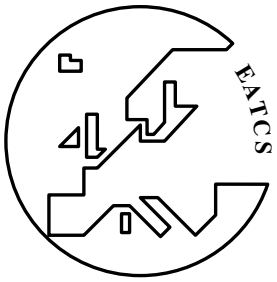
ISSN 0252–9742

Bulletin

of the

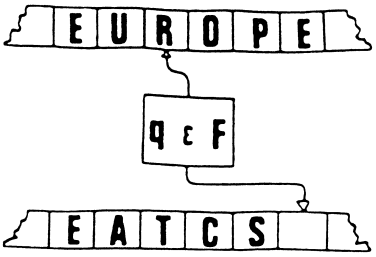
European Association for Theoretical Computer Science

EATCS



Number 111 October 2013

Council of the European Association for Theoretical Computer Science



President:	Luca Aceto	Iceland
Vice Presidents:	Paul Spirakis	Greece
	Burkhard Monien	Germany
	Antonin Kucera	Czech Republic
Treasurer:	Dirk Janssens	Belgium
Bulletin Editor:	Kazuo Iwama	Kyoto, Japan

Susanne Albers	Germany	Juhani Karhumaki	Finland
Lars Arge	Denmark	Elvira Mayordomo	Spain
Jos Baeten	The Netherlands	Luke Ong	UK
Paul Beame	USA	Catuscia Palamidessi	France
Josep Díaz	Spain	David Peleg	Israel
Zoltán Ésik	Hungary	Giuseppe Persiano	Italy
Fedor Fomin	Norway	Jean-Eric Pin	France
Leslie Ann Goldberg	UK	Vladimiro Sassone	UK
Monika Henzinger	Austria	Roger Wattenhofer	Switzerland
Giuseppe F. Italiano	Italy	Thomas Wilke	Germany
Christos Kaklamanis	Greece	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)		

Secretary Office:	Ioannis Chatzigiannakis	Greece
	Efi Chita	Greece

EATCS Council Members

email addresses

Luca Aceto . . . . . luca@ru.is
Susanne Albers . . . . . albers@informatik.hu-berlin.de
Lars Arge . . . . . large@madalgo.au.dk
Jos Baeten . . . . . jos.baeten@cwi.nl
Paul Beame . . . . . beame@cs.washington.edu
Josep Díaz . . . . . diaz@lsi.upc.es
Zoltán Ésik . . . . . ze@inf.u-szeged.hu
Fedor Fomin . . . . . fomin@ii.uib.no
Leslie Ann Goldberg . . . . . leslie.goldberg@cs.ox.ac.uk
Monika Henzinger . . . . . monika.henzinger@univie.ac.at
Giuseppe F. Italiano . . . . . italiano@disp.uniroma2.it
Dirk Janssens . . . . . Dirk.Janssens@ua.ac.be
Christos Kaklamanis . . . . . kakl@ceid.upatras.gr
Juhani Karhumäki . . . . . karhumak@cs.utu.fi
Antonin Kucera . . . . . tony@fi.muni.cz
Elvira Mayordomo . . . . . elvira@unizar.es
Burkhard Monien . . . . . bm@uni-paderborn.de
Luke Ong . . . . . luke.Ong@cs.ox.a.uk
Catuscia Palamidessi . . . . . catuscia@lix.polytechnique.fr
David Peleg . . . . . peleg@wisdom.weizmann.ac.il
Giuseppe Persiano . . . . . giuper@dia.unisa.it
Jean-Eric Pin . . . . . jean-Eric.Pin@liafa.univ-paris-diderot.fr
Vladimiro Sassone . . . . . vs@ecs.soton.ac.uk
Paul Spirakis . . . . . spirakis@cti.gr
Roger Wattenhofer . . . . . wattenhofer@tik.ee.ethz.ch
Thomas Wilke . . . . . wilke@ti.informatik.uni-kiel.de
Gerhard Wöeginger . . . . . g.j.woeginger@math.utwente.nl

Bulletin Editor: Kazuo Iwama, Kyoto, Japan
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.

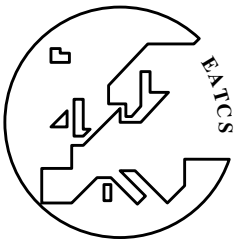
The details can be found at `http://www.eatcs.org/bulletin`, including class files, their documentation, and guidelines to deal with things such as pictures and overfull boxes. When in doubt, email `bulletin@eatcs.org`.

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 colleagues,

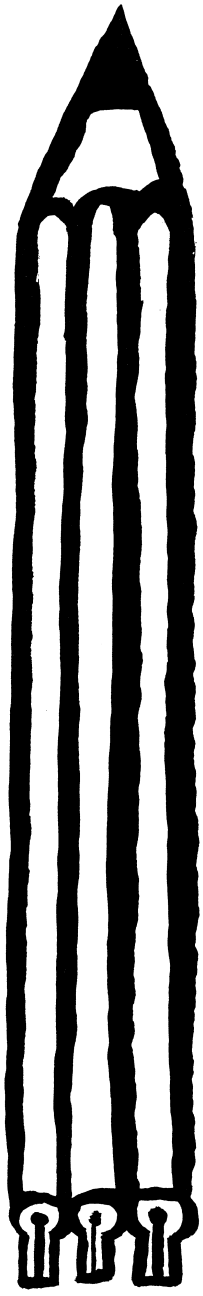
First of all, I hope that you had a good summer break and that you have recharged your batteries for whatever challenges await you in the new academic year.

I enjoyed meeting several of you at ICALP 2013 in Riga. It was a pleasure to see many young researchers and students at the conference, and I really appreciated the good attendance we had at the EATCS general assembly after the conference excursion. Thanks to all of you who took part in the general assembly!

The 40th ICALP was an excellent conference, both scientifically and socially. The organizers did their very best to make it a memorable event, and I like to think that all the participants felt welcome and enjoyed the conference. On the behalf of the EATCS, I warmly thank the local organizers and their assistants for a job well done.

ICALP 2014 will be held in Copenhagen, Denmark, immediately after SEA 2013 and SWAT 2013. It will be a four-day ICALP and the general chair for the conference is Thore Husfeldt. I hope that you will submit your best work to the conference and that you will make the trip to wonderful Copenhagen for what promises to be yet another exciting ICALP conference.

The EATCS council meeting and the general assembly at ICALP 2013 were interesting and fruitful. We made some steps forward, but there is still a lot that could be done, especially for young researchers, if suitable resources were available. I hope



that you will join the EATCS and encourage your colleagues and students to do so. The EATCS membership fee is low and, by becoming a member, you will contribute to the activities of our organization and will support the development of TCS.

The general assembly of the EATCS decided that ICALP 2015 will be held in Kyoto, Japan, and will be co-located with LICS 2015. Kazuo Iwama, the new editor in chief of the Bulletin, is the ICALP 2015 general chair. After 42 years, this will be the first ever ICALP outside Europe and I am very excited at the prospect of holding ICALP in Kyoto.

As I announced at the general assembly, the EATCS will start an EATCS Fellows programme. The goal of this programme is to honour members of the EATCS who have contributed outstanding scientific achievements in the field of Theoretical Computer Science. The first deadline for nominations is at the end of this year and we plan to make the first announcement of fellows at ICALP 2014 in Copenhagen. You can find the first call for nominations in this issue of the Bulletin and I hope that you will take the time to submit nominations. The first EATCS Fellows Selection Committee consists of Rocco De Nicola (IMT Lucca, Italy), Paul Goldberg (Oxford, UK), Anca Muscholl (LaBRI, France; chair), Dorothea Wagner (Karlsruhe, Germany) and Roger Wattenhofer (ETH Zurich, CH). I thank these colleagues for their willingness to serve on this committee.

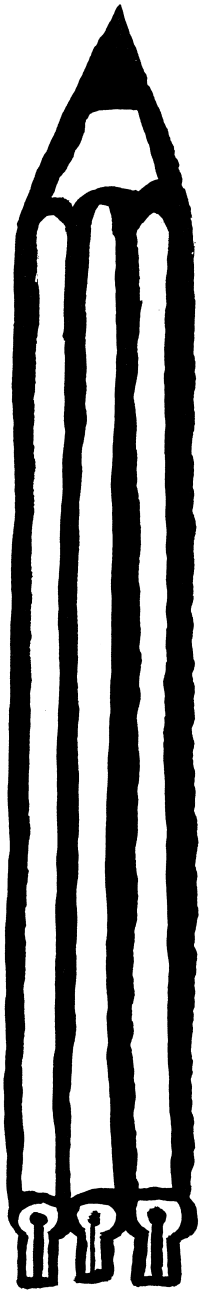
The EATCS will also begin a new EATCS Young Researcher School Series from 2014. The first school in the series will be organized by Tony Kucera in a beautiful

location in Moravia in late July/early August. The theme of the school will be Automata, Logic and Games. Tony has assembled a stellar PC for the event and I look forward to seeing the programme of lectures.

As usual at this time of the year, the EATCS issues calls for nominations for the EATCS Award, the Gödel Prize and the Presburger Award. You can read the calls in this issue of the Bulletin. I hope that you will take the time to nominate excellent researchers and papers for these awards, which will all be presented at ICALP 2014 in Copenhagen.

Last, but by no means least, let me remind you that you are always welcome to write to me at president@eatcs.org. I look forward to hearing your suggestions for improving the impact of the EATCS on the TCS community and for making EATCS membership more attractive for young researchers, who are the future of our field of study.

Luca Aceto, Reykjavik
October 2013



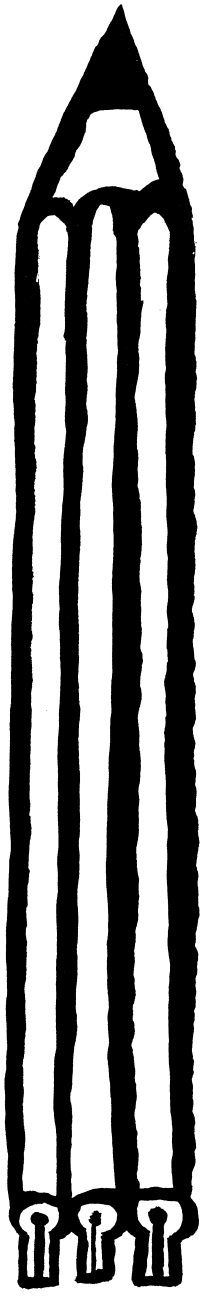
Dear Reader,

First of all, it is my great pleasure and honour to be able to serve as editor-in-chief of our Bulletin. It is just fantastic to see the first issue under my editorship coming out. I would like to express my sincere appreciation to Maria for her great job. Also thank you very much, Luca, it would be totally impossible to start this job without your tremendous help. And most importantly I need YOUR help to maintain this wonderful publication, many thanks in advance.

A couple of days back, I went to the library of the department with a small hope (sorry, our library is small) for back-numbers of the Bulletin. Fortunately (and a bit surprisingly) they are there, almost everything except the first several issues. I spent a couple of hours just browsing pages, especially photos in 80's. Grzegorz Rozenberg was editor-in-chief from 1981 to 2003 and it is he who established the current style. I am happy to keep this style basically unchanged.

As you can see, we have four columns in this issue: The Computational Complexity Column by Vikraman Arvind, The Distributed Computing Column by Panagiota Fatourou, The Logic in Computer Science Column by Yuri Gurevich, and The Formal Language Theory Column, by our new editor Giovanni Pighizzini. I hope we can add a couple more columns from as early as the next issue.

This issue has the Contributions by EATCS Award Recipients section. The first



article is by Erik Demaine who received this year's Presburger Award. Luca Moscardelli (University of Chieti-Pescara) received the Young Researcher in Theoretical Computer Science Award 2013, which was presented for the first time by the Italian Chapter of the EATCS at ICTCS 2013, the 14th Italian Conference on Theoretical Computer Science, 9-11 September 2013, Palermo (Italy). At the same event, Jacopo Mauro and Alessandra Scafuro received the Best Ph.D. Thesis in Theoretical Computer Science Award 2013 of the Italian Chapter of the EATCS, The contributions by Mauro, Moscardelli and Scafuro appearing in this issue of the Bulletin are based on the talks they delivered at ICTCS 2013 and highlight the work for which they received the aforementioned awards.

Finally I would like to remind you that we always welcome "news" of any kind, as you saw in the call-for-submission email a couple of weeks ago. Let me add a word about this call-for-submission. It is at the same time a "call-for-suggestions" of those who will be able to submit such articles. If you give me candidates, I would be happy to send him/her an individual strong request. By the way, Dominik Scheder quickly responded to our request; please enjoy his report on the startup of the Simon Institute.

ICALP 2015 will be in Kyoto. So please prepare for your trip to this wonderful city.

Kazuo Iwama, Kyoto
October 2013

Report on the EATCS General Assembly

ICALP 2013, Wednesday, 10 July 2013

The 2013 General Assembly (GA) of the EATCS took place on Wednesday, 10 July 2013, in Riga (Latvia), the venue of ICALP 2013. The President, Luca Aceto, opened the GA at 18:10. The agenda consisted of the following items:

- Reports on ICALP 2013
- Award of the best student papers
- Report on the organization of ICALP 2014 (Philip Bille)
- Proposal for ICALP 2015 for approval by the General Assembly (Kazuo Iwama)
- Orna Kupferman. The Gender Challenge at TCS.
- Report from the president
- Any other business

Before starting the GA, the President asked the attendees to honour the memory of two colleagues who have recently passed away, namely Philippe Darondeau and Kohei Honda. Kohei was one of the invited speakers for ICALP 2012 and Philippe was an old friend of the EATCS and of ICALP.

The slides used by the President at the GA are available at `http://www.ru.is/~luca/EATCS/GA2013.pdf`.

