kE.$$

Intuitively, KE is the event that all players *know* the event E and CKE is the event that there is *common knowledge* of the event E among all players.

Using the function $\mathbf{s}$ we now formalize the notion that player i is rational. To start with, given a node v at which player i moves, his strategy t_i, and the function $\mathbf{s}_{-i} : \Omega \to S_{-i}$ defined in the expected way, we denote by

$$[o_i(leaf((\mathbf{s}_{-i}, t_i)^v)) > o_i(leaf(\mathbf{s}^v))]$$

the event

$$\{\omega \in \Omega \mid o_i(leaf((\mathbf{s}_{-i}(\omega), t_i)^v)) > o_i(leaf(\mathbf{s}(\omega)^v)\}.$$

It states that in the subgame G^v the outcome for player i is higher if he selects strategy t_i instead of the strategy he chooses according to $\mathbf{s}$.

Similarly, for a joint strategy t we denote by

$$[\mathbf{s} = t]$$

the event

$$\{\omega \in \Omega \mid \mathbf{s}(\omega) = t\}.$$

Recall now that for a player i we denoted by V_i the set of nodes at which he moves. We define

$$R_i := \bigcap_{v \in V_i} \bigcap_{t_i \in S_i} \neg K_i[o_i(leaf((\mathbf{s}_{-i}, t_i)^v)) > o_i(leaf(\mathbf{s}^v))].$$

where $\neg$ denotes complementation w.r.t. Ω. Intuitively, this event states that for all nodes v at which player i moves and all his strategies t_i, player i does not know whether t_i would yield a higher outcome than the strategy he chooses according to $\mathbf{s}$. So R_i is the event formalizing that player i is *rational*.

Finally, we define

$$R := \bigcap_{i=1}^{n} R_i.$$

Intuitively, R is the event that each player rational.

We still need to formalize the event that the outcome of the game is prescribed by the backward induction. To this end Aumann assumes that the game is generic, so that the game has a unique subgame perfect equilibrium, but thanks to Corollary 11 it suffices to assume that the game is without relevant ties. Then the backward induction has a unique outcome which is the subgame perfect equilibrium of the game. Denote the latter by s^*. The intended event I is then defined by

$$I := [\mathbf{s} = s^*].$$

We can now state the main result of [3].

Theorem 13. *Consider a finite extensive game G without relevant ties. Then*

$$CKR \subseteq I.$$

This inclusion formalizes the announced statement that common knowledge of players' rationality implies that the backward induction yields the outcome of the game.

The proof of the theorem relies on a number of simple properties of the operators K_i and CK that we list without proof in the following lemma.

Lemma 14.

(i) $CKE = K_i CKE$.

(ii) *If* $E \subseteq F$, *then* $K_i E \subseteq K_i F$.

(iii) $K_i E \cap K_i F = K_i(E \cap F)$.

(iv) $CKE \subseteq E$.

(v) $K_i \neg K_i E = \neg K_i E$.

(vi) $K_i E \subseteq E$.

Given a joint strategy s and a node v that is not a leaf, we define

$$s(v) := s_i(v),$$

where $i = turn(v)$. So if s is the used joint strategy, then $s(v)$ is the move resulting from it at node v. For each such node v we define the function $\mathbf{s}(v) : \Omega \to V$ in the expected way.

We shall also need the following two observations concerning players' knowledge the proofs of which we omit.

Lemma 15.

(i) For all $t_i \in S_i$, $[\mathbf{s}_i = t_i] \subseteq K_i[\mathbf{s}_i = t_i]$.

(ii) For all $v \in V_i$, $I^v \subseteq K_i I^v$, *where* $I^v := [\mathbf{s}(v) = s^*(v)]$.

Intuitively, (*i*) states that if player i chooses the strategy t_i he knows that he chooses it and (*ii*) states that if player i chooses the move $s^*(v)$ at the node v, then he knows this. Note that by Lemma 14(*vi*) we can replace in (*i*) and (*ii*) $\subseteq$ by $=$.

Proof of Theorem 13.

We have $I = \bigcap_{v \in V \setminus Z} I^v$, so it suffices to prove that for all $v \in V \setminus Z$, $CKR \subseteq I^v$. Given two nodes v and w we write $w < v$ if w is a (possibly indirect) descendant of v.

We proceed by induction w.r.t. $<$. Take a node v and suppose that $CKR \subseteq I^w$ for all $w < v$. Let $i = turn(v)$. For a joint strategy s denote by $s^{<v}$ the joint strategy s^v with the pair $(v, s_i(v))$ removed from s_i, and define the function $\mathbf{s}^{<v} : \Omega \to S_1 \times \cdots \times S_n$ by

$$\mathbf{s}^{<v}(\omega) := \mathbf{s}(\omega)^{<v}.$$

By Lemma 14(*i*) and (*ii*) and the induction hypothesis $CKR = K_i CKR \subseteq K_i I^w$ for all $w < v$, so by Lemma 14(*iii*)

$$CKR \subseteq \bigcap_{w<v} K_i I^w = K_i \bigcap_{w<v} I^w = K_i[\mathbf{s}^{<v} = (s^*)^{<v}]. \tag{2}$$

Also, by Lemma 14(*iv*) and the definition of R_i with t_i set to s_i^*

$$CKR \subseteq R \subseteq R_i \subseteq \neg K_i[o_i(leaf((\mathbf{s}_{-i}, s_i^*)^v)) > o_i(leaf(\mathbf{s}^v))]. \tag{3}$$

Further, by Lemma 14(*iii*) and the fact that since $i = turn(v)$,

$$\begin{aligned}
& K_i[\mathbf{s}^{<v} = (s^*)^{<v}] \cap K_i[o_i(leaf((\mathbf{s}_{-i}, s_i^*)^v)) > o_i(leaf(\mathbf{s}^v))] \\
= \;& K_i[\mathbf{s}^{<v} = (s^*)^{<v} \wedge o_i(leaf((\mathbf{s}_{-i}, s_i^*)^v)) > o_i(leaf(\mathbf{s}^v))] \\
= \;& K_i[\mathbf{s}^{<v} = (s^*)^{<v} \wedge o_i(leaf((s^*)^v)) > o_i(leaf((s_{-i}^*, \mathbf{s}_i)^v))] \\
= \;& K_i[\mathbf{s}^{<v} = (s^*)^{<v}] \cap K_i[o_i(leaf((s^*)^v)) > o_i(leaf((s_{-i}^*, \mathbf{s}_i)^v))],
\end{aligned}$$

so by taking complement w.r.t. $K_i[\mathbf{s}^{<v} = (s^*)^{<v}]$

$$\begin{aligned}
& K_i[\mathbf{s}^{<v} = (s^*)^{<v}] \cap \neg K_i[o_i(leaf((\mathbf{s}_{-i}, s_i^*)^v)) > o_i(leaf(\mathbf{s}^v))] \\
= \;& K_i[\mathbf{s}^{<v} = (s^*)^{<v}] \cap \neg K_i[o_i(leaf((s^*)^v)) > o_i(leaf((s_{-i}^*, \mathbf{s}_i)^v))] \\
\subseteq \;& \neg K_i[o_i(leaf((s^*)^v)) > o_i(leaf((s_{-i}^*, \mathbf{s}_i)^v))].
\end{aligned} \tag{4}$$

Finally, by (2)–(4), the fact that for each node v, s^v is a unique subgame perfect equilibrium in G^v, Lemma 15, and Lemma 14(*v*) and (*vi*)

$$\begin{aligned}
& CKR \subseteq K_i[\mathbf{s}^{<v} = (s^*)^{<v}] \cap \neg K_i[o_i(leaf((\mathbf{s}_{-i}, s_i^*)^v)) > o_i(leaf(\mathbf{s}^v))] \\
\subseteq \;& \neg K_i[o_i(leaf((s^*)^v)) > o_i(leaf((s_{-i}^*, \mathbf{s}_i)^v))] = \neg K_i[\mathbf{s}(v) \neq s^*(v)] \\
= \;& \neg K_i \neg I^v = \neg K_i \neg K_i I^v = \neg\neg K_i I^v = K_i I^v \subseteq I^v,
\end{aligned}$$

as desired. □

We conclude by the following observation of [3] showing that non-trivial knowledge systems can be easily constructed.

Note 16. *For every finite extensive game without relevant ties there exists a knowledge system such that $CKR \neq \emptyset$.*

Proof. It suffices to choose Ω to be a singleton set $\{\omega\}$ and set $\mathbf{s}(\omega) := s^*$, where s^* is the unique subgame perfect equilibrium of the considered game. Then $CKR = \Omega$. □

Aumann's paper dealt with concepts also studied by philosophers and psychologists. As a result it became highly influential and attracted wide attention. In particular Stalnaker pointed out in [21] that Aumann's notion of rationality involves reasoning about situations (nodes) that the agent knows will never be reached and constructed a model in which common knowledge of players' rationality does not imply that the game reaches the backward induction outcome.