1 Reports on ICALP 2013

On behalf of the ICALP 2013 organizing committee, Agnis Škuskovniks presented a summary of the state of the organization. He reported that 276 people (not counting the locals) have registered to ICALP 2013, but 53 of them only for the three satellite workshops. Agnis also presented interesting statistics on the geographical distribution of the attendees, which culminated in a Eurovision-Song-Festival-style countdown that revealed that Germany provided the largest number of attendees for ICALP 2013. The President thanked the local organizers for making the 40th ICALP such a festive occasion.

The PC chairs for the three tracks of ICALP 2013 were Fedor V. Fomin (Track A), Marta Kwiatkowska (Track B) and David Peleg (Track C).

Marta Kwiatkowska presented the report from th PC chairs on behalf of her colleagues. The 124 contributed papers for ICALP 2013 were divided into the three tracks of ICALP 2013 as follows:

- 71 papers for "Track A: Algorithms, Complexity and Games", which were selected from 249 submissions;
- 33 papers for "Track B: Logic, Semantics, Automata and Theory of Programming", which were selected from 114 submissions; and
- 20 papers for "Track C: Foundations of Networked Computation", which were selected from 60 submissions.

To give you an idea of the amount of work that is involved in the selection of papers for ICALP, the PCs for the three tracks of ICALP 2013 had 71 members in total, 1299 reviews were produced during the PC work, of which 794 reviews were written by 684 external reviewers. It is thus fair to say that a large fraction of the TCS community was actively involved in the PC work for ICALP 2013.

The President takes this opportunity of thanking the PC chairs, their PCs and the sub-reviewers for doing an exceptional job.

2 Best Student Papers 2013

The prize for the Best Student Paper of track A of ICALP 2013 is awarded to Radu Curticapean for his paper *Counting matchings of size k is #W*[1]*-hard*.

The prize for the Best Student Paper of track B of ICALP 2013 goes to Nicolas Basset for his paper *A maximal entropy stochastic process for a timed automaton*.

There was no eligible accepted paper for Track C this year.

3 ICALP 2014

The annual main conference of the EATCS since 1972 is the ICALP conference. ICALP 2014 will be held from Tuesday, 8 July 2014, to Friday, 11 July 2014, on the premises of the IT University in Copenhagen, Denmark, after SEA 2014 (29 June–1 July 2014) and SWAT 2014 (2–4 July 2013). It will be a four-day ICALP and the general chair for the conference is Thore Husfeldt.

The PC chairs for ICALP 2014 are Elias Koutsoupias (Oxford, UK) for Track A, Javier Esparza (TU Münich, Germany) for Track B and Pierre Fraigniaud (Paris Diderot, FR) for Track C.

The invited speakers are Sanjeev Arora (Princeton University, USA), Maurice Herlihy (Brown University, USA), Viktor Kuncak (EPFL, CH) and Claire Mathieu (ENS Paris, France).

During the GA, Philip Bille reported on the organization of ICALP 2014. The preliminary call for papers for the conference was made available during the GA.

4 Proposal for ICALP 2015

The general assembly decided that ICALP 2015 will be held in Kyoto, Japan, and will be co-located with LICS 2015. Kazuo Iwama is the ICALP 2015 general chair. This will be the first ever ICALP outside Europe.

5 The Gender Challenge in TCS

During the general assembly, Orna Kupferman gave a thought-provoking talk on *The Gender Challenge in TCS*. It really got the audience thinking about this important matter.

After Orna's presentation, the President showed some EATCS-related data on the gender challenge.

- The EATCS Council has 28 members, including six women.
- The Gödel prize has gone to two female authors since 1993 (Shafi Goldwasser (1993 and 2001) and Éva Tardos (2012))
- One of the five Presburger Award recipients is a woman.
- At ICALP 2011 and 2013, $1/3$ of the PC chairs and $2/5$ of the invited speakers were women.

6 Report of the EATCS President:

Luca Aceto reports about the main activities of EATCS in the period from July 2012 till June 2013. (See also the Annual Activity Report on the EATCS website.) In particular, the President reports that the number of members slightly decreased during the last year down to 832 (not counting the ICALP 2013 registrations). Concerning the sponsoring situation, the President reports that the EATCS is still in good shape. All the sponsors (the Centum Wiskunde & Informatica (CWI), the Danish Center for Massive Data Algorithmics (MADALGO), the Greek Technology and Research Institute on Computers and Informatics (CTI), the publishing

house Springer, and Microsoft Research) have promised to extend their commitments.

One of the major activities of the EATCS is to promote research in theoretical computer science. The EATCS attempts to stimulate scientific excellence by supporting and encouraging the very best, truly creative scientists. In this context, the EATCS has contributed again to a variety of awards, namely, the Gödel Prize, the EATCS Award, the Presburger Award, and the Dijkstra Prize. The Gödel Prize 2013 was awarded to Antoine Joux for his paper "One Round Protocol for Tripartite Diffie-Hellman", and Dan Boneh, Matthew K. Franklin for their paper "Identity-Based Encryption from the Weil Pairing". The presentation took place this year at STOC. The EATCS Award 2013, which is sponsored by Springer, goes to Martin Dyer and the Presburger Award 2013, which is sponsored by CWI, is presented to Erik Demaine. The President reminded the attendees that on Tuesday afternoon of the ICALP week there was a special Award Ceremony where those two scientists were honoured. During that session, in addition to the presentation of the EATCS and the Presburger awards, honorary doctorates were bestowed on Jozef Gruska and Juris Hartmanis.

The decision on the Dijkstra Prize 2013 has not been specified at the time of the GA and will be announced at this year's PODC conference. Moreover, the EATCS was involved in various Best Paper Awards and Best Student Paper Awards at major conferences in theoretical computer science (ICALP, ETAPS, ESA, MFCS).

The President reported on the following recent developments:

- The EATCS Young Researcher School Series will kick off in 2014. The first installment will be a school on *Automata, Logic and Games* organized by Tony Kucera.

- The EATCS Fellows Programme will be launched soon. The deadline for nominations is 31 December 2013 and the first fellows will be announced at ICALP 2014.

- The EATCS participates in the new EATCS-IPEC Nerode Prize for outstanding papers in the area of multivariate algorithmics.

- From this year, there is a five euro discount on the EATCS registration fee for those who register to both the EATCS and one of its chapters.

- Kazuo Iwama will be the new editor in chief of the Bulletin of the EATCS from the October 2013 issue.

7 EATCS Council Election 2013

The President reports that every odd year, half of the members of the council are elected for a term of four years, and every even year, the officials are elected for a term of two years. Consequently, this year the term of office of the following ten council members that were elected in 2009 expires: Susanne Albers (DE), Leslie Ann Goldberg, (UK), Giuseppe F. Italiano (IT), Christos Kaklamanis (GR), Catuscia Palamidessi (FR), Giuseppe Persiano (IT), Jean-Eric Pin (FR), Thomas Wilke (DE). In September EATCS members will be asked to elect up to ten Council members through an electronic vote. The President reported the list of nominations received before the GA. According to the decision taken at the Council in 2012 all nominees (who are willing to accept to be candidates) will be invited to post a brief statement of their views on EATCS in their web sites. The electronic ballot will be held in the period September 19th–October 19th. After brief discussion the GA approved the following list of nominees: Mikolaj Bojanczyk (PL), Pierre Fraigniaud (FR), Leszek Gasieniec (UK), Leslie-Ann Goldberg (UK), Christos Kaklamanis (GR), Alberto Marchetti Spaccamela (IT), Uwe Nestmann (DE), Catuscia Palamidessi (FR), Giuseppe Persiano (IT), Andrew Pitts (UK), Alberto Policriti (IT), Dimitrios M. Thilikos (FR and GR), Pascal Weil (FR), Thomas Wilke (DE), Peter Widmayer (CH).

8 Report of the EATCS Treasurer

There was no report from the treasurer during the GA. The President thanked Dirk Janssens for his wonderful work and summarized the financial status by saying that the EATCS is in good shape.

The financial situation of the EATCS is still solid, and indeed, the balance for the period June 15, 2012 to June 15, 2013 is slightly positive. The surplus amounted to roughly 13 thousand Euros. Overall, the EATCS has approximately 178 thousand Euros at its disposal. The financial data may be found in the EATCS Annual Report.

9 Report of the EATCS Secretary

There was no report from the EATCS Secretary at the GA. All the information can be found in the EATCS Annual Report. Luca Aceto thanked the EATCS Secretary office, which is doing a truly outstanding job for the association.

10 Miscellaneous

In conclusion, the President also expressed his most sincere thanks to all those present for their contributions and interest, and closed the 2013 General Assembly of the EATCS at 20:10, Wednesday, 10 July 2013.

The EATCS Award 2014

Call for Nominations

Deadline: December 31, 2013.

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 2014, candidates may be nominated to the Award Committee consisting of

- Leslie Ann Goldberg (University of Oxford)
- Kim Guldstrand Larsen (Aalborg University)
- Vladimiro Sassone (University of Southampton)

The deadline for nominations is **December 31, 2013**. 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:

Leslie Ann Goldberg,
Department of Computer Science,
University of Oxford,
Wolfson Bldg, Parks Rd,
Oxford OX1 3QD United Kingdom
Email: leslie.goldberg@cs.ox.ac.uk

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)

The next award will be presented during ICALP 2014 in Copenhagen, Denmark.

Gödel Prize 2014

Call for Nominations

Deadline: January 17, 2014.

The Gödel Prize for outstanding papers in the area of theoretical computer science is sponsored jointly by the European Association for Theoretical Computer Science (EATCS) and the Association for Computing Machinery, Special Interest Group on Algorithms and Computation Theory (ACM-SIGACT). The award is presented annually, with the presentation taking place alternately at the International Colloquium on Automata, Languages, and Programming (ICALP) and the ACM Symposium on Theory of Computing (STOC). The 22nd GÃűdel Prize will be awarded at the 41st ICALP in Copenhagen in July 2014.

The Prize is named in honor of Kurt GÃűdel in recognition of his major contributions to mathematical logic and of his interest, discovered in a letter he wrote to John von Neumann shortly before von NeumannâĂŹs death, in what has become the famous âĂIJP versus NPâĂİ question. The Prize includes an award of USD 5000. Award Committee: The winner of the Prize is selected by a committee of six members. The EATCS President and the SIGACT Chair each appoint three members to the committee, to serve staggered three-year terms. The committee is chaired alternately by representatives of EATCS and SIGACT. The 2014 Award Committee consists of Krzysztof Apt (CWI Amsterdam), Giuseppe F. Italiano (UniversitÃă di Roma Tor Vergata), Joseph Mitchell (State University of New York at Stony Brook), Andrew Pitts (University of Cambridge), Daniel Spielman (Yale University), and ÃL'va Tardos (Cornell University).

Eligibility: The rules for the 2014 Prize are given below and they supersede any different interpretation of the generic rule to be found on websites of both SIGACT and EATCS. Any research paper or series of papers by a single author or by a team of authors is deemed eligible if

(i) the paper was published in a recognized refereed journal no later than December 31, 2013;

(ii) the main results were not published (in either preliminary or final form) in a journal or conference proceedings before January 1st, 2001.

The research work nominated for the award should be in the area of theoretical computer science. The term âĂIJtheoretical computer scienceâĂİ is meant to encompass, but is not restricted to, research areas covered by ICALP and STOC. Nominations are encouraged from the broadest spectrum of the theoretical computer science community so as to ensure that potential award winning papers are not overlooked. The Award Committee shall have the ultimate authority to decide whether a particular paper is eligible for the Prize.

Nominations: Nominations for the award should be submitted by email to the Award Committee Chair

Giuseppe F. Italiano: goedelprize at gmail dot com

Please make sure that the Subject line of all nominations and related messages begin with Goedel Prize 2014. To be considered, nominations for the 2014 Prize must be received by **January 17, 2014.**

Any member of the scientific community can make nominations. The Award Committee may actively solicit nominations. A nomination should contain a brief summary of the technical content of the paper(s) and a brief explanation of its significance. A printable copy of the research paper or papers should accompany the nomination. The nomination must state the date and venue of the first conference or workshop publication or state that no such publication has occurred. The work may be in any language. However, if it is not in English, a more extended summary written in English should be enclosed. To be considered for the award, the paper or series of papers must be recommended by at least two individuals, either in the form of two distinct nominations or one nomination including recommendations from two different people. Additional recommendations may also be enclosed and are generally useful. The Award Committee encourages recommendation and support letters to be mailed separately, without being necessarily shared with the nominator(s). The rest of the nomination package should be sent in a single email whenever possible. Those intending to submit a nomination should contact the Award Committee Chair by email well in advance. The Chair will answer questions about eligibility, encourage coordination among different nominators for the same paper(s), and also accept informal proposals of potential nominees or tentative offers to prepare formal nominations. The committee maintains a database of past nominations for eligible papers, but fresh nominations for the same papers (especially if they highlight new evidence of impact) are always welcome.

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

process left unspecified in this document are left to the discretion of the Award Committee.

Recent Winners
(all winners since 1993 listed at http://www.sigact.org/Prizes/Godel/):

2013: Antoine Joux, **"A One Round Protocol for Tripartite Diffie-Hellman,"** ***J. Cryptology,*** **17 (4) (2003), 263–276.**

Dan Boneh, Matthew K. Franklin, **"Identity-Based Encryption from the Weil Pairing,"** ***SIAM Journal of Computing,*** **32(3) (2003), 586–615.**

2012: Tim Roughgarden, ÃL'va Tardos, **"How bad is selfish routing?"** ***Journal of the ACM,*** **49 (2) (2002), 236âĂŞ-259.**

Dan Boneh, Matthew K. Franklin, **"Identity-Based Encryption from the Weil Pairing,"** ***SIAM Journal of Computing,*** **32(3) (2003), 586–615.**

Noam Nisan and Amir Ronen, **"Algorithmic Mechanism Design,"** ***Games and Economic Behavior,*** **35(1-2) (2001), 166âĂŞ-196.**

2011: Johan Hástad, **"Some optimal inapproximability results,"** ***Journal of the ACM,*** **48 (2001), 2798âĂŞ-859.**

2010: S. Arora, **"Polynomial-time approximation schemes for Euclidean TSP and other geometric problems,"** ***Journal of the ACM,*** **45(5) (1998), 753–782.**

J.S.B. Mitchell, **"Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems,"** ***SIAM J. Computing,*** **28(4) (1999), 1298–1309.**

The Presburger Award for Young Scientists 2014

Call for Nominations

Deadline: December 31, 2013.

Starting in 2010, the European Association for Theoretical Computer Science (EATCS) established the **Presburger Award**. The Award is conferred annually at the International Colloquium on Automata, Languages and Programming (ICALP) to a young scientist (in exceptional cases to several young scientists) for outstanding contributions in theoretical computer science, documented by a published paper or a series of published papers. The Award is named after Mojżesz Presburger who accomplished his path-breaking work on decidability of the theory of addition (which today is called Presburger arithmetic) as a student in 1929.

Nominations for the Presburger Award can be submitted by any member or group of members of the theoretical computer science community except the nominee and his/her advisors for the master thesis and the doctoral dissertation. Nominated scientists have to be at most 35 years at the time of the deadline of nomination (i.e., for the Presburger Award of 2014 the date of birth should be in 1978 or later). The Presburger Award Committee of 2014 consists of Antonín Kučera (Brno, chair), Claire Mathieu (Paris), and Peter Widmayer (Zürich). 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:

Antonín Kučera
kucera@fi.muni.cz

The subject line of every nomination should start with *Presburger Award 2014*, and the message must be received before **December 31st, 2013**.

The award includes an amount of 1000 Euro and an invitation to ICALP 2014 for a lecture. The Presburger Award is sponsored by CWI, Centrum Wiskunde & Informatica.

Previous Winners:

- Mikołaj Bojańczyk, 2010
- Patricia Bouyer-Decitre, 2011
- Venkatesan Guruswami and Mihai Pătraşcu, 2012
- Erik Demaine, 2013

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

Call for Nominations For EATCS-Fellows 2014

Deadline: December 31, 2013.

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

- Rocco De Nicola (IMT Lucca, Italy)
- Paul Goldberg (Oxford, UK)
- Anca Muscholl (Bordeaux, France, chair)
- Dorothea Wagner (Karlsruhe, Germany)
- Roger Wattenhofer (ETH Zurich, Switzerland)

INSTRUCTIONS:

A nomination should consist of answers to the questions below. It can be co-signed by several EATCS members. At least 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 2014 must be received by **December 31, 2013.**

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

5. Submission:

Submit by **December 31** of the current year for Fellow consideration by email to the EATCS Secretary (secretary@eatcs.org).

The subject line of the email should read "EATCS Fellow Nomination - <surname of candidate>".

Institutional Sponsors

CTI, Computer Technology Institute
Patras, Greece

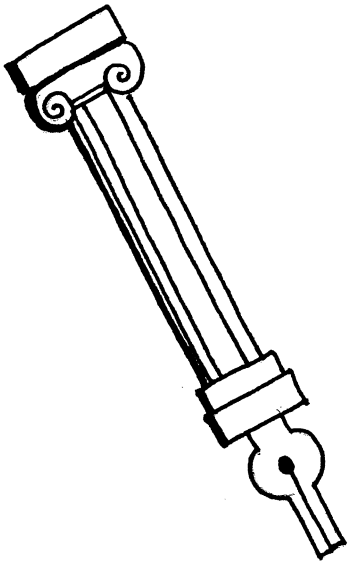
CWI, Centum Wiskunde
Informatica
Amsterdam, The Netherlands

MADALGO, Center for Massive Data Algorithmics
Aarhus, Denmark

Microsoft Research Cambridge
Cambridge, United Kingdom

Springer-Verlag
Heidelberg, Germany

EATCS
Columns



THE COMPUTATIONAL COMPLEXITY COLUMN

BY

VIKRAMAN ARVIND

Institute of Mathematical Sciences, CIT Campus, Taramani
Chennai 600113, India
arvind@imsc.res.in
http://www.imsc.res.in/~arvind

Expander graphs are of much importance in theoretical computer science, and the construction of expander graphs involves different areas of mathematics. It has attracted mathematicians and theoretical computer scientists alike and continues to be a flourishing area of research [14].

In this essay we discuss the Alon-Roichman theorem which states that for any finite group G, if S is a randomly picked multiset of $O(\log |G|)$ elements then the symmetric Cayley graph $\mathrm{Cay}(G, S)$ is a spectral expander with high probability. We explain a proof of this theorem based on Erdős-Rényi sequences, which are interesting in their own right, and also outline a $|G|^{O(1)}$ time derandomized construction of the set S.

We also discuss faster, $(\log |G|)^{O(1)}$ time, derandomizations of the Alon-Roichman theorem for finite groups given by small generating sets as input and raise some open questions.

The Alon-Roichman Theorem

V. Arvind
Institute of Mathematical Sciences, Chennai, India
arvind@imsc.res.in

1 Introduction

A primary research goal in the study of expander graphs is the construction of *explicit* expander graph families. Namely, we want to construct a family of graphs $\{G_n\}_{n\in\mathbb{N}}$, where G_n is typically an n vertex graph of small degree d (preferably constant) and the second largest eigenvalue of its normalized adjacency matrix A_G is bounded by a constant $\lambda < 1$. The graph G_n is said to be an (n, d, λ) spectral expander. It turns out that this spectral condition guarantees "high connectivity" for G_n as a result of which G_n has small diameter and, moreover, random walks on G_n converge "rapidly" to the uniform distribution. An excellent source for basic material, a wide range of applications as well as current research is the monograph on expander graphs by Hoory, Linial and Wigderson [14].

The usual source for explicit constructions of expanders is finite groups. Let G be a finite group and let $S = \{g_1, g_2, \ldots, g_k\}$ be a generator set for G. We form the symmetric Cayley graph $\mathrm{Cay}(G, S \cup S^{-1})$ whose vertex set is G and an unordered pair (x, y) is an edge in the graph if and only if $x^{-1}y \in S \cup S^{-1}$. Clearly, $\mathrm{Cay}(G, S \cup S^{-1})$ is a $2k$-regular multigraph. Suppose the group G has an explicit description (i.e. the elements of G have small encodings, are efficiently recognizable and the group operations can be efficiently performed). Furthermore, suppose the generating set S is explicit and of small size then the Cayley graph $\mathrm{Cay}(G, S \cup S^{-1})$ is explicit and has small degree. The best known explicit constructions of expander graphs are Cayley graphs of a subgroup of 2×2 matrices over a finite field $\mathbb{F}_p$. These expander graph families, known as Ramanujan graphs, have constant degree d and $\lambda = \Theta(1/\sqrt{d})$, which is optimal to a constant factor, and matches the λ for random d-regular multigraphs [17].

A general aspect in constructing such expander families lies, of course, in understanding which families of finite groups G_n have small size expanding generating sets S so that $\mathrm{Cay}(G, S \cup S^{-1})$ is a λ-spectral expander.

For any finite group G we know that it has a generating set of size $\log |G|$. Indeed, given G as a multiplication table we can compute a $\log |G|$ size generating set in $|G|^{O(1)}$ time. It is a simple greedy algorithm: having picked i elements $g_1, g_2, \ldots, g_i$ from G into the generating set we list out the elements of the subgroup H generated by the set $\{g_1, g_2, \ldots, g_i\}$. If $H \neq G$ we pick any $g_{i+1} \in G \setminus H$ as the next element in the generating

set. The new subgroup $\langle g_1, g_2, \ldots, g_{i+1}\rangle$ obtained contains H properly and its size is at least $2|H|$ by Lagrange's theorem (which states that the size of a finite group is divisible by the size of any subgroup of it). Hence, $\log|G|$ elements suffice to generate G. It turns out that $\log|G|$ is also optimal for certain finite groups. E.g. if G is the additive group $\mathbb{F}_2^n$ then a generating set for it is also a spanning set for the vector space $\mathbb{F}_2^n$ and hence has to have at least $n = \log|G|$ elements in it as $\mathbb{F}_2^n$ is n-dimensional over $\mathbb{F}_2$.

In general, for a finite group G, a natural question that arises is whether G also has an expanding generating set of size $\log|G|$ or at least $O(\log|G|)$. Alon and Roichman in [3] prove the following beautiful result which answers this question in the affirmative.

Theorem 1 (Alon-Roichman Theorem)**.** *[3] Let $\lambda > 0$ and G be any finite group. Then a random multiset $S \subset G$ of size $c\log|G|$ makes the symmetric Cayley graph* $\mathrm{Cay}(G, S \cup S^{-1})$ *a λ-spectral expander with high probability.*

The theorem suggests a simple efficient Las Vegas algorithm for the problem of computing an $O(\log|G|)$ size expanding generating set for G, where G is given as input by its multiplication table: we sample $c\log|G|$ many elements from G uniformly at random with replacement to obtain the multiset S. We can check in $|G|^{O(1)}$ time if $\mathrm{Cay}(G, S \cup S^{-1})$ is a λ-spectral expander by estimating its second largest eigenvalue and checking that it is bounded in magnitude by λ.

A natural question is whether we can compute such an expanding generating set in deterministic $|G|^{O(1)}$ time. This is along the lines of constructing explicit expander families, and it was answered in the affirmative by Wigderson and Xiao [25] who gave an efficient derandomization of the Alon-Roichman theorem using a representation-theoretic approach. More precisely, in [25] they use Chernoff bounds for matrix-valued random variables (due to Ahlswede and Winter [1]) combined with the application of the method of conditional probabilities [21]. This representation-theoretic approach to the Alon-Roichman theorem is based on alternative proofs of the theorem due to [15, 16]. The original proof of Alon and Roichman [3] is combinatorial in flavour. Igor Pak [20] gives another combinatorial proof for the Alon-Roichman theorem based on Erdős-Rényi sequences [12]. In this essay we give a somewhat different account of Pak's proof which is amenable to a derandomized construction [6]. This actually yields a $|G|^{O(1)}$ time combinatorial derandomization of Alon-Roichman, which is quite different from the previously mentioned one [25]. In the second part of this article we consider finite groups G given by small generating sets as input and address the question of $(\log|G|)^{O(1)}$ time derandomization of the Alon-Roichman theorem for some interesting classes of groups.

2 Randomized construction

We now discuss a version of Pak's proof which is amenable to derandomization [6]. The connection between mixing times of random walks on a graph and its spectral expansion is well studied. For undirected graphs we have the following.

Theorem 2. [22, Theorem 1] *Let A be the normalized adjacency matrix of an undirected graph. For every initial distribution, suppose the distribution obtained after t steps of the random walk following A is ϵ-close to the uniform distribution in the L_1 norm. Then the spectral gap $(1 - |\lambda_1|)$ of A is $\Omega(\frac{1}{t}\log\left(\frac{1}{\epsilon}\right))$.*

Even for directed graphs a connection between mixing times of random walks and the spectral properties of the underlying Markov chain is known.

Theorem 3. [19, Theorem 5.9] *Let λ_{max} denote the second largest magnitude (complex valued) eigenvalue of the normalized adjacency matrix P of a strongly connected aperiodic Markov Chain. Then the mixing time is lower bounded by $\tau(\epsilon) \geq \frac{\log(1/2\epsilon)}{\log(1/|\lambda_{max}|)}$, where ϵ is the difference between the resulting distribution and the uniform distribution in the L_1 norm.*

In [20], Pak uses this connection to prove an analogue of the Alon-Roichman theorem for *directed* Cayley graphs.

Let D_1 and D_2 be probability distributions on the set $\{1, 2, \ldots, n\}$. We use the L_2 norm $\|D_1 - D_2\|_2 = \left[\sum_{x\in[n]} |D_1(x) - D_2(x)|^2\right]^{\frac{1}{2}}$ to measure the distance between them.

We say that a distribution D is δ-close to the uniform distribution U, if $\|D-U\|_2 \leq \delta$. The *collision probability* of a distribution D is $\mathrm{Coll}(D) = \sum_{i\in[n]} D(i)^2$. It is easily seen that $\mathrm{Coll}(D) \leq 1/n + \delta$ if and only if $\|D - U\|_2^2 \leq \delta$ and $\mathrm{Coll}(D)$ attains its minimum value $1/n$ if and only if $D = U$.

Let G be an n-element group. For a sequence of group elements $J = \langle g_1, \ldots, g_k\rangle$ in G, consider the directed Cayley graph $\mathrm{Cay}(G, J)$, which is a multigraph with in-degrees and out-degrees of all vertices equal to k. Let A denote the adjacency matrix of $\mathrm{Cay}(G, J)$. Consider the "lazy" random walk defined by the probability transition matrix $(A + I)/2$ where I is the identity matrix. That is to say, with probability $1/2$ the random walk stays at the same vertex and with probability $1/2$ it moves to one of its k out-neighbors (each destination with probability $1/2k$).

Let Q_J be the probability distribution after m steps of the lazy random walk. Strictly speaking, Q_J depends on the initial distribution. However, we wish to bound the worst-case distance $\|Q_J - U\|_2$ of Q_J from the uniform. Hence the initial distribution does not matter. Pak [20] has analyzed Q_J and shown that for a random J of $O(\log n)$ size and $m = O(\log n)$, Q_J is $1/n^{O(1)}$-close to the uniform distribution. Pak works with the L_∞ norm. Since our aim is to give a derandomization of this construction, the L_2 norm and the collision probability are the right objects to work with since we can compute these quantities exactly as we fix elements of J one by one in the derandomization.

Pak's randomized construction is based on *Erdős-Rényi sequences* for finite groups introduced by Erdős and Rényi in [12].