The apparent contradiction between Aumann's and Stalnaker's results was clarified by Halpern in [8]. The difference can be explained by adding to Aumann's knowledge system for an extensive game one more parameter, a function

$$f : \Omega \times V \setminus Z \to \Omega,$$

that for a given state ω and a non-leaf node v yields a state ω' that is 'nearest' (in a well-defined sense) to v and is such that v is *reached* in ω', i.e., is such that v lies on $play(\mathbf{s}(\omega'))$.

Then according to Stalnaker, a player i is *substantively rational* in a state ω, if for each node $v \in V_i$ he is rational in the state $\omega' = f(\omega, v)$, where the latter means that

$$\omega' \in \bigcap_{t_i \in S_i} \neg K_i[o_i(leaf((\mathbf{s}_{-i}, t_i)^v)) > o_i(leaf(\mathbf{s}^v))].$$

Stalnaker's model refers to substantive rationality and not rationality.

7 Weak dominance and backward induction

Iterated elimination of weakly dominated strategies is defined for strategic games, so it can be also applied to the strategic forms of extensive games. For the class of finite extensive games discussed in the previous section this procedure is closely related to backward induction. The aim of this section is to clarify this relation.

The following notion will be needed. Consider a node w in the game tree of an extensive game G such that $turn(w) = i$. We say that a strategy s_i of player i ***can reach*** w if for some s_{-i} the node w lies on the path $play(s_i, s_{-i})$.

We begin by introducing Algorithm 2 that is a modification of the backward induction algorithm 1 from Section 5 in which the input and output are modified and line **7** is added.

Algorithm 2:

Input: A finite extensive game with no relevant ties
$G := (T, turn, o_1, \ldots, o_n)$ with $T = (V, E, v_0)$ and
$\Gamma(G) = (S_1, \ldots, S_n, p_1, \ldots, p_n)$.

Output: The subgame perfect equilibrium s in G, extensions of the functions $o_1, \ldots, o_n$ to all nodes of T such that $o(v_0) = o(leaf(s))$, and a trivial strategic game $(S_1, \ldots, S_n, p_1, \ldots, p_n)$ that includes s.

1 **while** $|V| > 1$ **do**
2 **choose** $v \in V$ that is a preleaf of T;
3 $i := turn(v)$;
4 **choose** $w \in C(v, T)$ such that $o_i(w)$ is maximal;
5 $s_i(v) := w$;
6 $o(v) := o(w)$;
7 $S_i := S_i \setminus \{s_i' \in S_i \mid s_i' \text{ can reach } v \text{ and } s_i'(v) \neq w\}$;
8 $V := V \setminus C(v, T)$;
9 $E := E \cap (V \times V)$;
10 $T := (V, E, v_0)$

The following theorem makes precise the mentioned relation between two concepts.

Theorem 17. *Consider a finite extensive game G without relevant ties and Algorithm 2 applied to it.*

*(i) Each strategy removed in line **7** is weakly dominated in the current version of the strategic game.*

(ii) The strategic game $(S_1, \ldots, S_n, p_1, \ldots, p_n)$ which is generated upon termination of the algorithm is trivial and includes the subgame perfect equilibrium of G.

Proof. Below s is the subgame perfect equilibrium of the game G. Consider the assertion $I(u)$ defined by

$$I(u) \equiv \forall t \in S \ : [\text{if } u \text{ appears in } play(t) \text{ then } o(leaf(s^u)) = o(leaf(t^u))],$$

where $S = S_1 \times \cdots \times S_n$.

First notice that if during an execution of the algorithm for some node u the assertion $I(u)$ becomes true, then it remains true. The reason is that the considered set S of joint strategies never increases.

We now show that after each loop iteration $I(v)$ holds, where v is the node being dealt with in current loop iteration and S refers to the current set of joint strategies. Fix an execution of the algorithm. We proceed by induction on the order in which the nodes of T are selected in line **2**. Consider first a preleaf v in the original game tree T. Take $t \in S$ such that v appears in $play(t)$ and let $i = turn(v)$. Then

$$\begin{array}{ll} & o(leaf(t^v)) \\ = & \quad \{ \text{ by line } \mathbf{7}\ t_i(v) = w \} \\ & o(w) \\ = & \quad \{ \text{ by line } \mathbf{5}\ s_i(v) = w \} \\ & o(leaf(s^v)). \end{array}$$

Next, consider a node v in the original game tree T selected in line **2** that is neither a preleaf nor a leaf and consider the program state after the current loop iteration. Then both $i = turn(v)$ and $s_i(v) = w$.

By the order in which the nodes are selected in line **2**, all nodes $u \in C(v)$ have been dealt with in earlier loop iterations. So by the induction hypothesis $I(u)$ holds for $u \in C(v)$. In particular $I(w)$ holds. Take $t \in S$ such that v appears in $play(t)$. Then

$$\begin{array}{ll} & o(leaf(t^v)) \\ = & \quad \{ \text{ by line } \mathbf{7}\ t_i(v) = w \} \\ & o(leaf(t^w)) \\ = & \quad \{\ I(w))\ \} \\ & o(leaf(s^w)) \\ = & \quad \{ \text{ by line } \mathbf{5}\ s_i(v) = w \} \\ & o(leaf(s^v)). \end{array}$$

(i) Fix the program state after an arbitrary loop iteration of the algorithm with the current values of $v, w, i, S_1, \ldots, S_n$. In particular $turn(v) = i$.

We first prove that for $u \in C(v)$, $u \neq w$

$$o_i(leaf(s^w)) > o_i(leaf(s^u)). \tag{5}$$

Let t_i be the strategy of player i that differs from s_i only for the node v to which it assigns u. We have $s_i(v) = w$ and by definition s^v is a Nash equilibrium in the subgame G^v, so

$$o_i(leaf(s^w)) = o_i(leaf(s^v)) \geq o_i(leaf(t_i^v, s_{-i}^v)) = o_i(leaf(t_i^u, s_{-i}^u)) = o_i(leaf(s^u)).$$

But $turn(v) = i$, $u, w \in C(v)$, $u \neq w$, and G^v is without relevant ties, so (5) follows.

Take now a strategy s_i' removed in line **7** and suppose $s_i'(v) = u$. Consider the strategy t_i that differs from s_i' only for the node v to which it assigns w (i.e., $s_i(v)$). We claim that after line **7** t_i weakly dominates s_i' in the current version of $(S_1, \ldots, S_n, p_1, \ldots, p_n)$.

Take some $t_{-i} \in S_{-i}$. If $v \notin play(t_i, t_{-i})$, then $v \notin play(s_i', t_{-i})$, so $play(t_i, t_{-i}) = play(s_i', t_{-i})$ and hence

$$p_i(t_i, t_{-i}) = o_i(leaf(t_i, t_{-i}) = o_i(leaf(s_i', t_{-i}) = p_i(s_i', t_{-i}).$$

If $v \in play(t_i, t_{-i})$, then also $w \in play(t_i, t_{-i})$, $v \in play(s_i', t_{-i})$ and $u \in play(s_i', t_{-i})$, where, recall, $s_i'(v) = u$. Since $u, w \in C(v)$, these two nodes have been dealt with in earlier loop iterations. So both $I(u)$ and $I(w)$ hold.

Hence

$$p_i(t_i, t_{-i}) = o_i(leaf(t_i, t_{-i})) = o_i(leaf((t_i, t_{-i})^w)) = o_i(leaf(s^w))$$

and

$$p_i(s_i', t_{-i}) = o_i(leaf(s_i', t_{-i})) = o_i(leaf((s_i', t_{-i})^u)) = o_i(leaf(s^u)).$$

So $p_i(t_i, t_{-i}) > p_i(s_i', t_{-i})$ by (5).

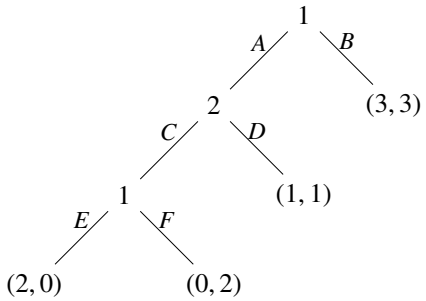
Now, t_i does not need to be a strategy from S_i but thanks to Lemma 2 we can conclude that a strategy t_i' in S_i exists that weakly dominates s_i' in the current version of $(S_1, \ldots, S_n, p_1, \ldots, p_n)$.

(*ii*) Upon termination of the algorithm $I(v_0)$, i.e., $\forall t \in S : o(leaf(s)) = o(leaf(t))$ holds. This means that upon termination the final game $(S_1, \ldots, S_n, p_1, \ldots, p_n)$ is trivial. Further, for each player each strategy removed in line **7** differs from his strategy in the subgame equilibrium, which means that the final game includes this equilibrium. □

This theorem shows that every finite extensive game G without relevant ties can be solved by an IEWDS. Recall from Corollary 11 that such a game has a unique subgame perfect equilibrium. However, not all instances of the IEWDS behave the desired way. The following example, taken from [16, page 109], shows that some generic extensive games can be solved by an IEWDS that removes the unique subgame perfect equilibrium. This explains why in line **7** in Algorithm 2 only specific weakly dominated strategies are removed.