Definition 4. *Let G be a finite group and $J = \langle g_1, \ldots, g_k\rangle$ be a sequence of elements in G. For $\delta > 0$, J is an* Erdős-Rényi sequence *for G with* closeness parameter *δ, if the probability distribution D_J on G given by $g_1^{\epsilon_1} \ldots g_k^{\epsilon_k}$, where the $\epsilon_i \in \{0, 1\}$ are independent unbiased random bits, is δ-close to the uniform distribution in the L_2-norm.*

Erdős and Rényi proved the following theorem.

Theorem 5 (Erdős Rényi)**.** *([12]) Let G be a finite group and U be the uniform distribution on G. Let $J = \langle g_1, \ldots, g_k \rangle$ denote a sequence of k elements of G picked independently and uniformly at random. Then the expected value*

$$\mathbb{E}_J \|D_J - U\|_2^2 \;=\; 1/2^k(1 - 1/n).$$

In particular, it implies that a random sequence J of $O(\log n)$ elements is an *Erdős-Rényi sequence* for G with *closeness parameter* $1/n^{O(1)}$.

Consider any m-length sequence $I = \langle i_1, \ldots, i_m \rangle \in [k]^m$, where i_j's are indices that refer to elements in the sequence J. Let R_I^J be the following probability distribution on G. For $g \in G$: $R_I^J(g) = \mathsf{Pr}_{\bar{\epsilon}}[g_{i_1}^{\epsilon_1} \cdot \ldots \cdot g_{i_m}^{\epsilon_m} = g]$, where $\bar{\epsilon} = (\epsilon_1, \ldots, \epsilon_m)$ and the $\epsilon_i \in \{0, 1\}$ are independent and uniformly random. For each $g \in G$ we have: $Q_J(g) = \frac{1}{k^m} \sum_{I \in [k]^m} R_I^J(g)$.

Each R_I^J is the distribution defined by the Erdős-Rényi sequence $\langle g_{i_1}, g_{i_2}, \ldots, g_{i_m} \rangle$. Hence, the above equation implies that the distribution Q_J is the average of R_I^J over $I \in [k]^m$.

The indices in $I \in [k]^m$ need not be distinct. Let $L(I)$ denote the subsequence of distinct indices in the order of their *first occurrence* in I, from left to right. We refer to $L(I)$ as the L-subsequence of I. The L-subsequence $L(I)$ also defines a probability distribution $R_{L(I)}^J$ on the group G.

For analyzing the random walk and the distribution G_J, it is more convenient to deal with $R_{L(I)}^J$ rather than R_I^J. Fortunately, we can show that the two are tied together pretty closely. More precisely, suppose the elements of J are picked from G independently and uniformly at random. Then we can show for each $I \in [k]^m$ that, in expectation, if $R_{L(I)}^J$ is δ-close to uniform distribution (in L_2 norm) then so is R_I^J. We state this in terms of collision probabilities.

Lemma 6. *For a fixed I, If $\mathbb{E}_J[\mathrm{Coll}(R_{L(I)}^J)] = \mathbb{E}_J[\sum_{g \in G} R_{L(I)}^J(g)^2] \leq 1/n + \delta$ then $\mathbb{E}_J[\mathrm{Coll}(R_I^J)] = \mathbb{E}_J[\sum_{g \in G} R_I^J(g)^2] \leq 1/n + \delta$.*

Pak actually proves a similar lemma for the L_∞ norm [20]. When elements of J are picked uniformly and independently from G, by Theorem 5,

$$\mathbb{E}_J[\mathrm{Coll}(R_{L(I)}^J)] = \mathbb{E}_J[\sum_{g \in G} R_{L(I)}^J(g)^2] = \frac{1}{n} + \frac{1}{2^\ell}(1 - \frac{1}{n}),$$

where ℓ is the length of the L-subsequence. Thus the expectation is small provided ℓ is large enough. It turns out, with an easy counting argument, that most $I \in [k]^m$ have sufficiently long L-subsequences (Lemma 7 below).

Lemma 7. [20] *For any k, ℓ, the probability that a sequence of length m over $[k]$ does not have an L-subsequence of length ℓ is at most $\frac{(ae)^{\frac{k}{a}}}{a^m}$ where $a = \frac{k}{\ell - 1}$.*

To ensure the above probability is bounded by $\frac{1}{2^m}$, it suffices to choose $m = \lceil \frac{(k/a)\log(ae)}{\log(a/2)} \rceil$. Here a is a constant so that both m and ℓ are $\Theta(k)$.

Lemma 8. $\mathbb{E}_J[\mathrm{Coll}(Q_J)] \leq \frac{1}{n} + \frac{1}{2^{\Theta(m)}}$.

Proof. We call $I \in [k]^m$ *good* if it has an L-subsequence of length at least ℓ, else we call it *bad*.

$$\begin{aligned}
\mathbb{E}_J[\mathrm{Coll}(Q_J)] &= \mathbb{E}_J[\sum_{g\in G} Q_J^2(g)] \\
&= \mathbb{E}_J[\sum_{g\in G} (\mathbb{E}_I(R_I(g))^2] \\
&\leq \mathbb{E}_J[\sum_{g\in G} \mathbb{E}_I(R_I^2(g))] \text{ By Cauchy-Schwartz inequality} \qquad (1)\\
&= \mathbb{E}_I[\mathbb{E}_J[\mathrm{Coll}(R_I)]] \\
&\leq \frac{1}{k^m}\mathbb{E}_J[\sum_{\substack{I\in[k]^m \\ I \text{ is good}}} \sum_{g\in G} (R_L^J(g))^2 + \sum_{\substack{I\in[k]^m \\ I \text{ is bad}}} 1] \\
&\leq \mathsf{Pr}_I[I \text{ is good}]\left(\frac{1}{n} + \frac{1}{2^\ell}\right) + \mathsf{Pr}_I[I \text{ is bad}] \qquad (2)
\end{aligned}$$

The last step follows from Lemma 6 and Theorem 5. Fix m in Lemma 7 to $O(\log n)$ such that $\mathsf{Pr}_I[I \text{ is bad}] \leq \frac{1}{2^m}$ and let $\ell = \Theta(m)$ to yield $\mathbb{E}_J[\mathrm{Coll}(Q_J)] \leq \frac{1}{n} + \frac{1}{2^{\Theta(m)}}$. In particular, m is chosen so that $\mathsf{Pr}_I[I \text{ is bad}] \leq \frac{1}{2^m}$. □

Clearly, $\frac{1}{2^{\Theta(m)}} < \frac{1}{n^c}$ for a given $c > 0$, by choosing $m = O(\log n)$. We also choose $\ell = \Theta(m)$ in the proof of Lemma 8. Then, from the relation that $m = \lceil \frac{(k/a)\log(ae)}{\log(a/2)} \rceil$, we fix k to be $O(\log n)$ suitably. Since random walks on $\mathrm{Cay}(G, J)$ mix well, as a consequence of Theorem 3 we obtain the following.

Theorem 9. [20] *Let $\lambda > 0$ and G be any finite group. Then, with high probability, a random multiset $J \subset G$ of size $c \log |G|$ makes the directed Cayley graph* $\mathrm{Cay}(G, J)$ *a spectral expander (i.e. its second largest eigenvalue in absolute value is bounded by ϵ).*

2.1 Derandomizing the construction

We outline a derandomization [6] of the randomized Cayley expanders $\mathrm{Cay}(G, J)$ given by Theorem 9.

Given a group G with n elements, we need to compute in deterministic $|G|^{O(1)}$ time, a multiset J of k group elements of G such that:

$$\mathrm{Coll}(Q_J) = \sum_{g\in G} Q_J(g)^2 \;\leq\; 1/n + 1/n^c, \qquad (3)$$

where $c > 0$ is a given constant and both k and m are $O(\log n)$. By Theorem 9 a random set J satisfies this with high probability. For each J observe, by the Cauchy-Schwartz inequality, that

$$\mathrm{Coll}(Q_J) = \sum_{g\in G} Q_J(g)^2 \leq \sum_{g\in G} \frac{1}{k^m} \sum_{I\in[k]^m} R_I^J(g)^2 = \frac{1}{k^m} \sum_{I\in[k]^m} \mathrm{Coll}(R_I^J). \tag{4}$$

Thus, it suffices to compute a multiset J of group elements such that the average collision probability $\frac{1}{k^m}\sum_{I\in[k]^m} \mathrm{Coll}(R_I^J) \leq 1/n + 1/n^c$.

We start with $J = \{X_1, \ldots, X_k\}$ where each X_i is an independent random variable uniformly distributed in G. By Theorem 9 (in particular from Equation 3), for a given $c > 1$ there are k and m, both $O(\log n)$ such that:

$$\mathbb{E}_J[\mathrm{Coll}(Q_J)] = \mathbb{E}_J[\mathbb{E}_{I\in[k]^m}\mathrm{Coll}(R_I^J)] \leq \frac{1}{n} + \frac{1}{n^c}. \tag{5}$$

The algorithm is based on the method of conditional probabilities. It fixes elements in J one by one. Suppose at the j^{th} stage, for $j < k$, $J = J_j = \{x_1, x_2, \ldots, x_j, X_{j+1}, \ldots, X_k\}$, where x_r $(1 \leq r \leq j)$ are fixed elements of G and the remaining X_s, $s = j+1, \ldots, k$ are still random variables such that $\mathbb{E}[\mathbb{E}_{I\in[k]^m}\mathrm{Coll}(R_I^J)] \leq 1/n + 1/n^c$, where the outer expectation is over these X_s.

It suffices to give a polynomial-time procedure that fixes X_{j+1} to an $x_{j+1} \in G$ such that $\mathbb{E}[\mathbb{E}_{I\in[k]^m}\mathrm{Coll}(R_I^J)] \leq 1/n + 1/n^c$. Given $J = J_j = \{x_1, \ldots, x_j, X_{j+1}, \ldots, X_k\}$ with j fixed elements and $k - j$ random elements, we partition $[k]^m$ into subsets $S_{r,\ell}$ where $I \in S_{r,\ell}$ if and only if there are exactly r indices in I from $\{1, \ldots, j\}$, and of the remaining $m - r$ indices of I there are exactly ℓ distinct indices.

An (r, ℓ)-*normal sequence* for J is a sequence $\langle n_1, n_2, \ldots, n_r, \ldots, n_{r+\ell}\rangle \in [k]^{r+\ell}$ such that n_s, $1 \leq s \leq r$ are in $\{1, 2, \ldots, j\}$ and the n_s, $s > r$ are *all distinct* and in $\{j+1, \ldots, k\}$. In other words, the first r indices (possibly with repetition) are from the fixed part of J and the last ℓ are all distinct indices from the random part of J.

It turns out that a sequence $I \in [k]^m$ can be transformed, by group conjugation, into (r, ℓ)-normal form such that the expected collision probability of the distribution generated by the (r, ℓ)-normal form gives an upper bound on $\mathbb{E}[\mathrm{Coll}(R_I^J]$. This upper bound plays the role of a pessimistic estimator in the derandomization.

In order to transform a sequence in $S_{r,\ell}$ into (r, ℓ)-normal form we will make repeated use of the fact that if $y \in G$ is picked uniformly at random and $x \in G$ be any element independent of y, then the distribution of xyx^{-1}, namely the x-conjugate of y, is uniform in G.

Let $I = \langle i_1, \ldots, i_m\rangle \in S_{r,\ell}$ be a sequence. Let $F = \langle i_{f_1}, \ldots, i_{f_r}\rangle$ be the index subsequence whose corresponding elements are from the fixed part $\{x_1, x_2, \ldots, x_j\}$ of J. Let $R = \langle i_{s_1}, \ldots, i_{s_{m-r}}\rangle$ be the index subsequence for the random part of J. Let $L = \langle i_{e_1}, \ldots, i_{e_\ell}\rangle$ be the L-subsequence in R. More precisely, notice that R is a sequence in $\{j+1, \ldots, k\}^{m-r}$ and L is the L-subsequence for R. The (r, ℓ)-normal sequence $\widehat{I}$ of $I \in S_{r,\ell}$ is $\langle i_{f_1}, \ldots, i_{f_r}, i_{e_1}, \ldots, i_{e_\ell}\rangle$.

Denote the elements of J by $g_t, 1 \leq t \leq k$, where $g_t = x_t$ for $t \leq j$ and $g_t = X_t$ for $t > j$. Consider the distribution R_I^J consisting of the products $g_{i_1}^{\epsilon_1} \dots g_{i_m}^{\epsilon_m}$ where $\epsilon_i \in \{0, 1\}$ are independent and uniformly picked at random. Then, using repeated conjugation of the group elements to move the fixed part to the left, we can write

$$g_{i_1}^{\epsilon_1} \dots g_{i_m}^{\epsilon_m} = g_{i_{f_1}}^{\epsilon_{f_1}} \dots g_{i_{f_r}}^{\epsilon_{f_r}} h_{e_1}^{\epsilon_{e_1}} \dots h_{e_\ell}^{\epsilon_{e_\ell}} y(\bar{\epsilon}),$$

where $y(\bar{\epsilon})$ is an element in G that depends on J, I and $\bar{\epsilon}$, where $\bar{\epsilon}$ consists of all the ϵ_j for $i_j \in I \setminus (F \cup L)$. Here, $(h_{e_1}, h_{e_2}, \dots, h_{e_\ell})$ is the sequence of all *independent* random elements in the above product $\prod_{a=1}^{m-r} h_{s_a}^{\epsilon_{s_a}}$ consisting of conjugates of the original L-subsequence in J.

Let J_I denote the multiset of group elements obtained from J by replacing the subset $\{g_{i_{e_1}}, g_{i_{e_2}}, \dots, g_{i_{e_\ell}}\}$ in J with $\{h_{e_1}, h_{e_2}, \dots, h_{e_\ell}\}$. Note that, in this substitution, we are replacing a uniformly distributed random variable $g_{i_{e_j}}$ over G with another uniformly distributed random variable h_{e_j} over G, where the later is obtained from the former by a conjugacy transformation. Clearly, J_I also has j fixed elements $x_1, x_2, \dots, x_j$ and $k - j$ uniformly distributed independent random elements. Recall that $\widehat{I} = \langle i_{f_1}, i_{f_2}, \dots, i_{f_r}, i_{e_1}, i_{e_2}, \dots, i_{e_\ell} \rangle$ is the (r, ℓ)-normal sequence for I. The probability distributions R_I^J and $R_{\widehat{I}}^{J_I}$ are compared in the following lemma.

Lemma 10. *For each $j \leq k$ and $J = \{x_1, \dots, x_j, X_{j+1}, \dots, X_k\}$ (where $x_1, \dots, x_j \in G$ are fixed elements and $X_{j+1}, \dots, X_k$ are independent uniformly distributed in G), and for each $I \in [k]^m$, $\mathbb{E}[\mathrm{Coll}(R_I^J)] \leq \mathbb{E}[\mathrm{Coll}(R_{\widehat{I}}^{J_I})]$, where $\mathbb{E}[\mathrm{Coll}(R_I^J)]$ is computed over random elements in J and $\mathbb{E}[\mathrm{Coll}(R_{\widehat{I}}^{J_I})]$ over random elements in J_I.*

It follows that $\mathbb{E}_J[\mathrm{Coll}(Q_J)] \leq \mathbb{E}_{J_I}\mathbb{E}_{I \in [k]^m}[\mathrm{Coll}(R_{\widehat{I}}^{J_I})]$. Furthermore, it turns out that given $J = \{x_1, \dots, x_j, X_{j+1}, \dots, X_k\}$ we can compute $\mathbb{E}_{J_I}\mathbb{E}_{I \in [k]^m}[\mathrm{Coll}(R_{\widehat{I}}^{J_I})]$ in deterministic polynomial (in n) time by reducing it to the problem of counting directed s-t paths in a weighted directed acyclic graph. This completes the proof outline of the following.

Theorem 11. [6] *Let G be a group with n elements, given as its multiplication table. For any constant $c > 1$, there is a deterministic* poly(n) *time algorithm that computes a generating set J of size $O(\log n)$ for the given group G, such that for any initial distribution on G the lazy random walk of $O(\log n)$ steps on the directed Cayley graph* $\mathrm{Cay}(G, J)$ *yields a distribution that is $\frac{1}{n^c}$-close (in L_2 norm) to the uniform distribution.*

Together with Theorem 3 this yields the following corollary.

Corollary 12. [6] *Given a finite group G and any $\epsilon > 0$, there is a deterministic polynomial-time algorithm to construct an $O(\log n)$ size generating set J such that* $\mathrm{Cay}(G, J)$ *is a spectral expander (i.e. its second largest eigenvalue in absolute value is bounded by ϵ).*

Undirected Cayley graphs

This approach can be adapted for undirected Cayley graphs as well [6]. The key point is a suitable generalization of Erdős-Rényi sequences. We consider the distribution on G defined by $g_1^{\epsilon_1} \dots g_k^{\epsilon_k}$ where $\epsilon_i \in_R \{-1, 0, 1\}$. Using these generalized Erdős-Rényi sequences we can analyze lazy random walks on the *undirected* Cayley graph $\mathrm{Cay}(G, J \cup J^{-1})$ for a random multiset J of $O(\log |G|)$ size. More precisely, we consider the lazy random walk described by the symmetric transition matrix $A_J = \frac{1}{3}I + \frac{1}{3k}(P_J + P_{J^{-1}})$ where P_J and $P_{J^{-1}}$ are the adjacency matrices of the directed Cayley graphs $\mathrm{Cay}(G, J)$ and $\mathrm{Cay}(G, J^{-1})$ respectively. We obtain the following results.

Theorem 13. [6] *Let G be a finite group of order n and $c > 1$ be any constant. There is a deterministic* poly(n) *time algorithm that computes a generating set J of size $O(\log n)$ for G, such that an $O(\log n)$ step lazy random walk on G, governed by the transition matrix A_J described above, is $\frac{1}{n^c}$-close to the uniform distribution, for any given initial distribution on G.*

Theorem 13 and the connection between mixing time and spectral expansion for undirected graphs given by Theorem 2 yields an alternative proof of the following [6].

Corollary 14. [25] *Given a finite group G by its multiplication table, there is a deterministic polynomial (in $|G|$) time algorithm to construct a generating set J such that $\mathrm{Cay}(G, J \cup J^{-1})$ is a spectral expander.*

3 Faster derandomizations

We now explore the question of computing expanding generating sets for finite groups G in time polynomial in $\log |G|$. Since every finite group G has a generating set of size $\log |G|$, we can assume that $G = \langle S \rangle$ is given as input by a small generating set S and the goal is to compute an expanding generating set for G in time polynomial in $\log |G|$ and $|S|$.

For example, let G be any subgroup of the group S_n. Then G has a generating set $S \subset S_n$ of size at most $n \log n$. Furthermore, the group operation is permutation composition, and for two given permutations $\pi, \pi' \in S_n$ we can compute $\pi\pi'$ in time polynomial in n.

Another example: consider subgroups $G = \langle S \rangle$ of the group of invertible $n \times n$ matrices over a finite field $\mathbb{F}_q$ under matrix product. Matrix product can be performed in time polynomial in n and $\log q$ and every such group has a generating set of size $n^2 \log q$ (as there are at most q^{n^2} many invertible $n \times n$ matrices).

An algorithmic framework for finite groups input by their generating sets is the notion of *black-box groups* due to Babai and Szemerédi [10]. The elements of a finite black-box group G are assumed to be uniformly encoded as binary strings of some length m (where m would typically be polynomial in $\log |G|$). Each group operation

is performed by a black-box in time polynomial in m. The group G is given by a generating set S.

In a general result about finite black-box groups Babai has shown [9] that it is possible to sample nearly uniformly in polynomial time from $G = \langle S \rangle$, where S is an arbitrary generating set. Interestingly, Babai's sampling algorithm is based on Erdős-Rényi sequences. The randomized algorithm computes with high probability an Erdős-Rényi sequence $\{g_1, g_2, \ldots, g_k\}$ for G with closeness parameter $2^{-O(m)}$, where k is polynomial in m. Once we have an Erdős-Rényi sequence the sampling algorithm simply outputs $\prod_{i=1}^{k} g_i^{\varepsilon_i}$ where each $\varepsilon_i \in \{0, 1\}$ is independently and uniformly picked at random. We can summarize this result as follows.

Theorem 15 (Babai)**.** [9] *Let $G = \langle S \rangle$ be a finite black-box group whose elements uniformly encoded as binary strings length m. Then there is a randomized* $\mathrm{poly}(m, |S|)$ *time algorithm that outputs with high probability an Erdős-Rényi sequence with closeness parameter $2^{-O(m)}$. As a consequence, there is a randomized* $\mathrm{poly}(m, |S|)$ *time algorithm for sampling almost uniformly at random from G.*

Therefore, by the Alon-Roichman theorem we have a randomized polynomial-time algorithm for the problem (although it is not Las Vegas since we do not know how to certify an expanding generator set in polynomial time). More precisely, we have the following consequence of Alon-Roichman for general black-box groups.

Proposition 16. *Given $\lambda > 0$ and a finite black-box group $G = \langle S \rangle$ whose elements uniformly encoded as binary strings length m. There is a randomized* $\mathrm{poly}(m, |S|, 1/\lambda)$ *time Monte-Carlo algorithm that outputs with high probability an expanding generating set T of size $(O(\log |G|/\lambda^2)$ for G such that* $\mathrm{Cay}(G, T \cup T^{-1})$ *is a λ-spectral expander.*

The algorithmic question we now address is to obtain a *deterministic* polynomial (in $\log |G|$ and $|S|$) time algorithm for computing small expanding generating sets for G. A precise formulation of the problem is as follows:

Problem 17. *Given a finite group $G = \langle S \rangle$ by a small generating set S and a $\lambda > 0$ the problem is to compute, in deterministic time polynomial in $|S|$, $\log |G|$, and $1/\lambda$, a generating set T for G such that $|T| = O(\log |G|/\lambda^2)$ and* $\mathrm{Cay}(G, T \cup T^{-1})$ *is a λ-spectral expander.*

In Problem 17, the real challenge seems to be computing an expanding generating set T of the size $O(\log |G|/\lambda^2)$ promised by the Alon-Roichman theorem. We will discuss deterministic polynomial time algorithms that compute somewhat larger generating sets.

3.1 Small bias spaces

We will first consider the additive group $\mathbb{F}_2^n$ which is the simplest of groups. Its elements are the 2^n binary vectors and the group operation is coordinate-wise addition

modulo 2. The group is abelian and each nonzero element has order 2. By the Alon-Roichman theorem, for any $\epsilon > 0$ the group $\mathbb{F}_2^n$ has an expanding generating set T of size $O(n/\epsilon^2)$ that makes the Cayley graph ϵ-spectral.

Although we do not know any polynomial-time deterministic construction of size $O(n/\epsilon^2)$, it turns out that we can compute T of size $O(n^2/\epsilon^2)$ or of size $O(n/\epsilon^{O(1)})$. This is because expanding generating sets for $\mathbb{F}_2^n$ are precisely ϵ-bias spaces in $\mathbb{F}_2^n$ whose constructions are well studied in the context of almost k-wise independent sample spaces [5, 2].

We explain this connection between small bias spaces in $\mathbb{F}_2^n$ and expanding generating sets for $\mathbb{F}_2^n$. We will require some elementary group representation theory. A *character* χ of the group $\mathbb{F}_2^n$ is a *group homomorphism* from $\mathbb{F}_2^n$ to the multiplicative group of complex numbers $\mathbb{C}^*$. As all elements of $\mathbb{F}_2^n$ are or order either 1 or 2, and χ is a group homomorphism, it follows that $\chi(a) \in \{-1, 1\}$ for each $a \in \mathbb{F}_2^n$. The *trivial* character χ_0 maps all elements to 1. Each vector $b \in \mathbb{F}_2^n$ defines a character

$$\chi_b(a) = (-1)^{a.b},$$

where $a.b = \sum_i a_i b_i \bmod 2$.

The set of all functions $f : \mathbb{F}_2^n \to \mathbb{C}$ forms a 2^n-dimensional vector space over $\mathbb{C}$ and it turns out that the set of characters $\{\chi_b \mid b \in \mathbb{F}_2^n\}$ spans the vector space. This is a consequence of the fact that these 2^n characters $\chi_b, b \in \mathbb{F}_2^n$ are mutually orthogonal under the inner product $\langle f, g\rangle = \frac{1}{2^n} \sum_{a\in\mathbb{F}_2^n} f(a)\overline{g(a)}$.

Now, consider a generating set $S \subset \mathbb{F}_2^n$ for the group $\mathbb{F}_2^n$ and the resulting symmetric Cayley graph $\mathrm{Cay}(\mathbb{F}_2^n, S)$. Note that any subset S is already symmetric as $S = S^{-1}$. Let A_S denote its normalized adjacency matrix. The following nice fact about its eigenvectors suitably generalizes to the setting of all abelian groups.

Claim 18. *The vectors $\{\chi_b \mid b \in \mathbb{F}_2^n\}$ are the eigenvectors of the symmetric matrix A_S.*

Indeed, an easy calculation shows that the vector $A_S\chi_b = (1/|S| \sum_{s\in S} \chi_b(s))\chi_b$. Thus the eigenvalues of A_S are $\frac{1}{|S|} \sum_{s\in S} \chi_b(s)$ for $b \in \mathbb{F}_2^n$. The trivial character χ_0 is the eigenvector for eigenvalue 1. We can summarize the discussion in the following.

Proposition 19 (folklore)**.** *The Cayley graph* $\mathrm{Cay}(\mathbb{F}_2^n, S)$ *is an ϵ-spectral expander if and only if for all nontrivial characters χ_b*

$$\frac{1}{|S|}\left|\sum_{s\in S} \chi_b(s)\right| \le \epsilon.$$

The latter condition is precisely the definition of an ϵ-bias space. The known deterministic efficient constructions of Alon et al [4] of size $O(n^2/\epsilon^2)$ and [5] of size $O(n/\epsilon^{O(1)})$ fall short of constructing $O(n/\epsilon^2)$ size ϵ-bias spaces promised by the Alon-Roichman theorem.

3.2 General case: Divide and Conquer Constructions

We now consider deterministic construction of expanding generating sets for more general groups. Let $G = \langle g_1, g_2, \ldots, g_k \rangle$ be a finite group given by generators g_i. We will now outline a divide and conquer strategy [7] for the problem of computing expanding generating sets that works quite well for a large class of finite groups. The idea is to decompose G into smaller groups, compute expanding generating sets for the smaller groups and put these generating sets together suitably for G.

Exploiting normal subgroups

Let G be a finite group and N be a *normal* subgroup of G. I.e. N is a subgroup such that $g^{-1}Ng = N$ for all $g \in G$. Suppose $A \subset N$ is an expanding generating set for N so that $\mathrm{Cay}(N, A \cup A^{-1})$ is a λ-spectral expander. Similarly, consider the quotient group G/N (which is well defined by virtue of N's normality). Suppose $B \subset G$ such that $\hat{B} = \{Nx \mid x \in B\}$ is an expanding generating set for the quotient group G/N and the corresponding Cayley graph $\mathrm{Cay}(G/N, \hat{B} \cup \hat{B}^{-1})$ is also λ-spectral. Then we can prove the following.

Lemma 20. [7] *Suppose both* $\mathrm{Cay}(N, A \cup A^{-1})$ *and* $\mathrm{Cay}(G/N, \hat{B} \cup \hat{B}^{-1})$ *are* λ*-spectral and let* $C = A \cup B$*. Then* $\mathrm{Cay}(G, C \cup C^{-1})$ *is a* $(1 + \lambda)/2$*-spectral expander.*

See [7] for proof details. The overall idea is similar in spirit to the analysis of the zig-zag product construction [24]. There are some additional issues in this construction that makes $C \cup C^{-1}$ an expanding generating set for G which are taken care of because N is a normal subgroup of G. This theorem provides us a divide-and-conquer tool in the following sense. Suppose G is a finite group with a *normal series*

$$G = G_0 \rhd G_1 \rhd \cdots \rhd G_r = \{1\},$$

where each G_i is a normal subgroup of G. I.e. $G \rhd G_i$ for each i. Suppose we are given expanding generating sets for each quotient group G_i/G_{i+1}. Using the above theorem we can put them together efficiently to construct an expanding generator set for G.