Example 11. Consider the two-player generic extensive game and its associated strategic game given in Figure 10. In the figure, the nodes are labelled with the player whose turn it is to move.

Consider now an IEWDS that consists of the following sequence of elimination of weakly dominated strategies: AE, D, AF. The resulting trivial subgame has two joint strategies (BE, C), (BF, C). So this instance of the IEWDS eliminates (BE, D), which is the unique subgame perfect equilibrium. □

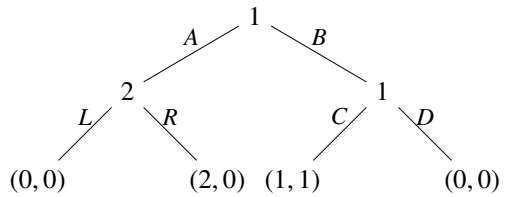


	C	D
AE	$2,0$	$1,1$
AF	$0,2$	$1,1$
BE	$3,3$	$3,3$
BF	$3,3$	$3,3$

Figure 10: An extensive game (left) and its associated strategic game (right).

8 Weak dominance and strictly competitive games

We now continue an account of iterated elimination of weakly dominated strategies. In Theorem 17 we showed that each finite extensive game without relevant ties can be solved by an IEWDS. A natural question is whether we can extend this result to arbitrary finite extensive games. The following example taken from [16, pages 109-110] shows that this fails to be the case already for two-player games.



	L	R
AC	$0,0$	$2,0$
AD	$0,0$	$2,0$
BC	$1,1$	$1,1$
BD	$0,0$	$0,0$

Figure 11: An extensive game (left) and its associated strategic game (right).

Example 12. Consider the two-player extensive game and its associated strategic game given in Figure 11. In the game tree, the nodes are labelled with the player whose turn it is to move. For this game there is just one instance of IEWDS which consists of eliminating the strategy BD. The resulting subgame is not trivial, so no instance of IEWDS can solve this extensive game. □

On the other hand, as shown in [7], finite extensive zero-sum games can be solved by an IEWDS in which at each step all weakly dominated strategies are removed. The aim of this section is to present this result for the larger class of finite extensive strictly competitive games for which the same proof remains valid.

From now on, given a strategic game H we denote by H^1 a subgame of H obtained by the elimination of *all* strategies that are weakly dominated in H, and put $H^0 := H$ and $H^{k+1} := (H^k)^1$, where $k \geq 1$. So, in contrast to Sections 2 and 3 each H^k is now uniquely defined.

Below for a strategic game H we denote by H_i the set of strategies of player i. Also, for a finite extensive game G we write $\Gamma^k(G)$ instead of $(\Gamma(G))^k$, $\Gamma_i(G)$ instead of $(\Gamma(G))_i$, and $\Gamma_i^k(G)$ instead of $(\Gamma^k(G))_i$. In particular $\Gamma^0(G) = \Gamma(G)$.

Further, for a finite strictly competitive strategic game $H = (S_1, S_2, p_1, p_2)$ we define for each player i

$$\begin{array}{l} p_i^{max}(H) := \max_{s \in S} p_i(s), \\ win_i(H) := \{s_i \in S_i \mid \forall s_{-i} \in S_{-i}\ p_i(s_i, s_{-i}) = p_i^{max}(H)\}, \\ lose_{-i}(H) = \{s_{-i} \in S_{-i} \mid \exists s_i \in S_i\ p_i(s_i, s_{-i}) = p_i^{max}(H)\}. \end{array}$$

So $p_i^{max}(H)$ is the maximal payoff player i can receive in the game H, $win_i(H)$ is the set of strategies of player i for which he always gets $p_i^{max}(H)$, while $lose_{-i}(H)$ is the set of strategies of player $-i$ for which his opponent i can get his maximally possible payoff $p_i^{max}(H)$.

The following lemma, with a rather involved proof, is crucial.

Lemma 18. *Let G be a finite strictly competitive extensive game. For all $i \in \{1, 2\}$ and for all $k \geq 0$, if $win_i(\Gamma^k(G)) = \emptyset$ then $lose_{-i}(\Gamma^k(G)) \cap \Gamma_{-i}^{k+2}(G) = \emptyset$.*

This lemma implies that if for all $i \in \{1, 2\}$, $win_i(\Gamma^k(G)) = \emptyset$ then two further rounds of eliminations of weakly dominated strategies remove from $\Gamma^k(G)$ at least two outcomes.

Proof. Fix i and k and suppose $win_i(\Gamma^k(G)) = \emptyset$. So for all $s_i \in \Gamma_i^k(G)$ we have $\min_{s_{-i} \in \Gamma_{-i}^k(G)} p_i(s_i, s_{-i}) < p_i^{max}(H)$, and hence $maxmin_i(\Gamma^k(G)) < p_i^{max}(\Gamma^k(G))$.

By Corollary 10 the strategic game $\Gamma(G)$ has a Nash equilibrium. By the repeated application of Corollary 5 we have $maxmin_i(\Gamma(G)) = maxmin_i(\Gamma^k(G))$. Therefore

$$maxmin_i(\Gamma(G)) < p_i^{max}(\Gamma^k(G)). \tag{6}$$

Take now $s_{-i} \in lose_{-i}(\Gamma^k(G))$. We need to prove that $s_{-i} \notin \Gamma_{-i}^{k+2}(G)$.

For some $s_i \in \Gamma_i^k(G)$ we have $p_i(s_i, s_{-i}) = p_i^{max}(\Gamma^k(G))$. By Lemma 2 we can assume that $s_i \in \Gamma_i^{k+1}(G)$. Consider now the path $play(s_i, s_{-i})$. Then

- by (6) for the first node u lying on $play(s_i, s_{-i})$ (so the root),
  $maxmin_i(\Gamma(G^u)) < p_i^{max}(\Gamma^k(G))$,
- for the last node u lying on $play(s_i, s_{-i})$ (so the leaf),
  $p_i^{max}(\Gamma^k(G)) = p_i(s_i, s_{-i}) = o_i(u) = maxmin_i(\Gamma(G^u))$.

So for some adjacent nodes u, w lying on the path $play(s_i, s_{-i})$

$$maxmin_i(\Gamma(G^u)) < p_i^{max}(\Gamma^k(G)) \leq maxmin_i(\Gamma(G^w)). \tag{7}$$

Further, if for some adjacent nodes u', w' lying on the path $play(s_i, s_{-i})$ we have $turn(u') = i$ and $p_i^{max}(\Gamma^k(G)) \leq maxmin_i(\Gamma(G^{w'}))$, then $maxmin_i(\Gamma(G^{u'})) = maxmin_i(\Gamma(G^{w'}))$. So $turn(u) = -i$ and $s_{-i}(u) = w$.

If $s_{-i} \notin \Gamma_{-i}^{k+1}(G)$ then $s_{-i} \notin \Gamma_{-i}^{k+2}(G)$. So suppose $s_{-i} \in \Gamma_{-i}^{k+1}(G)$. We prove that then s_{-i} is weakly dominated in $\Gamma^{k+1}(G)$. The dominating strategy is obtained in two steps.

By Corollary 10 the game $\Gamma(G^u)$ has a Nash equilibrium s^*. First, we introduce the strategy $t_{-i} \in \Gamma_{-i}(G)$ defined as follows:

$$t_{-i}(x) := \begin{cases} s_{-i}(x) & \text{if } x \text{ not in } T^u, \\ s^*_{-i}(x) & \text{if } x \text{ in } T^u, \end{cases}$$

where $turn(x) = -i$ and T is the game tree of G.

We now establish two claims relating t_{-i} to s_{-i}.

Claim 1. $\forall s'_i \in \Gamma_i^{k+1}(G) : p_{-i}(s'_i, t_{-i}) \geq p_{-i}(s'_i, s_{-i})$.

Proof. Suppose by contradiction that there exists $s'_i \in \Gamma_i^{k+1}(G)$ such that $p_{-i}(s'_i, t_{-i}) < p_{-i}(s'_i, s_{-i})$. The strategy t_{-i} differs from s_{-i} only on the nodes in T^u, so the difference in the payoffs implies that u appears both in $play(s'_i, t_{-i})$ and $play(s'_i, s_{-i})$. This implies

$$maxmin_{-i}(\Gamma(G^u)) \leq p_{-i}((s'_i)^u, s^*_{-i}) = p_{-i}(s'_i, t_{-i}) < p_{-i}(s'_i, s_{-i}). \tag{8}$$

By Theorem 4 s^*_{-i} is a security strategy of player $-i$ in the game $\Gamma(G^u)$. Further, the node u appears in $play(s'_i, s_{-i})$, so $s'' := (s'_i, s_{-i})^u$ is a joint strategy in G^u. This and (8) imply

$$p_{-i}(s^*) = maxmin_{-i}(\Gamma(G^u)) < p_{-i}(s'_i, s_{-i}) = p_{-i}(s''),$$

so by (1), the fact that G^u is strictly competitive, and Theorem 4

$$p_i(s'_i, s_{-i}) = p_i(s'') < p_i(s^*) = maxmin_i(\Gamma(G^u)). \tag{9}$$

Next we introduce the strategy $t_i \in \Gamma_i(G)$ defined as follows (recall that $w = s_{-i}(u)$):

$$t_i(x) := \begin{cases} s'_i(x) & \text{if } x \text{ not in } T^w, \\ t^*_i(x) & \text{if } x \text{ in } T^w, \end{cases}$$

where $turn(x) = i$ and t^*_i is a security strategy of player i in the game $\Gamma(G^w)$.