Lemma 21. [7] *Let* $G \leq S_n$ *with normal series* $\{G_i\}_{i=0}^{r}$ *be as above. Further, for each* i *let* B_i *be a generator set for* G_i/G_{i+1} *such that* $\mathrm{Cay}(G_i/G_{i+1}, B_i)$ *is a* $1/4$*-spectral expander. Let* $s = \max_i\{|B_i|\}$*. Then in deterministic time polynomial in* n *and* s *we can compute a generator set* B *for* G *such that* $\mathrm{Cay}(G, B)$ *is a* $1/4$*-spectral expander and* $|B| = c^{\log^r} s$ *for some constant* $c > 0$*.*

The proof of this lemma is essentially based on repeated application of Lemma 20.

The case of solvable permutation groups

In order to actually apply Lemma 21 we need to compute a normal series for $G = \langle S \rangle$ efficiently. Moreover, we will also need to compute expanding generating sets for the

quotient groups G_i/G_{i+1}. A large class of groups for which this approach works well is solvable permutation groups. First we recall some definitions.

Let G be a finite group. The *commutator subgroup* of G is the subgroup G' generated by elements $xyx^{-1}y^{-1}$ where $x, y \in G$. The commutator subgroup G' is the minimal normal subgroup of G such that the quotient G/G' is abelian. The *derived series* for G is the following chain of subgroups of G:

$$G = G_0 \triangleright G_1 \triangleright \cdots \triangleright G_k$$

where, for each i, G_{i+1} is the *commutator subgroup* of G_i.

The group G is said to be *solvable* if the derived series terminates in $G_k = \{1\}$ for some k, where k is the *length* of the derived series for G. We note that the derived series for G is a normal series.

A group G is *non-solvable* if the derived series does not terminate at $\{1\}$. For instance, the group $G = A_5$, consisting of all even permutations on five elements, is non-solvable because $G_1 = G$.

A *permutation group* is a subgroup $G = \langle S \rangle$ of the group S_n of all permutations on $[n]$. Given a permutation group $G = \langle S \rangle$ by a generating set we can compute its derived series in deterministic polynomial time [18]. Dixon [11] has shown that derived series of solvable subgroups of S_n have length bounded by $5 \log_3 n$. The above observations combined with Theorem 21 yields the following consequence.

Lemma 22. *Suppose $G \leq S_n$ is a solvable group with derived series*

$$G = G_0 \triangleright G_1 \triangleright \cdots \triangleright G_k = \{1\},$$

and we have $B_i \subset G_i/G_{i+1}$ such that $\mathrm{Cay}(G_i/G_{i+1}, B_i \cup B_i^{-1})$ *is a* $1/4$*-spectral expander. Let $s = \max_i\{|B_i|\}$. Then in deterministic* $\mathrm{poly}(n, s)$ *time we can compute a subset B of G such that* $\mathrm{Cay}(G, B \cup B^{-1})$ *is a* $1/4$*-spectral expander and $|B| = 2^{O(\log k)}s = (\log n)^{O(1)}s$.*

In order to compute an expanding generating set for a solvable subgroup G of S_n we first need to compute an expanding generating set B_i for G_i/G_{i+1} such that $\mathrm{Cay}(G_i/G_{i+1}, B_i)$ is $1/4$-spectral and then apply the above lemma.

Abelian quotient groups

We now explain the computation of expanding generating sets for the abelian quotient groups G_i/G_{i+1}, where $G_{i+1} \lhd G_i \leq S_n$. Let $p_1 < p_2 < \ldots < p_k$ be the list of all primes bounded by n. Let $e = \lceil \log n \rceil$. As G_i/G_{i+1} is abelian, there is an onto group homomorphism ϕ from the product group $\mathbb{Z}^n_{p_1^e} \times \mathbb{Z}^n_{p_2^e} \times \cdots \times \mathbb{Z}^n_{p_k^e}$ to G_i/G_{i+1}. Moreover, this homomorphism is easily computable. It suffices to compute an expanding generating set for $\mathbb{Z}^n_{p_1^e} \times \mathbb{Z}^n_{p_2^e} \times \cdots \times \mathbb{Z}^n_{p_k^e}$ because its ϕ-image will be an expanding generating set for G_i/G_{i+1}.

It turns out that we can compute an expanding generating set for $\mathbb{Z}^n_{p_1^e} \times \mathbb{Z}^n_{p_2^e} \times \cdots \times \mathbb{Z}^n_{p_k^e}$ of size $\tilde{O}(n^2)$ in polynomial time [7]. This construction is again a careful application of Lemma 20 combined with a result of Ajtai et al [2] of expanding generating sets for cyclic groups Z_N.

Theorem 23. [7] *Let $G = \langle S \rangle$ be a solvable subgroup of S_n. In deterministic polynomial time we can compute an expanding generating set of size $\tilde{O}(n^2)$ such that the Cayley graph* $\mathrm{Cay}(G, S \cup S^{-1})$ *is a* $1/4$*-spectral expander.*

For general permutation groups we have the following theorem based on derandomized squaring [23] about computing expanding generator sets. Details are given in [7].

Theorem 24. *Given $G \leq S_n$ by a generator set S' and $\lambda > 0$, we can deterministically compute (in time* $\mathrm{poly}(n, |S'|)$*) an expanding generator set T for G such that* $\mathrm{Cay}(G, T)$ *is a λ-spectral expander and $|T| = O(n^{16q+10}\left(\frac{1}{\lambda}\right)^{32q})$, where q is a constant.*

Small Bias Spaces for $\mathbb{Z}^n_d$

In conclusion, we note that the expanding generating set construction for abelian groups $\mathbb{Z}^n_{p_1^e} \times \mathbb{Z}^n_{p_2^e} \times \cdots \times \mathbb{Z}^n_{p_k^e}$ mentioned above also gives a new construction of ε-bias spaces for $\mathbb{Z}^n_d$, which we now describe.

In [8] Azar, Motwani, and Naor first considered the construction of ϵ-bias spaces for abelian groups, specifically for the group $\mathbb{Z}^n_d$. For arbitrary d and any $\epsilon > 0$ they construct ϵ-bias spaces of size $O((d+n^2/\epsilon^2)^C)$, where C is the constant in Linnik's Theorem. The construction involves finding a suitable prime (or prime power) promised by Linnik's theorem which can take time up to $O((d+n^2)^C)$. The current best known bound for C is $\leq 11/2$ (and assuming ERH it is 2). Their construction yields a polynomial-size ϵ-bias space for $d = n^{O(1)}$.

It is interesting to compare this result of [8] with the construction described above. Let d have prime factorization $p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}$. Each p_i is $O(\log d)$ bit sized and each e_i is bounded by $O(\log d)$. Given d in unary, we can efficiently find the prime factorization of d. Using the result of Wigderson and Xiao [25], we compute an $O(\log d)$ size expanding generator set for $\mathbb{Z}_{p_1 p_2 \cdots p_k}$ in deterministic time polynomial in d. Then we construct an expanding generator set of size $O((\log n)^{O(1)} \log d)$ for $\mathbb{Z}^m_{p_1} \times \ldots \times \mathbb{Z}^m_{p_k}$ for $m = O(\log n)$ based on Lemma 22. It then follows that we can construct an $O(n(\log n)^{O(1)} \log d)$ size expanding generator set for $\mathbb{Z}^n_{p_1} \times \ldots \times \mathbb{Z}^n_{p_k}$ in deterministic polynomial time. Finally, it follows that we can construct an $O(n(\log n \log d)^{O(1)})$ size expanding generator set for $\mathbb{Z}^n_d$ (which is isomorphic to $\mathbb{Z}^n_{p_1^{e_1}} \times \ldots \mathbb{Z}^n_{p_k^{e_k}}$) since each e_i is bounded by $\log d$. Given $\epsilon > 0$, the dependence of ϵ in the size of the generating set that makes the Cayley graph λ-spectral is $(1/\epsilon)^{32q}$.

Theorem 25. *Let d, n be any positive integers (in unary) and $\varepsilon > 0$. Then, in deterministic* $\mathrm{poly}(n, d, \frac{1}{\epsilon})$ *time, we can construct an* $O(n\mathrm{poly}(\log n, \log d))(1/\varepsilon)^{32q}$ *size ϵ-bias space for $\mathbb{Z}_d^n$.*

References

[1] R. Ahlswede and A. Winter. Strong Converse for Identification via Quantum Channels. *IEEE Transactions on Information Theory,* 48(3): 569âĂŞ579, 2003.

[2] Miklós Ajtai, Henryk Iwaniec, János Komlós, János Pintz, and Endre Szemerédi. . Construction of a thin set with small Fourier coefficients. *Bull. London Math. Soc.* 22:583-590, 1990.

[3] Noga Alon and Yuval Roichman. Random Cayley Graphs and Expanders. *Random Struct. Algorithms*, 5(2):271–285, 1994.

[4] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple construction of almost k-wise independent random variables. *Random Struct. Algorithms,* 3(3):289-304, 1992.

[5] Noga Alon, Jehoshua Bruck, Joseph Naor, Moni Naor, and Ron M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory,* 38(2), 1992.

[6] Vikraman Arvind, Partha Mukhopadhyay, and Prajakta Nimbhorkar. Erdős-Rényi Sequences and Deterministic Construction of Expanding Cayley Graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:81, 2011 and *Proc. of LATIN 2012,* Springer.

[7] Vikraman Arvind, Partha Mukhopadhyay, Prajakta Nimbhorkar, and Yadu Vasudev. Expanding generator sets for solvable permutation groups. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:140, 2011 and *Proc of MFCS 2012,* Springer.

[8] Yossi Azar, Rajeev Motwani, and Joseph Naor. Approximating Probability Distributions Using Small Sample Spaces. *Combinatorica,* 18(2):151–171, 1998.

[9] László Babai. Local expansion of vertex-transitive graphs and random generation in finite groups. *Proc. 23rd Annual ACM Symp. on Theory of computing,* 164-174, 1991.

[10] Laszlo Babai, Endre Szemerédi. On the Complexity of Matrix Group Problems I. *Proc. of the 25th IEEE FOCS Conference,* 229-240.

[11] John D. Dixon. The solvable length of a solvable linear group. *Mathematische Zeitschrift,* 107: 151-158, 1968.

[12] Paul Erdős and Alfréd Rényi. Probabilistic methods in group theory. *Journal D'analyse Mathematique*, 14(1):127–138, 1965.

[13] Martin Hildebrand. A survey of results on random random walks on finite groups. *Probability Surveys*, 2:33–63, 2005.

[14] Shlomo Hoory, Nati Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. AMS*, 43(4):439–561, 2006.

[15] Zeph Landau, Alexander Russell. Random Cayley Graphs are Expanders: a Simple Proof of the Alon-Roichman Theorem. *Electr. J. Comb.* 11(1) (2004).

[16] Po-Shen Loh, Leonard J. Schulman: Improved Expansion of Random Cayley Graphs. *Discrete Mathematics & Theoretical Computer Science* 6(2): 523-528, 2004.

[17] Alexander Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica,* 8(3): 261–277, 1988.

[18] Eugene M. Luks, *Permutation groups and polynomial-time computation*, DIMACS series in Discrete Mathematics and Theoretical Computer Science **11** (1993), 139–175.

[19] Ravi Montenegro and Prasad Tetali. Mathematical Aspects of Mixing Times in Markov Chains. *Foundations and Trends in Theoretical Computer Science*, 1(3), 2005.

[20] Igor Pak. Random Cayley Graphs with $O(\log[g])$ Generators Are Expanders. In *Proceedings of the 7th Annual European Symposium on Algorithms*, ESA '99, pages 521–526. Springer-Verlag, 1999.

[21] P. Raghavan. Probabilistic construction of deterministic algorithms: approximating packing integer programs. *Journal of Computer and System Sciences,* 37(2):130Âą143, 1988.

[22] Dana Randall. Rapidly Mixing Markov Chains with Applications in Computer Science and Physics. *Computing in Science and Engg.*, 8(2):30–41, 2006.

[23] Eyal Rozenman and Salil P. Vadhan. Derandomized squaring of graphs. *Proc. APPROX-RANDOM 2005,* LNCS vol. 3624: 436-447, 2005, Springer.

[24] Omer Reingold, Salil Vadhan and Avi Wigderson. Entropy Waves, The Zig-Zag Graph Product, and New Constant-Degree Expanders and Extractors. *Annals of Mathematics,* 155, 157-187, 2002.

[25] Avi Wigderson and David Xiao. Derandomizing the Ahlswede-Winter matrix-valued Chernoff bound using pessimistic estimators, and applications. *Theory of Computing*, 4(1):53–76, 2008.

THE DISTRIBUTED COMPUTING COLUMN

BY

PANAGIOTA FATOUROU

Department of Computer Science, University of Crete
P.O. Box 2208 GR-714 09 Heraklion, Crete, Greece
and
Institute of Computer Science (ICS)
Foundation for Research and Technology (FORTH)
N. Plastira 100. Vassilika Vouton
GR-700 13 Heraklion, Crete, Greece
faturu@csd.uoc.gr

AN INTRODUCTORY TUTORIAL TO CONCURRENCY-RELATED DISTRIBUTED RECURSION

Sergio Rajsbaum
Instituto de Matemarticas, UNAM,
Mexico
rajsbaum@im.unam.mx

Michel Raynal
Institut Universitaire de France,
IRISA, Université de Rennes,
35042 Rennes Cedex,
France
raynal@irisa.fr

Abstract

Recursion is a fundamental concept of sequential computing that allows for the design of simple and elegant algorithms. Recursion is also used in both parallel or distributed computing to operate on data structures, mainly by exploiting data independence (independent data being processed concurrently). This paper is a short introduction to recursive algorithms that compute tasks in asynchronous distributed systems where communication is through atomic read/write registers, and any number of processes can commit crash failures. In such a context and differently from sequential and parallel recursion, the conceptual novelty lies in the fact that the aim of the recursion parameter is to allow each participating process to learn the number of processes that it sees as participating to the task computation.