We now establish two claims relating t_i to s'_i:

$$\forall s'_{-i} \in \Gamma^k_{-i}(G) : p_i(t_i, s'_{-i}) \geq p_i(s'_i, s'_{-i}), \tag{10}$$

$$p_i(t_i, s_{-i}) > p_i(s'_i, s_{-i}). \tag{11}$$

To establish (10) consider any strategy $s'_{-i} \in \Gamma^k_{-i}(G)$. By the definition of t_i, if w does not appear in $play(t_i, s'_{-i})$ then $p_i(t_i, s'_{-i}) = p_i(s'_i, s'_{-i})$. So suppose w appears in $play(t_i, s'_{-i})$. By the definition of t_i, (7), and the fact that both s'_i and s'_{-i} are strategies in $\Gamma^k(G)$

$$p_i(t_i, s'_{-i}) = p_i(t^*_i, (s'_{-i})^w) \geq maxmin_i(\Gamma(G^w)) \geq p^{max}_i(\Gamma^k(G)) \geq p_i(s'_i, s'_{-i}).$$

To establish (11) recall that we noted already that u appears in $play(s'_i, s_{-i})$. Since $turn(u) = -i$ and $s_{-i}(u) = w$, also w appears in $play(s'_i, s_{-i})$. The strategy t_i differs from s'_i only on the nodes in T^w, so w apears in $play(t_i, s_{-i})$, as well. Therefore by the definition of t_i, (7) and (9)

$$p_i(t_i, s_{-i}) = p_i(t^*_i, (s_{-i})^w) \geq maxmin_i(\Gamma(G^w)) > maxmin_i(\Gamma(G^u)) > p_i(s'_i, s_{-i}).$$

By Lemma 2 there exists $t'_i \in \Gamma^k_i(G)$ such that $p_i(t'_i, s'_{-i}) \geq p_i(t_i, s'_{-i})$ for all $s'_{-i} \in \Gamma^k_{-i}(G)$. This, together with (10) and (11) implies that s'_i is weakly dominated by t'_i in $\Gamma^k(G)$. Hence $s'_i \notin \Gamma^{k+1}_i(G)$, which yields a contradiction. □

Claim 2. $p_{-i}(s_i, t_{-i}) > p_{-i}(s_i, s_{-i})$.

Proof. The node u appears in $play(s_i, s_{-i})$, so by Theorem 4 and (7)

$$p_i(s^*) = maxmin_i(\Gamma(G^u)) < p^{max}_i(\Gamma^k(G)) = p_i(s_i, s_{-i}) = p_i(s^u_i, s^u_{-i}),$$

and consequently by Theorem 4(i) and the fact that G^u is strictly competitive

$$maxmin_{-i}(\Gamma(G^u)) = p_{-i}(s^*) > p_{-i}(s^u_i, s^u_{-i}) = p_{-i}(s_i, s_{-i}). \tag{12}$$

Further, the strategy t_{-i} differs from s_{-i} only on the nodes in T^u, so u appears in $play(s_i, t_{-i})$, as well. Hence

$$p_{-i}(s_i, t_{-i}) = p_{-i}(s^u_i, s^*_{-i}) \geq maxmin_{-i}(\Gamma(G^u)). \tag{13}$$

Combining (12) and (13) we get the claim. □

By Lemma 2 there exists $t'_{-i} \in \Gamma^{k+1}_{-i}(G)$ such that $p_{-i}(s'_i, t'_{-i}) \geq p_{-i}(s'_i, t_{-i})$ for all $s'_i \in \Gamma^{k+1}_i(G)$. We conclude by Claims 1 and 2 that s_{-i} is weakly dominated by t'_{-i} in $\Gamma^{k+1}(G)$. Therefore $s_{-i} \notin \Gamma^{k+2}_{-i}(G)$, as desired. □

We can now establish the announced result.

Theorem 19. *Let G be a finite strictly competitive extensive game with at most m outcomes. Then $\Gamma^{m-1}(G)$ is a trivial game.*

Proof. We prove a stronger claim, namely that for all $m \geq 1$ and $k \geq 0$ if $\Gamma^k(G)$ has at most m outcomes, then $\Gamma^{k+m-1}(G)$ is a trivial game.

We proceed by induction on m. We can assume that $m > 1$. For $m = 2$ the claim follows by Lemma 6. Take $m > 2$.

Case 1. For some $i \in \{1, 2\}$, $win_i(\Gamma^k(G)) \neq \emptyset$.

For player i every strategy $s_i \in win_i(\Gamma^k(G))$ weakly dominates all strategies $s_i' \notin win_i(\Gamma^k(G))$. So in $\Gamma^{k+1}(G)$ the set of strategies of player i equals $win_i(\Gamma^k(G))$ and hence $p_i^{max}(\Gamma^k(G))$ is his unique payoff in this game. By (1) $\Gamma^{k+1}(G)$, and hence also $\Gamma^{k+m-1}(G)$, is a trivial game.

Case 2. For all $i \in \{1, 2\}$, $win_i(\Gamma^k(G)) = \emptyset$.

Take joint strategies s and t such that $p_1(s) = p_1^{\max}(\Gamma^k(G))$ and $p_2(t) = p_2^{\max}(\Gamma^k(G))$. Since $m > 1$ (1) implies that the outcomes $(p_1(s), p_2(s))$ and $(p_1(t), p_2(t))$ are different.

We have $s_2 \in lose_2(\Gamma^k(G))$ and $t_1 \in lose_1(\Gamma^k(G))$. Hence by Lemma 18 for no joint strategy s' in $\Gamma^{k+2}(G)$ we have $p_1(s') = p_1^{\max}(\Gamma^k(G))$ or $p_2(s') = p_2^{\max}(\Gamma^k(G))$.

So $\Gamma^{k+2}(G)$ has at most $m-2$ outcomes. By the induction hypothesis $\Gamma^{k+m-1}(G)$ is a trivial game. □

9 Weak acyclicity

By Theorem 10 every finite extensive game has a Nash equilibrium. A natural question is whether we can strengthen this result to show that finite extensive games have the FIP. The following example adapted from [15] shows that even for simplest extensive games the answer is negative.

Example 13. Consider the extensive form game given in Figure 1. Following the convention introduced in Example 4, the strategies for player 1 are C and D, while the strategies of player 2 are CC, CD, DC and DD.

Then the following improvement sequence generates an infinite improvement path in this game:

$$\begin{array}{ccccccccc} (D, \underline{DC}) & \rightarrow & (\underline{D}, CD) & \rightarrow & (C, \underline{CD}) & \rightarrow & (\underline{C}, DC) & \rightarrow & (D, \underline{DC}) \\ (3,0) & & (1,1) & & (2,2) & & (0,3) & & (3,0) \end{array}$$

For convenience of the reader in each joint strategy we underlined the strategy which is not a best response and listed the corresponding outcomes. □

However, a weaker result, due to [12], does hold. It implies that every finite extensive game has a Nash equilibrium, a result established earlier, in Corollary 10.

Theorem 20. *Every finite extensive game is weakly acyclic.*

Proof. We prove the claim by defining a weak potential. Take a finite extensive game $G := (T, turn, o_1, \ldots, o_n)$, with $T := (V, E, v_0)$ and let S be the set of joint strategies.

Consider first a function $R : S \times V \to \{0, 1\}$ defined as follows:

$$R(s, v) := \begin{cases} 1 \text{ if } s_i^v \text{ is a best response to } s_{-i}^v \text{ in the subgame } G^v \\ 0 \text{ otherwise,} \end{cases}$$

where $i = turn(v)$.

Let now $L := (v_1, \ldots, v_k)$ be a list of the nodes from V such that each node appears after all of its children in T. For example,

$$((2, 2), (0, 3), (3, 0), b, (1, 1), c, a)$$

is such a list of the nodes of the tree from Figure 1, where we identify each leaf with the corresponding outcome.

Finally define the function $P : S \to \{0, 1\}^k$ by putting

$$P(s) := (R(s, v_1), \ldots, R(s, v_k)),$$

where, recall, $L = (v_1, \ldots, v_k)$, and consider the strict lexicographic ordering $<_{lex}$ over $\{0, 1\}^k$. We show that P is a weak potential w.r.t. this ordering and appeal to the Weakly Acyclic Theorem 1.