Keywords: Asynchrony, Atomic read/write register, Branching time, Concurrency, Distributed algorithm, Concurrent object, Linear time, Participating process, Process crash failure, Recursion, Renaming, Shared memory, Task, Write-snapshot.

1 Introduction

Recursion Recursion is a powerful algorithmic technique that consists in solving a problem of some size (where the size of the problem is measured by the number of its input data) by reducing it to problems of smaller size, and proceeding the same way until we arrive at basic problems that can be solved directly. This algorithmic strategy is often capture by the Latin terms "*divide ut imperes*".

Recursive algorithms are often simple and elegant. Moreover, they favor invariant-based reasoning, and their time complexity can be naturally captured by recurrence equations. In a few words, recursion is a fundamental concept addressed in all textbooks devoted to sequential programming (e.g., [10, 14, 19, 25] to cite a few). It is also important to say that, among the strong associations linking data structures and control structures, recursion is particularly well suited to trees and more generally to graph traversal [10].

Recursive algorithms are also used since a long time in parallel programming (e.g., [2]). In this case, parallel recursive algorithms are mainly extensions of sequential recursive algorithms, which exploit data independence. Simple examples of such algorithms are the parallel versions of the quicksort and mergesort sequential algorithms.

Recursion and distributed computing In the domain of distributed computing, the first (to our knowledge) recursive algorithm that has been proposed is the algorithm solving the Byzantine general problem [22]. This algorithm is a message-passing synchronous algorithm. Its formulation is relatively simple and elegant, but it took many years to understand its deep nature (e.g., see [7] and textbooks such as [6, 24, 29]). Recursion has also been used to structure distributed systems to favor their design and satisfy dependability requirements [28].

Similarly to parallelism, recursion has been used in distributed algorithms to exploit data independence or provide time-efficient implementations of data structures. As an example, the distributed implementation of a store-collect object described in [4] uses a recursive algorithm to obtain an efficient tree traversal, which provides an efficient adaptive distributed implementation. As a second example, a recursive synchronous distributed algorithm has been introduced in [5] to solve the lattice agreement problem. This algorithm, which recursively divides a problem of size n into two sub-problems of size $n/2$, is then used to solve the snapshot problem [1]. Let us notice that an early formal treatment of concurrent recursion can be found in [12].

Capture the essence of distributed computing The aim of real-time computing is to ensure that no deadline is missed, while the aim of parallelism is to allow applications to be efficient (crucial issues in parallel computing are related to job partitioning and scheduling). Differently, when considering distributed computing, the main issue lies in mastering the uncertainty created by the multiplicity and the geographical dispersion of computing entities, their asynchrony and the possibility of failures.

At some abstract level and from a "fundamentalist" point of view, such a distributed context is captured by the notion of a task, namely, the definition of a distributed computing unit which capture the essence of distributed computing [17]. Tasks are the distributed counterpart of mathematical functions encountered in sequential computing (where some of them are computable while others are not).

At the task level, recursion is interesting and useful mainly for the following reasons: it simplifies algorithm design, makes their proofs easier, and facilitates their analyze (thanks to topology [13, 26]).

Content of the paper: recursive algorithms for computable tasks This paper is on the design of recursive algorithms that compute tasks [13]. It appears that, for each process participating to a task, the recursion parameter x is not related to the size of a data structure but to the number of processes that the invoking process perceives as participating to the task computation. In a very interesting way, it follows from this feature that it is possible to design a general pattern, which can be appropriately instantiated for particular tasks.

When designing such a pattern, the main technical difficulty come from the fact that processes may run concurrently, and, at any time, distinct processes can be executing at the same recursion level or at different recursion levels. To cope with such an issue, recursion relies on an underlying data structure (basically, an array of atomic read/write registers) which keeps the current state of each recursion level.

After having introduced the general recursion pattern, the paper instantiates it to solve two tasks, namely, the write-snapshot task [8] and the renaming task [3]. Interestingly, the first instantiation of the pattern is based on a notion of linear time (there is single sequence of recursive calls, and each participating process executes a prefix of it), while the second instantiation is based on a notion of branching time (a process executes a prefix of a single branch of the recursion tree whose branches individually capture all possible execution paths).

In addition to its methodological dimension related to the new use of recursion in a distributed setting, the paper has a pedagogical flavor in the sense that

it focuses on and explains fundamental notions of distributed computing. Said differently, an aim of this paper is to provide the reader with a better view of the nature of fault-tolerant distributed recursion when the processes are concurrent, asynchronous, communicate through read/write registers, and are prone to crash failures.

Road map The paper is made up of 6 sections. Section 2 presents the computation model and the notion of a task. Then, Section 3 introduces the basic recursive pattern in which the recursion parameter of a process represents its current approximation of the number of processes it sees as participating. The next two sections present instantiations of the pattern that solve the write-snapshot task (Section 4) and the renaming task (Section 5), respectively. Finally, Section 6 concludes the paper. (While this paper adopts a programming methodology perspective, the interested reader will find in [26] a topological perspective of recursion in distributed computing).

2 Computation Model, Notion of a Task, and Examples of Tasks

2.1 Computation model

Process model The computing model consists of n processes denoted p_1, ..., p_n. A process is a deterministic state machine. The integer i is called the index of p_i. The indexes can only be used for addressing purposes. Each process p_i has a name –or identity– id_i. Initially a process p_i knows only id_i, n, and the fact that no two processes have the same initial name. Moreover, process names belong to a totally ordered set and this is known by the processes (hence two identities can be compared).

The processes are asynchronous in the sense that the relative execution speed of different processes is arbitrary and can varies with time, and there is no bound on the time it takes for a process to execute a step.

Communication model and local memory The processes communicate by accessing atomic read/write registers. Atomic means that, from an external observer point of view, each read or write operation appears as if it has been executed at a single point of the time line between it start and end events [18, 20].

Each atomic register is a single-writer/multi-reader (SWMR) register. This means that, given any register, a single process (statically determined) can write in this register, while all the processes can read it. Let $X[1..n]$ be an array of atomic registers whose entries are the process indexes. By convention, $X[i]$ can be written only by p_i. Atomic registers are denoted with uppercase letters. All shared registers are initialized to a default value denoted $\perp$ and no process can write $\perp$ in a register. Hence, the meaning of $\perp$ is to state that the corresponding register has not yet been written.

A process can have local variables. Those are denoted with lowercase letters and sub-scripted by the index of the corresponding process. As an example, aaa_i denotes the local variable *aaa* of process p_i.

Failure model The atomic read/write registers are assumed to experience no failure. (For the interested reader, the construction of atomic reliable registers from basic atomic registers which can fail –crash, omission, or Byzantine failures- is addressed in [30]).

A process may crash (halt prematurely). A process executes correctly until it possibly crashes, and after it has crashed (if ever it does), it executes no step. Given a run, a process that crashes is *faulty*, otherwise it is *non-faulty*.

Any number of processes may crash (*wait-free* model [15]). Let us observe that the wait-free model prevents implicitly the use of locks (this is because a process that owns a lock and crashes before releasing it can block the whole system). (Locks can be implemented from atomic read/write registers only in reliable systems [30].)

2.2 The notion of a task

Informal definition As indicated in the Introduction, a task is the distributed counterpart of a mathematical function encountered in sequential computing.

In a task each process p_i has a private input value in_i and, given a run, the n input values constitute the input vector I of the considered run. each process knows initially only its input value, which is usually called *proposed value*. Then, from an operational point of view, the processes have to coordinate and communicate in such a way that each process p_i computes an output value out_i and the n output values define an output vector O, such that $O \in \Delta(I)$ where Δ is the mapping defining the task. An output value is also called *decided* value. The way a distributed task extends the notion of a sequential function is described in Figure 1, where the left side represents a classical a sequential function and the right side represents a distributed task.

As in sequential computing (Turing machines) where there are computable functions and uncomputable functions, there are computable tasks and uncomputable tasks. As we will see later write-snapshot and renaming are computable in asynchronous read/write systems despite asynchrony and any number of process failures, while consensus is not [11, 15, 23].

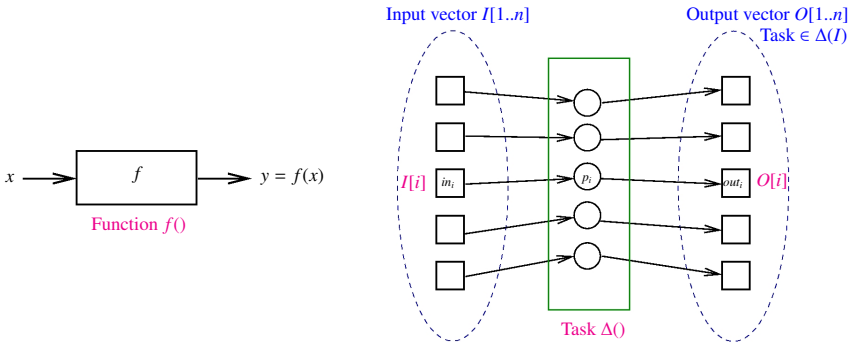


Figure 1: Function (left) and task (right)

Formal definition A task is a triple $\langle \mathcal{I}, \mathcal{O}, \Delta \rangle$ where

- $\mathcal{I}$ is the set of allowed input vectors,
- $\mathcal{O}$ is the set of allowed output vectors, and
- Δ is a mapping of $\mathcal{I}$ into $\mathcal{O}$ such that $(\forall I \in \mathcal{I}) \Rightarrow (\Delta(I) \in \mathcal{O})$.

Hence, $I[i]$ and $O[i]$ are the values proposed and decided by p_i, respectively, while $\Delta(I)$ defines the set of output vectors that can be decided from the input vector I. (More developments on the definition of tasks and their relation with topology can be found in [16, 17]).

If one or several processes p_i, ..., p_j, do not participate or crash before deciding an output value, we have $O[i] = \ldots = O[j] = \perp$, and the vector O has then to be such that there is a vector $O' \in \Delta(I)$ that covers O, i.e., $(O[i] \neq \perp) \Rightarrow (O'[i] = O[i])$.

A simple example: the binary consensus task In this task, a process proposes a value from the set $\{0, 1\}$, and all the non-faulty processes have to decide the same value which has to be a proposed value. Let X_0 and X_1 be the vector of size n containing only zeros and only ones, respectively.

The set $\mathcal{I}$ of input vectors is the set of all the vectors of zeros and ones. The set $\mathcal{O}$ of output vectors is $\{X_0, X_1\}$. The mapping Δ is such that (i) $\Delta($ any vector except $X_0, X_1) = \mathcal{O}$, (ii) $\Delta(X_0) = X_0$, and (iii) $\Delta(X_1) = X_1$.

Solving a task In the context of this paper, a distributed algorithm $\mathcal{A}$ is a set of n local automata (one per process) that communicate through atomic read/write registers.

The algorithm $\mathcal{A}$ solve a task T if, in any run in which each process proposes a value such that the input vector belongs to $\mathcal{I}$, each non-faulty process decides a value, and the vector O of output values belongs to the set $\Delta(I)$.

Tasks vs Objects A task is a mathematical object. From a programming point of view, a concurrent object can be associated with a task (a concurrent object is an object that can be accessed by several processes). Such an object is a one-shot object that provides the processes with a single operation ("one-shot" means that a process can invoke the object operation at most once).

To adopt a more intuitive presentation, the two tasks that are presented below use their object formulation. This formulation expresses the mapping Δ defining a task by a set of properties that the operation invocations have to satisfy. These properties can be more restrictive than Δ. This comes from the fact that there is no notion of time/concurrency/communication pattern in Δ, while the set of properties defining the object can implicitly refer to such notions.

2.3 The write-snapshot task

The write-snapshot task was introduced in [8] (where it is called *immediate snapshot*). A write-snapshot object provides processes with a single operation denoted write_snapshot(). When a process p_i invokes this operation, it supplies as input

parameters its identity id_i and the value it wants to deposit into the write-snapshot object. Its invocation returns a set $view_i$ composed of pairs (id_j, v_j).

As previously indicated, the specification Δ is expressed here a set of properties that the invocations of write_snapshot() have to satisfy.

- Self-inclusion. $\forall\ i$: $(id_i, v_i) \in view_i$.
- Containment. $\forall\ i, j$: $(view_i \subseteq view_j)\ \vee\ (view_j \subseteq view_i)$.
- Simultaneity.
  $\forall\ i, j$: $[((id_j, v_j) \in view_i)\ \wedge\ ((id_i, v_i) \in view_j)]\ \Rightarrow\ (view_i = view_j)$.
- Termination.
  Any invocation of write_snapshot() by a non-faulty process terminates.

A write-snapshot combines in a single operation the write of a value (here a pair (id_i, v_i)) and a snapshot [1] of the set of pairs already or concurrently written. Self-inclusion states that a process sees its write. Containment states the views of the pairs deposited are ordered by containment. Simultaneity states that if each of two processes sees the pair deposited by the other one, they have the same view of the deposited pairs. Finally, the termination property states that the progress condition associated with operation invocations is wait-freedom, which means that an invocation by a non-faulty process terminates whatever the behavior of the other processes (which can be slow, crashed, or not participating). An iterative implementation of write-snapshot can be found in [8, 30].