So consider a joint strategy s in G that is not a Nash equilibrium. Take a player i such that s_i is not a best response to s_{-i}. Let t_i be a best response of player i to s_{-i} and let $t = (t_i, s_{-i})$. Define s'_i as the modification of t_i such that its values on the nodes not lying on $play(t)$ are the values provided by s_i. More formally, for all nodes v such that $turn(v) = i$ we put

$$s'_i(v) := \begin{cases} t_i(v) & \text{if } v \text{ lies on } play(t) \\ s_i(v) & \text{otherwise} \end{cases}$$

Let $s' = (s'_i, s_{-i})$. Then $play(s') = play(t)$, so $o(leaf(s')) = o(leaf(t))$, and hence s'_i is also a best response of player i to s_{-i}. Since s_i is not a best response to s_{-i}, for some v from $play(s')$ such that $turn(v) = i$ player's i strategy s_i^v is not a best response to s_{-i}^v in the subgame G^v. Take the last such node v from $play(s')$.

So $R(v, s) = 0$ while $R(v, s') = 1$, since $(s'_i)^v$ is a best response to s_{-i}^v in the subgame G^v. Further, by the choice of v and the definition of s'_i we have that $R(w, s') = R(w, s)$ for all nodes w that precede v on the list L. We conclude that $P(s) <_{lex} P(s')$. □

10 Win or lose and chess-like games

In this section we discuss two classes of zero-sum extensive games introduced in Section 2. We begin with the win or lose games. Given such a game G we say that a strategy s_i of player i is ***winning*** if

$$\forall s_{-i} \in S_{-i} : o_i(leaf(s_i, s_{-i})) = 1,$$

and denote the (possibly empty) set of such strategies by $win_i(G)$.

The Matching Pennies game shows that in strategic win or lose games winning strategies may not exist. For finite win or lose extensive games the situation changes.

Theorem 21. *Let G be a finite win or lose extensive game. For all players i we have $win_i(G) \neq \emptyset$ iff $win_{-i}(G) = \emptyset$.*

Proof. Call the players white and black and call a finite win or lose extensive game *white* if the white player has a winning strategy in it and analogously for *black*. We prove that every such game is white or black. Clearly, these alternatives are mutually exclusive.

We proceed by induction on the number of nodes in the game tree. The claim clearly holds when the game tree has just one node. Consider a game G with the game tree T with more than one node. By the induction hypothesis for every child u of the root of T the subgame G^u is white or black.

Without loss of generality assume that in G the white player moves first. We claim that the game G is black if for every child u of the root of T the subgame G^u is black and otherwise that it is white. Indeed, in the first case no matter what is the first move of the white player he loses the game if the black player pursues his winning strategy in the resulting subgame, and otherwise the white player wins the game if he starts by selecting the move that leads to a white subgame and subsequently pursues in this subgame his winning strategy.

Note that we did not assume that the players alternate their moves. □

Next we consider chess-like games. We say that a strategy s_i of player i in such a game G ***guarantees him at least a draw*** if

$$\forall s_{-i} \in S_{-i} : o_i(leaf(s_i, s_{-i})) \geq 0,$$

and denote the (possibly empty) set of such strategies by $draw_i(G)$. The set $win_i(G)$ is defined as above.

Theorem 22. *Let G be a finite chess-like extensive game. We have*

$$win_1(G) \neq \emptyset \text{ or } win_2(G) \neq \emptyset \text{ or } (draw_1(G) \neq \emptyset \text{ and } draw_2(G) \neq \emptyset).$$

We reproduce a proof given in [2].

Proof. We introduce the following abbreviations:

- W_1 for $win_1(G) \neq \emptyset$,
- D_2 for $draw_2(G) \neq \emptyset$,
- W_2 for $win_2(G) \neq \emptyset$,
- D_1 for $draw_1(G) \neq \emptyset$.

Let G_1 and G_2 be the modifications of G in which each outcome $(0, 0)$ is replaced for G_1 by $(-1, 1)$ and for G_2 by $(1, -1)$. Then $win_1(G_1) = win_1(G)$, $win_2(G_1) = draw_2(G)$, $win_1(G_2) = draw_1(G)$, and $win_2(G_2) = win_2(G)$.

Hence by Theorem 21 applied to the games G_1 and G_2 we have $W_1 \vee D_2$ and $W_2 \vee D_1$, so $(W_1 \wedge W_2) \vee (W_1 \wedge D_1) \vee (D_2 \wedge W_2) \vee (D_2 \wedge D_1)$, which implies $W_1 \vee W_2 \vee (D_2 \wedge D_1)$, since $\neg(W_1 \wedge W_2)$, $(W_1 \wedge D_1) \equiv W_1$, and $(D_2 \wedge W_2) \equiv W_2$. □

The above result is attributed to [24]. However, in [19] it was pointed out that the paper contains only the idea and the corresponding result is not formally stated, and that the first rigorous statement of the result and its proof seems to have been provided in [10]. This result is stated in [22, page 125] and proved using backward induction (apparently the first use of it in the literature on game theory). In [6] a proof is provided that does not rely on backward induction and argument also covers chess-like games in which infinite plays, interpreted as draw, are allowed. As noticed in [2] Theorems 21 and 22 also hold for infinite extensive games in which every play is finite.

11 Conclusions

The aim of this tutorial was to provide a self-contained introduction to finite extensive games with perfect information aimed at computer scientists. Our objective was to provide a systematic presentation of the most important results concerning this class of games that in our view could be of interest to computer scientists.

In [2] we argued that the next most natural class of extensive games is the one in which every play is finite (in the set theory terminology the game trees are then *well-founded*). In such a class of games one can in particular consider *behavioural strategies* in the extensive games considered here, according to which a move consists of a probability distribution over the finite set of children of a given node.

Many textbooks on game theory rather choose as the next class extensive games with imperfect information. In these games players do not need to know

the previous moves made by the other players. An example is the Battleship game in which the first move for each player consists of a secret placing of the fleet on the grid. An interested reader is referred to Part III of [16].

Acknowledgements

We would like to thank Ruben Brokkelkamp, Marcin Dziubiński, R. Ramanujam, and an anonymous referee for useful comments on the first version of this paper. The second author was partially supported by the grant MTR/2018/001244.

References

[1] K. R. Apt and E. Grädel, editors. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011.

[2] K.R. Apt and S. Simon. Well-founded extensive games with perfect information. In *Proceedings 18th Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2021*, volume 335, pages 7–21. Electronic Proceedings in Theoretical Computer Science (EPTCS), 2021.

[3] R. Aumann. Backward induction and common knowledge of rationality. *Games and Economic Behavior*, 8:6–19, 1991.

[4] P. Battigalli. On rationalizability in extensive games. *Journal of Economic Theory*, 74:40–61, 1997.

[5] P.K. Dutta. *Strategies and Games*. MIT Press, Cambridge, MA, 2001.

[6] C. Ewerhart. Backward induction and the game-theoretic analysis of chess. *Games and Economic Behaviour*, 39:206–214, 2002.

[7] C. Ewerhart. Iterated weak dominance in strictly competitive games of perfect information. *Journal of Economic Theory*, 107(2):474–482, 2002.

[8] J. Halpern. Substantive rationality and backward induction. *Games and Economic Behavior*, 37(2):425–435, 2001.

[9] G.A. Jehle and P.J. Reny. *Advanced Microeconomic Theory*. Addison Wesley, New York, NY, second edition, 2001.

[10] L. Kalmár. Zur theorie der abstrakten spiele. *Acta Universitatis Szegediensis/Sectio Scientiarum Mathematicarum*, 4:65–85, 1928/29.

[11] H. W. Kuhn. Extensive games. *Proc. of the National Academy of Sciences*, 36:570–576, 1950.

[12] N.S. Kukushkin. Perfect information and potential games. *Games and Economic Behavior*, 38(2):306–317, 2002.

[13] L.M. Marx and J.M. Swinkels. Order independence for iterated weak dominance. *Games and Economic Behaviour*, 18:219–245, 1997.

[14] I. Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behaviour*, 13:111–124, 1996.

[15] I. Milchtaich. Schedulers, potentials and weak potentials in weakly acyclic games. Working paper 2013-03, Bar-Ilan University, Department of Economics, 2013.

[16] M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[17] R. Rosenthal. Games of perfect information, predatory pricing and the chain-store paradox. *Journal of Economic Theory*, 25(1):92–100, 1981.

[18] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, (2):65–67, 1973.

[19] U. Schwalbe and P. Walker. Zermelo and the early history of game theory. *Games and Economic Behavior*, 34(1):123–137, 2001.

[20] R. Selten. Spieltheoretische behandlung eines oligopolmodells mit nachfrageträgheit. *Zeitschrift für die gesamte Staatswisenschaft*, 121:301–324 and 667–689, 1965.

[21] R. Stalnaker. Knowledge, belief and counterfactual reasoning in games. *Economics and Philosophy*, 12(2):133–163, 1996.

[22] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior (60th Anniversary Commemorative Edition)*. Princeton Classic Editions. Princeton University Press, 2004.

[23] H. Peyton Young. The evolution of conventions. *Econometrica*, 61(1):57–84, 1993.

[24] E. Zermelo. Über eine anwendung der mengenlehre auf die theorie des schachspiels. In *Proc. of The Fifth International Congress of Mathematicians*, pages 501–504. Cambridge University Press, 1913.

The Memorial Column

BY

Stefan Schmid

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

A Visitor from Buenos Aires in the United States

Gregory Chaitin
University of Buenos Aires, Argentina
https://uba.academia.edu/GregoryChaitin

Abstract

Reminiscences of a crucial year in Chaitin's life, 1974.

In the first few months of 1974 I traveled from Buenos Aires to New York as a "summer visitor" at the IBM T. J. Watson Research Center in Yorktown Heights. I lived in the White Plains YMCA and commuted to the Watson Center by train and taxi.

It was during this visit that I discovered or invented the halting probability Ω. I remember the exact moment. I had been invited to give a lecture at a university somewhere in the United States—every week it was a different one—and was flying back to New York. At the precise moment that I realized that the halting probability was irreducible or algorithmically random, I was looking out the window and saw an unmistakable sight, the Pentagon in Washington, DC.

The author in 1974 in Terry Fine's office at Cornell University

Due to the usual delays for refereeing and such, the halting probability did not appear in print until the next year, 1975, in my fifth *Journal of the ACM* paper, "A Theory of Program Size Formally Identical to Information Theory."

By the way, the halting probability was originally ω, but the set theorist Robert Solovay, who was visiting the Watson Research Center, suggested to me that Ω might be better because in set theory ω stood for the set of natural numbers $\{0, 1, 2, 3, \ldots\}$.

During this visit to the Watson Research Center I also corrected the proofs of one of my first publications on incompleteness, destined to appear later in the year, an invited paper "Information-Theoretic Computational Complexity" in the *IEEE Transactions on Information Theory*, with an appendix giving the mathematical details, which proofs I was to send to Gödel, as I will tell below.[1]

And I had two very interesting experiences.

The first was that I attended a lecture at the New York Academy of Sciences in Manhattan by a mathematician I admired, Mark Kac. The lecture was on randomness, and Kac's thesis was that randomness was an interesting but slippery notion that resisted precise definition. He concluded his lecture with the following words: "In spite of this, a definition of randomness has been proposed by Kolmogorov and by a young fellow in Argentina, Gregory Chaitin." I stood up and said, "No, I'm here now!" Pandemonium, over which Kac declared, "This was not rehearsed!"

After the talk a gentleman came up to me and said, "I'm Dennis Flanagan, the Editor of *Scientific American*." And he told me the following story about Gödel. At the time Flanagan was living in Princeton, New Jersey, and he had just published a wonderful article, "Gödel's Proof" by Ernest Nagel and James R. Newman (1956), later expanded into a small book that completely obsessed me from the moment it appeared in the New York City public library (at that time I lived in Manhattan). Gödel was not known to the general intellectual public yet—that article and that book were to change that—and few people had seen a photo of Gödel and knew how he looked. However, Flanagan had sent the well-known portrait photographer Arnold Newman to Princeton in order to be able to include an image of Gödel in the article about him in *Scientific American*, resulting in a stark portrait of an angry-to-be-disturbed Gödel sitting in front of a bare blackboard that has been reproduced many times.

So Flanagan knew how Gödel looked. And one hot, humid summer day Flanagan was walking down the street in Princeton, a small town, and saw Gödel approaching. He prepared to introduce himself as the publisher of the article about

[1] However, my best paper on incompleteness was probably "Gödel's Theorem and Information" published in the *International Journal of Theoretical Physics* years later, in 1982, and then reprinted in Tymoczko, *New Directions in the Philosophy of Mathematics*, together with the paper that I sent to Gödel.

Gödel's proof. At that moment, however, a scantily clad beautiful young female student (we used to call them "co-eds" from the word "co-education") passed by, and Gödel stopped dead in his tracks to admire her. As they say in French, "La belle opportunité est perdu!" Flanagan did not dare to interrupt Gödel!

The second amazing experience was that I somehow managed to make a phone call to Gödel's office at the Princeton Institute for Advanced Study (IAS), a cold call as they say in the world of sales, and Gödel himself picked up the phone. "Professor Gödel," I said, "I am extremely fascinated [obsessed would have been more accurate] by your incompleteness theorem, and I have a new proof based on the Berry paradox instead of the Epimenides paradox [the paradox of the liar, 'This statement is false']." He replied, "It doesn't matter which paradox you use!" In fact, he says this in the introduction to his famous 1931 paper, which I was familiar with. So I was prepared, and I immediately answered, "Yes of course, but this suggests to me a new information-theoretic view of incompleteness, which I would very much like to visit you and tell you about." He replied, "Send me a paper of yours on this subject, and I will take a look at it and decide if I give you an appointment." So I sent him the proofs of my as-yet-unpublished 1974 IEEE paper. Then I called him back, and he commented "Very interesting, your complexity measure is an absolute notion [like computability as contrasted with provability, which depends on the axioms]." And he gave me an appointment!

The great day arrived, and I had already figured out how to take the train from Yorktown Heights into New York City and from there to Princeton, New Jersey, and how long that would take. It was the week before Easter, and that weekend I was supposed to leave NY and fly back to Buenos Aires. There had been a Spring snowstorm, nothing serious, nothing that would stop me from visiting my hero, Kurt Gödel. I was about to leave my office at IBM for the train station, when the phone rang, and a voice, a terrible voice, that of Gödel's secretary, announced that Gödel was very careful about his health and because it had snowed he was not coming into his office that day and therefore my appointment was canceled!

So this is how I spoke to Gödel on the phone twice but never met him. In retrospect, I think this is a much more interesting story than if I had actually met Gödel. It illustrates the surreal quality of interactions with Gödel.

The next week I stopped on my way back to Buenos Aires to present "A Theory of Program Size Formally Identical to Information Theory" at Stanford University.

However, the *annus mirabilis* 1974 was not yet over. Back in Buenos Aires, I was summoned by the head of IBM Argentina, Mr Benito Esmerode. The moment I sat down in Mr Esmerode's office, the phone rang. It was the head of IBM, Thomas J. Watson Jr. "Yes," said Mr Esmerode, "he is here in my office now, and yes, of course we will pay for his trip to the University of Notre Dame!"

What had happened? The IEEE was holding their 1974 International Sym-

posium on Information Theory later that year at Notre Dame University, and the organizers wanted me to present "A Theory of Program Size Formally Identical to Information Theory" in their opening plenary session. But I had told them I couldn't travel to Indiana. So the president of Notre Dame wrote to Thomas J. Watson Jr. and asked for his help. Problem solved.

That was my second trip from Buenos Aires to the United States in 1974. I was transferred from IBM Argentina to the Watson Research Center in 1975, the year that my article on "Randomness and Mathematical Proof" appeared in *Scientific American*[2].

Years later my friend Cristian Calude from the University of Auckland was visiting me at the Watson Research Center, and we decided to make a pilgrimage to Princeton. We found Einstein's former home near the IAS, Gödel's former home in a much poorer part of town, and Gödel's and John von Neumann's graves in the Princeton Cemetery. Einstein is not there. He was cremated and his ashes were scattered at an undisclosed location, as he had wished.

Cris Calude and Greg Chaitin at Gödel's grave in Princeton, New Jersey

Furthermore, as we stood looking at Gödel's home, the couple who were renting it from the current owner came out and invited us in. It turns out that much remained exactly as it had been when Kurt and his wife Adele lived there, in particular the heavy sound-proofing so that Gödel could work undisturbed in his study, and a shrine to the Virgin Mary in the garden, but not Adele's infamous pink flamingo, which Gödel found "charming."

[2]To be followed by "Randomness in Arithmetic" in 1988 and by "The Limits of Reason" in 2006.

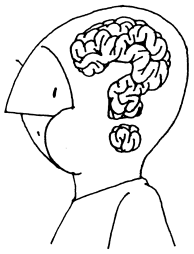
Gregory Chaitin is an Argentine-American mathematician living in Rio de Janeiro, and a lifetime honorary professor of the University of Buenos Aires with an honorary doctorate in philosophy from the University of Córdoba, the oldest university in Argentina and one of the oldest in South America. He was formerly at the IBM Watson Research Center in New York, where he was part of a small team that developed the Power processor architecture and its associated software.

On the theoretical side, Chaitin is best known for his discovery of the remarkable Ω number, a concrete example of irreducible complexity in pure mathematics, and which shows that mathematics is infinitely complex. For this he was awarded the Leibniz Medallion by Wolfram Research in 2007. He has also proposed modeling evolution as a random walk in software space ("metabiology").

Among his books are: *Algorithmic Information Theory*; *Conversations with a Mathematician*; *Meta Math!*; and *Proving Darwin*.

Festschriften: Cristian S. Calude, *Randomness and Complexity, from Leibniz to Chaitin*, World Scientific, Singapore, 2007; Gregory Chaitin, *Thinking about Gödel and Turing: Essays on Complexity, 1970–2007*, World Scientific, Singapore, 2007; Shyam Wuppuluri, Francisco Antonio Doria, *Unravelling Complexity: The Life and Work of Gregory Chaitin*, World Scientific, Singapore, 2020.

News and Conference Reports



ICALP 2022

49th EATCS International Colloquium on Automata, Languages and Programming
4-8 July 2022, in Paris, France, and online
https://icalp2022.irif.fr/

Call for Papers

The 49th International Colloquium on Automata, Languages, and Programming (ICALP) will take place

** in Paris, France, and online on 4-8 July 2022. **

The 2022 edition has the following special features:

- Submissions are anonymous, and there is a rebuttal phase.

- The conference is hybrid.

- This will be the 50th birthday of the conference and some special events are planned.

ICALP is the main conference and annual meeting of the European Association for Theoretical Computer Science (EATCS). As usual, ICALP will be preceded by a series of workshops, which will take place on July 4. The 2022 edition will be the occasion to celebrate the 50th anniversary of both EATCS and the first ICALP, which was first held in 1972 in Rocquencourt, in the Paris area.

Important dates

Submissions: February 9, 2022 AoE
Rebuttal: March 21-23
Notification: April 11
Camera-ready version: April 25
Early registration: TBA
Conference: 4-8 July, 2022

Deadlines are firm; late submissions will not be considered.

Conference website: https://icalp2022.irif.fr/

Submission: https://easychair.org/my/conference?conf=icalp2022#

Invited Speakers

Albert Atserias, Universitat Politècnica de Catalunya
Constantinos Daskalakis, MIT
Leslie Ann Goldberg, Oxford University
Madhu Sudan, Harvard
Stéphan Thomassé, ENS Lyon
Santosh Vempala, Georgia Tech

Submission Guidelines

1) Papers must present original research on the theory of computer science. No prior publication and no simultaneous submission to other publication outlets (either a conference or a journal) is allowed. Authors are encouraged to also make full versions of their submissions freely accessible in an on-line repository such as ArXiv, HAL, ECCC.

2) Submissions take the form of an extended abstract of no more than 15 pages, excluding references and a clearly labelled appendix. The appendix may consist either of omitted proofs or of a full version of the submission, and it will be read at the discretion of program committee members. The extended abstract has to present the merits of the paper and its main contributions clearly, and describe the key concepts and technical ideas used to obtain the results. Submissions must provide the proofs which can enable the main mathematical claims of the paper to be fully verified.

3) Submissions are anonymous. The conference will employ a fairly lightweight double-blind reviewing process. Submissions should not reveal the identity of the authors in any way. In particular, authors' names, affiliations, and email addresses should not appear at the beginning or in the body of the submission. Authors should not include obvious references that reveal their own identity, and should ensure that any references to their own related work are in the third person (e.g., not "We build on our previous work . . . " but rather "We build on the work of . . . ").

The purpose of this double-blind process is to help PC members and external reviewers come to an initial judgment about the paper without bias, and not to make it impossible for them to discover who the authors are if they were to try. Nothing should be done in the name of anonymity that weakens the submission or makes the job of reviewing the paper more difficult. In particular, important references should not be omitted or anonymized. In addition, authors should feel free to disseminate their ideas or draft versions of their paper as they normally would. For example, authors may post drafts of their papers on the web, submit them to arXiv, and give talks on their research ideas.

4) The submissions are done via Easychair to the appropriate track of the conference (see topics below). The use of pdflatex and the LIPIcs style are mandatory:

papers that deviate significantly from the required format risk rejection without consideration of merit.

5) During the rebuttal phase, authors will have three days, March 21-23, to view and respond to initial reviews. Further instructions will be sent to authors of submitted papers before that time.

6) One author per accepted paper is expected to present the work in Paris, unless there are strong reasons not to do so, including high environmental cost of travel or impossibility to travel. We will be monitoring the current situation and are aware of possible travel restrictions, but we aim to organize the conference as a hybrid event with a strong in-person attendance. If no speaker can attend, a remote presentation and participation to the discussion session are mandatory.

7) Papers authored only by students should be marked as such upon submission in order to be eligible for the best student paper awards of the track.

Awards

During the conference, the following awards will be given:
- the EATCS award (https://eatcs.org/index.php/eatcs-award),
- the Gödel prize (https://eatcs.org/index.php/goedel-prize),
- the Presburger award (https://eatcs.org/index.php/presburger),
- the EATCS distinguished dissertation award (https://eatcs.org/index.php/dissertation-award),
- the best papers for Track A and track B,
- the best student papers for Track A and track B (see submission guidelines).

Proceedings

ICALP proceedings are published in the Leibniz International Proceedings in Informatics (LIPIcs) series. This is a series of high-quality conference proceedings across all fields in informatics established in cooperation with Schloss Dagstuhl - Leibniz Center for Informatics. LIPIcs volumes are published according to the principle of Open Access, i.e., they are available online and free of charge.

Topics

Papers presenting original research on all aspects of theoretical computer science are sought. Typical but not exclusive topics of interest are:

Track A: Algorithms, Complexity and Games

- Algorithmic and Complexity Aspects of Network Economics

- Algorithmic Aspects of Biological and Physical Systems
- Algorithmic Aspects of Networks and Networking
- Algorithmic Aspects of Security and Privacy
- Algorithmic Game Theory and Mechanism Design
- Approximation and Online Algorithms
- Combinatorial Optimization
- Combinatorics in Computer Science
- Computational Complexity
- Computational Geometry
- Computational Learning Theory
- Cryptography
- Data Structures
- Design and Analysis of Algorithms
- Distributed and Mobile Computing
- Foundations of Machine Learning
- Graph Mining and Network Analysis
- Parallel and External Memory Computing
- Parameterized Complexity
- Quantum Computing
- Randomness in Computation
- Sublinear Time and Streaming Algorithms
- Theoretical Foundations of Algorithmic Fairness

Track B: Automata, Logic, Semantics, and Theory of Programming

- Algebraic and Categorical Models of Computation

- Automata, Logic, and Games
- Database Theory, Constraint Satisfaction Problems, and Finite Model Theory
- Formal and Logical Aspects of Learning
- Formal and Logical Aspects of Security and Privacy
- Logic in Computer Science and Theorem Proving
- Models of Computation: Complexity and Computability
- Models of Concurrent, Distributed, and Mobile Systems
- Models of Reactive, Hybrid, and Stochastic Systems
- Principles and Semantics of Programming Languages
- Program Analysis, Verification, and Synthesis
- Type Systems and Typed Calculi

ICALP 2022 Programme Committee

Track A: Algorithms, complexity, and games
Petra Berenbrink - University of Hamburg
Sergio Cabello - University of Ljubljana
Yixin Cao - Hong Kong Polytechnic University
Sitan Chen - University of California Berkeley
Xi Chen - Columbia University
Ilias Diakonikolas - University of Wisconsin-Madison
David Doty - University of California Davis
Yuval Filmus - Technion
Cyril Gavoille - Université de Bordeaux
Sevag Gharibian - Paderborn University
Seth Gilbert - National University of Singapore
Nick Gravin - Shanghai University of Finance and Economics
Kasper Green Larsen - Aarhus University
Abhradeep Guha Thakurta - Google Research
Hamed Hatami - McGill University
Sandy Irani - University of California Irvine
Yuval Ishai - Technion

Aayush Jain - NTT Research/CMU
Ken-ichi Kawarabayashi - National Institute of Informatics
Yuqing Kong - Peking University
Michal Koucky - Charles University
Stefano Leonardi - Sapienza Universita di Roma
Nutan Limaye - IT University of Copenhagen
Frederic Magniez - CNRS
Audra Mcmillan - Apple
Slobodan Mitrovic - MIT / University of California Davis
Wolfgang Mulzer - Freie Universitat Berlin
Cameron Musco - University of Massachusetts Amherst
Anand Natarajan - MIT
Jelani Nelson - University of California Berkeley
Evdokia Nikolova - University of Texas at Austin
Debmalya Panigrahi - Duke University
Richard Peng - Georgia Tech
Vijaya Ramachandran - University of Texas at Austin
Saket Saurabh - Institute of Mathematical Sciences, Chennai
Christian Sohler - University of Cologne
Thomas Steinke - Google Research
Vasilis Syrgkanis - Microsoft Research
Emanuele Viola - Northeastern University
Adrian Vladu - CNRS
Jan Vondrak - Stanford
Hoeteck Wee - NTT Research / ENS
David Woodruff - CMU (chair)
Christian Wulf-Nilsen - University of Copenhagen

Track B: Automata, Logic, Semantics, and Theory of Programming
Luca Aceto - Reykjavik University
Isolde Adler - University of Leeds
Antoine Amarilli - Télécom Paris
Pablo Barcelo - Catholic University of Chile
Libor Barto - Charles University
Mikołaj Bojańczyk - University of Warsaw (chair)
Laura Ciobanu - Heriot-Watt University
Erich Grädel - RWTH Aachen University
Christoph Haase - University of Oxfordv
Marcin Jurdziński - University of Warwick
Benjamin Kaminski - University College London
Joost-Pieter Katoen - RWTH Aachen University

Bartek Klin - University of Oxford
Naoki Kobayashi - University of Tokyo
Dexter Kozen - Cornell University
Orna Kupferman - Hebrew University
Jérôme Leroux - CNRS / University of Bordeaux
Nathan Lhote - Aix-Marseille University
Markus Lohrey - University of Siegen
Joël Ouaknine - Max Planck Institute
Prakash Panangaden - McGill University
Michael Pinsker - Vienna University of Technology
Sven Schewe - University of Liverpool
Jeffrey Shallit - University of Waterloo
Mahsa Shirmohammadi - CNRS / University of Paris
Sebastian Siebertz - University of Bremen
Alex Simpson - University of Ljubljana
Lidia Tendera - University of Opole

ICALP 2022 Workshop Chairs
Track A: Valia Mitsou
Track B: Mahsa Shirmohammadi
ICALP 2022 Proceedings Chairs
Emanuela Merelli
ICALP 2022 Organizing Committee
Sandrine Cadet
Olivier Carton
Thomas Colcombet
Geoffroy Couteau
Hugo Férée
Irène Guessarian
Natalia Hacquart
Florian Horn
Simon Mauras
Valia Mitsou
Sylvain Perifel
Amaury Pouly
Arnaud Sangnier
Sylvain Schmitz
Mahsa Shirmohammadi

TheoretiCS

TheoretiCS is a new Diamond Open Access electronic journal covering all areas of Theoretical Computer Science (TCS), that will be launched during Fall 2021. The scope of the journal is TCS broadly construed, including, but not restricted to, the Theory of Computing and the Theory of Programming. One of our aims is to bridge the, often artificial, division between TCS-A and TCS-B.

Access to all papers is free. Authors are not required to pay any publication fees or article processing charges, and retain copyright under a Creative Commons license.

TheoretiCS strives for top quality scholarship in Theoretical Computer Science. It publishes original research, explicitly including suitably revised and extended versions of conference papers. To be accepted, a paper must make a significant contribution of lasting value to a relevant area of TCS, and its presentation must be of high quality.

TheoretiCS is published by the TheoretiCS Foundation e.V., a non-profit organisation registered in Germany. The journal is a joint effort of the TCS community. Its policies are decided by the Advisory Board, mostly consisting of representatives of major TCS conferences.

TheoretiCS is an overlay journal of the Computing Research Repository (CoRR) on arXiv. It is hosted by the Episciences platform for overlay journals, which is kindly provided by the CCSD.

Editorial Board. The inaugural Editors-in-Chief are Javier Esparza (TU Munich) and Uri Zwick (Tel Aviv U.). The Editorial Board can be found at `https://theoretics.episciences.org/`, see also below.

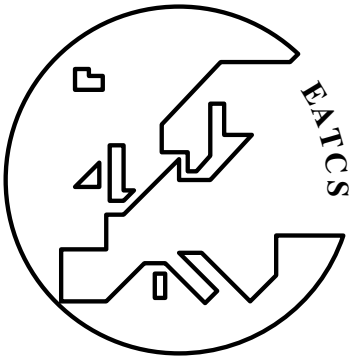
Organization and some history. The project started in 2019 and underwent a long gestation. Our aim was to rapidly become a reference journal and to contribute to the unity of the Theoretical Computer Science global community. From the start, we wanted to have a thorough discussion with a wide representation of

the community, on how to best implement the guiding principles of TheoretiCS. It was deemed essential to make sure that all fields of theoretical computer science would feel at home in this journal, and that it would be recognized as a valid venue for publication all over the world.

This resulted in the creation of an Advisory Board, composed of representatives of most of the main conferences in the field (currently APPROX, CCC, COLT, CONCUR, CSL, FOCS, FoSSaCS, FSCD, FSTTCS, ICALP, ICDT, ITCS, LICS, MFCS, PODC, SoCG, SODA, STACS, STOC, TCC) and of so-called members-at-large.

TheoretiCS acknowledges endorsement by Noga Alon, Shafi Goldwasser, Donald E. Knuth, Robert E. Tarjan, Leslie Valiant, Moshe Y. Vardi, and Andrew C.-C. Yao.

European Association for Theoretical Computer Science



E A T C S

EATCS

HISTORY AND ORGANIZATION

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

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

MAJOR ACTIVITIES OF EATCS

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

Other activities of EATCS include the sponsorship or the cooperation in the organization of various more specialized meetings in theoretical computer science. Among such meetings are: CIAC (Conference of Algorithms and Complexity), CiE (Conference of Computer Science Models of Computation in Context), DISC (International Symposium on Distributed Computing), DLT (International Conference on Developments in Language Theory), ESA (European Symposium on Algorithms), ETAPS (The European Joint Conferences on Theory and Practice of Software), LICS (Logic in Computer Science), MFCS (Mathematical Foundations of Computer Science), WADS (Algorithms and Data Structures Symposium), WoLLIC (Workshop on Logic, Language, Information and Computation), WORDS (International Conference on Words).

Benefits offered by EATCS include:

- Subscription to the "Bulletin of the EATCS;"
- Access to the Springer Reading Room;
- Reduced registration fees at various conferences;
- Reciprocity agreements with other organizations;
- 25% discount when purchasing ICALP proceedings;
- 25% discount in purchasing books from "EATCS Monographs" and "EATCS Texts;"
- Discount (about 70%) per individual annual subscription to "Theoretical Computer Science;"
- Discount (about 70%) per individual annual subscription to "Fundamenta Informaticae."

Benefits offered by EATCS to Young Researchers also include:

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

(1) THE ICALP CONFERENCE

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

SITES OF ICALP MEETINGS:

- Paris, France 1972
- Saarbrücken, Germany 1974
- Edinburgh, UK 1976
- Turku, Finland 1977
- Udine, Italy 1978
- Graz, Austria 1979
- Noordwijkerhout, The Netherlands 1980
- Haifa, Israel 1981
- Aarhus, Denmark 1982
- Barcelona, Spain 1983
- Antwerp, Belgium 1984
- Nafplion, Greece 1985
- Rennes, France 1986
- Karlsruhe, Germany 1987
- Tampere, Finland 1988
- Stresa, Italy 1989
- Warwick, UK 1990
- Madrid, Spain 1991
- Wien, Austria 1992
- Lund, Sweden 1993
- Jerusalem, Israel 1994
- Szeged, Hungary 1995
- Paderborn, Germany 1996
- Bologne, Italy 1997
- Aalborg, Denmark 1998
- Prague, Czech Republic 1999
- Genève, Switzerland 2000
- Heraklion, Greece 2001
- Malaga, Spain 2002
- Eindhoven, The Netherlands 2003
- Turku, Finland 2004
- Lisabon, Portugal 2005
- Venezia, Italy 2006
- Wrocław, Poland 2007
- Reykjavik, Iceland 2008
- Rhodes, Greece 2009
- Bordeaux, France 2010
- Zürich, Switzerland 2011
- Warwick, UK 2012
- Riga, Latvia 2013
- Copenhagen, Denmark 2014
- Kyoto, Japan 2015
- Rome, Italy 2016
- Warsaw, Poland 2017
- Prague, Czech Republic 2018
- Patras, Greece 2019
- Saarbrücken, Germany (virtual conference) 2020

(2) THE BULLETIN OF THE EATCS

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

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

Contributions to any of the above areas are solicited, in electronic form only according to for-

mats, deadlines and submissions procedures illustrated at `http://www.eatcs.org/bulletin`. Questions and proposals can be addressed to the Editor by email at `bulletin@eatcs.org`.

(3) OTHER PUBLICATIONS

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

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

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

The editors of the EATCS Monographs and Texts are now M. Henzinger (Vienna), J. Hromkovič (Zürich), M. Nielsen (Aarhus), G. Rozenberg (Leiden), A. Salomaa (Turku). Potential authors should contact one of the editors.

EATCS members can purchase books from the series with 25% discount. Order should be sent to:

Prof.Dr. G. Rozenberg, LIACS, University of Leiden,
P.O. Box 9512, 2300 RA Leiden, The Netherlands

who acknowledges EATCS membership and forwards the order to Springer-Verlag.

The journal *Theoretical Computer Science*, founded in 1975 on the initiative of EATCS, is published by Elsevier Science Publishers. Its contents are mathematical and abstract in spirit, but it derives its motivation from practical and everyday computation. Its aim is to understand the nature of computation and, as a consequence of this understanding, provide more efficient methodologies.

The Editor-in-Chief of the journal currently are D. Sannella (Edinburgh), L. Kari and P.G. Spirakis (Patras).

ADDITIONAL EATCS INFORMATION

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

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

EATCS MEMBERSHIP

DUES

The dues are € 40 for a period of one year (two years for students / Young Researchers). Young Researchers, after paying, have to contact `secretary@eatcs.org`, in order to get additional years. A new membership starts upon registration of the payment. Memberships can always be prolonged for one or more years.

In order to encourage double registration, we are offering a discount for SIGACT members, who can join EATCS for € 35 per year. We also offer a five-euro discount on the EATCS membership fee to those who register both to the EATCS and to one of its chapters. Additional € 35 fee is required for ensuring the *air mail* delivery of the EATCS Bulletin outside Europe.

HOW TO JOIN EATCS

You are strongly encouraged to join (or prolong your membership) directly from the EATCS website `www.eatcs.org`, where you will find an online registration form and the possibility of secure online payment. Alternatively, contact the Secretary Office of EATCS:

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

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

For adittional information please contact the Secretary of EATCS:

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