2.4 The adaptive renaming task

This task has been introduced in [3] in the context of asynchronous crash-prone message-passing systems. Thereafter, a lot of renaming algorithms suited to read/write communication have been proposed. An introduction to shared memory renaming, and associated lower bounds, is presented in [9].

While there are only n process identities, the space name is usually much bigger than n (as a simple example this occurs when the name of a machine is the IP address). The aim of the adaptive renaming task is to allow the processes to obtain new names from a new name space which has to depend only on the number p of processes that want to obtain a new name ($1 \leq p \leq n$), and be as small as possible. It is shown in [17] that $2p - 1$ is a lower bound on the size of the new name space.

When considering the adaptive renaming task from the point of view of its associated one-shot object, a process p_i that wants to acquire a new name invokes an operation denoted new_name(id_i). The set of invocations has to satisfy the following set of properties.

- Validity. The size of the new name space is $2p - 1$.
- Agreement. No two processes obtain the same new name.
- Termination.
  Any invocation of new_name() by a non-faulty process terminates.

As for the write-snapshot task, the termination property states that a non-faulty process that invokes the operation new_name() obtains a new name whatever the behavior of the other processes. Agreement states the consistency condition associated with new names. Validity states the domain of the new names: if a

single process wants to obtain a new name, it obtains the name 1, if only two processes invoke new_name() they obtain new names in the set $\{1, 2, 3\}$, etc. This show that the termination property (wait-freedom progress condition) has a cost in the size of the new name space: while only p new names are needed, the new name space needs $(p - 1)$ additional potential new names to allow the invocations issued by non-faulty processes to always terminate.

3 A Concurrency-related Recursive Pattern for Distributed Algorithms

The recursion parameter As already announced, the recursion parameter (denoted x) in the algorithms solving the tasks we are interested is the number of processes that the invoking process perceives as participating processes. As initially a process has no knowledge of how many processes are participating, it conservatively considers that all other processes participate, and consequently issues a main call wit $x = n$.

Atomic read/write registers and local variables The pattern manages an array $SM[n..1]$, where each $SM[x]$ is a sub-array of size n such that $SM[x][i]$ can be written only by p_i. A process p_i starts executing the recursion level x by depositing a value in $SM[x][i]$. »From then on, it is a participating process at level x.

Each process manages locally three variables whose scope is a recursive invocation. $sm_i[n..1]$ is used to save a copy of the current value of $SM[x][1..n]$; $part_i$ keeps the number of processes that p_i sees as participating at level x; and res_i is used to save the result returned by the current invocation.

```
operation recursive_pattern(x, input) is
(01)  SM[x][i] ← input;
(02)  for each j ∈ {1, ..., n} do sm_i[j] ← SM[x][j] end for;
(03)  part_i ← |{sm_i[j] ≠ ⊥}|;
(04)  if (part_i = x) then statements specific to the task, possibly including a recursive call;
(05)                      computation of res_i
(06)                 else res_i ← recursive_pattern(x − 1, input)
(07)  end if
(08)  return(res_i)
end operation.
```

Figure 2: Concurrency-related recursive pattern

The recursion pattern The generic recursive pattern is described in Figure 2. The invoking process p_i first deposits its input parameter value in $SM[x][i]$ (line 1), and read the content of the shared memory attached to its recursion level x (line 2). Let us notice that the entries of the array $SM[x][1..n]$ are read in any order and asynchronously. Then, p_i computes the number of processes it sees as participating in the recursion level x (line 3), and checks if this number is equal to its current recursion level x.

- if $x = part_i$ (lines 4-5), p_i discovers that x processes are involved in the recursion level x. In this case, it executes statements at the end of which it computes a local result res_i. These local statements are task-dependent and may or not involve a recursive call with recursion level $x - 1$.
- if $x \neq part_i$, p_i sees less than x processes participating to the recursion level x. In this case, it invokes the recursion pattern at level $x - 1$ with the same input parameter $input$, and continues until it attains a recursion level $x' \leq x - 1$ at which it sees exactly x' processes that have attained this recursion level x'.

A process p_i starts with its recursion parameter x equal n, and then its recursion parameter decreases until the invoking process returns a result. Hence, a process executes at most n recursive calls before terminating. The correctness proof of this recursive pattern is the same as the one of Theorem 1 which considers its write-snapshot instantiation.

Linear time vs branching time If line 4 does not include a recursive call, the recursive pattern is a linear time pattern. Each participating process executes line 6 until its stops at line 4 (or crashes before). Hence, each process executes a prefix of the same sequence of recursive calls, each with its initial input parameter $input$. The algorithm, whose instantiation from the recursive pattern is described in Section 4, is a linear time implementation of write-snapshot.

If there are recursive calls at line 4, the recursive pattern is a branching time pattern. Such a recursion pattern is characterized by a tree of recursive calls, and a participating process executes a prefix of a single branch of this tree. In this case, each $SM[x]$ is composed of several sub-arrays, each of them being an array of n SWMR atomic registers. The algorithm, whose instantiation from the recursive pattern is described in Section 5, is a branching time implementation of renaming.

4 Linear Time Recursion

4.1 A recursive write-snapshot algorithm

An instantiation of the recursive pattern which implements write-snapshot is described in Figure 3. This recursive implementation has been introduced in [13], and the representation adopted here is from [30]. This instantiation is nearly the same as the original recursive pattern. More precisely, the input parameter $input$ of a process p_i is the pair (id_i, v_i).

The line numbering is the same as in the recursive pattern. As there is no specific statement to instantiate at line 4 of the recursive pattern, its lines 4 and 5 are instantiated by a single line denoted 4+5.

A process p_i invokes first write_snapshot$(n, (id_i, v_i))$ where v_i is the value it wants to deposit in the write-snapshot object.

As already said, the recursion of this algorithm is a linear time recursion. This appears clearly from the arrays of atomic read/write registers accessed by the recursive calls issued by the processes: each process accesses first $SM[n]$, then $SM[n-1]$, etc., until it stops at $SM[x]$ where $n \geq x \geq 1$.

```
operation write_snapshot(x, (id_i, v_i)) is
(1)     SM[x][i] ← (id_i, v_i);
(2)     for each j ∈ {1, ..., n} do sm_i[j] ← SM[x][j] end for;
(3)     part_i ← |{sm_i[j] ≠ ⊥}|;
(4+5)   if (part_i = x) then res_i ← {sm_i[j] ≠ ⊥}
(6)                     else res_i ← write_snapshot(x − 1, (id_i, v_i))
(7)     end if
(8)     return(res_i)
end operation.
```

Figure 3: A recursive write-snapshot algorithm [13]

4.2 Proof of the algorithm

Theorem 1. [13] *The algorithm described in Figure 3 implements a write-snapshot object. For a process p_i, The step complexity (number of shared memory accesses) for a process p_i is $O(n(n - |res_i| + 1))$, where res_i is the set returned by the invocation of* write_snapshot() *issued by p_i.*

Proof This proof is from [30]. While a process terminates an invocation when it executes the return() statement at line 8, we say that it terminates at line 4+5 or line 6, according to the line where the returned value res_i has been computed.

Claim C. If at most x processes invoke write_snapshot$(x, -)$, (a) at most $(x-1)$ processes invoke write_snapshot$(x-1, -)$, and (b) at least one process stops at line 4+5 of its invocation of write_snapshot$(x, -)$.

Proof of claim C. Assuming that at most x processes invoke write_snapshot$(x, -)$, let p_k be the last process that writes into $SM[x][1..n]$ (as the registers are atomic, the notion of "last" is well-defined). We necessarily have $part_k \leq x$. If p_k finds $part_k = x$, it stops at line 4+5. Otherwise, we have $part_k < x$ and p_k invokes write_snapshot$(x-1, -)$ at line 6. But in this case, as p_k is the last process that wrote into the array $SM[x][1..n]$, it follows from $part_k < x$ that fewer than x processes have written into $SM[x][1..n]$, and consequently, at most $(x-1)$ processes invoke write_snapshot$(x-1, -)$. End of the proof of claim C.

To prove termination, let us consider a non-faulty process p_i that invokes write_snapshot$(n, -)$. It follows from Claim C and the fact that at most n processes invoke write_snapshot$(n, -)$ that either p_i stops at that invocation or belongs to the set of at most $(n-1)$ processes that invoke write_snapshot$(n-1, -)$. It then follows, by induction from the claim C, that if p_i has not stopped during a previous invocation, it is the only process that invokes write_snapshot$(1, -)$. It then follows from the text of the algorithm that it stops at that invocation.

The proof of the self-inclusion property is trivial. Before stopping at recursion level x (line 4+5), a process p_i has written v_i into $SM[x][i]$ (line 1), and consequently we have then $(id_i, v_i) \in view_i$, which concludes the proof of the self-inclusion property.

To prove the self-containment and simultaneity properties, let us first consider the case of two processes that return at the same recursion level x. If a process p_i returns at line 4+5 of recursion level x, let $res_i[x]$ denote the corresponding value of res_i. Among the processes that stop at recursion level x, let p_j be the last process which writes into $SM[x][1..n]$. As p_i stops, this means that $SM[x][1..n]$ has exactly

x entries different from $\perp$ and (due to Claim C) no more of its entries will be set to a non-$\perp$ value. It follows that, as any other process p_j that stops at recursion level x reads x non-$\perp$ entries from $SM[x][1..n]$, we have $res_i[x] = res_j[x]$ which proves the properties.

Let us now consider the case of two processes p_i and p_j that return at line 6 of recursion level x and y, respectively, with $x > y$ (i.e., p_i returns $res_i[x]$ while p_j returns $res_j[y]$). The self-containment follows then from $x > y$ and the fact that p_j has written into all the arrays $SM[z][1..n]$ with $n \geq z \geq y$, from which we conclude that $res_j[y] \subseteq res_i[x]$. Moreover, as $x > y$, p_i has not written into $SM[y][1..n]$ while p_j has written into $SM[x][1..n]$, and consequently $(id_j, v_j) \in res_i[x]$ while $(id_i, v_i) \notin res_j[y]$, from which both he containment and immediacy properties follow.

As far as the number of shared memory accesses is concerned we have the following. Let *res* be the set returned by an invocation of write_snapshot$(n, -)$. Each recursive invocation costs $n+1$ shared memory accesses (lines 1 and 2). Moreover, the sequence of invocations, namely write_snapshot$(n, -)$, write_snapshot$(n - 1, -)$, etc., until write_snapshot$(|res|, -)$ (where $x = |res|$ is the recursion level at which the recursion stops) contains $n - |res| + 1$ invocations. It follows that the step complexity for a process p_i is $O(n(n - |res_i| + 1))$ accesses to atomic registers. □*Theorem* 1

4.3 Example of an execution

This section described simple executions where $n = 5$ and process p_5 crashes before taking any step (or –equivalently– does not participate). These executions are described in Table 1 and Table 2. In these tables write_snapshot() is abbreviated as ws().

	p_1	p_2	p_3	p_4	p_5
τ_1			ws$(5, (id_3, v_3))$		
τ_2			ws$(4, (id_3, v_3))$		
τ_3			crashes		
τ_4				ws$(5, (id_4, v_4))$	
τ_5				... ws$(1, (id_4, v_4))$	
τ_6				$\{(id_4, v_4)\}$	
τ_7	ws$(5, (id_1, v_1))$	ws$(5, (id_2, v_2))$			
τ_8	ws$(4, (id_1, v_1))$	ws$(4, (id_2, v_2))$			
τ_9	res_1	res_2			

Table 1: Write-snapshot execution: an example

A first execution

1. At time τ_1, p_3 invokes write_snapshot$(5, (id_3, v_3))$. This triggers at time τ_2 the recursive invocation write_snapshot$(4, (id_3, v_3))$. Then, p_3 crashes after it has written id_3 into $SM[4][3]$ at time τ_3.
2. At a later time τ_4, p_4 invokes write_snapshot$(5, (id_4, v_4))$, which recursively ends up with the invocation write_snapshot$(1, (id_4, v_4))$ at time τ_5, and consequently p_4 returns the singleton set $\{id_4, v_4)\}$ at time τ_6.
3. At time τ_7, processes p_1 and p_4 start executing synchronously: p_1 invokes write_snapshot$(5, (id_1, v_1))$, while p_2 invokes write_snapshot$(5, (id_2, v_2))$, which entails at time τ_8 –always synchronously– the recursive invocations write_snapshot$(4, (id_1, v_1))$ and write_snapshot$(4, (id_2, v_2))$. As $SM[4]$ contains four non-$\perp$ entries, both p_1 and p_2 returns res_1 and res_2 which are such that $res_1 = res_2 = \{(id_1, v_1), (id_2, v_2), (id_3, v_3), (id_3, v_4)\}$.

	p_1	p_2	p_3	p_4	p_5
τ_{10}			ws$(3, (id_3, v_3))$		
τ_{11}			ws$(2, (id_3, v_3))$		
τ_{12}			res_3		

Table 2: Write-snapshot execution: continuing the example

Continuing the example Let us assume that instead of crashing at time τ_3, p_3 paused for an arbitrary long period starting after it has read $SM[4][1..5]$ (hence it has seen only two non-$\perp$ values in $SM[4]$).

1. At time τ_{10}, p_3 wakes up and, as $part_3 \neq 4$, it it issues the recursive invocation write_snapshot$(3, (id_3, v_3))$, which entails at time τ_{11} the invocation write_snapshot$(2, (id_3, v_3))$.
2. As at time τ_{12}, the shared array $SM[2]$ contains two non-$\perp$ values, process p_4 returns $res_3 = \{(id_3, v_3), (id_3, v_4)\}$.

The reader can check that, if before pausing at time τ_3, p_3 has read only $SM[4][4]$ and $SM[4][5]$, it will read the other entries $SM[4][1]$, $SM[4][2]$, and $SM[4][3]$, when it wakes up, and its invocation write_snapshot$(4, (id_3, v_3))$ will stop the recursion and return $res_3 = res_1 = res_2$.

5 Branching Time Recursion

5.1 A recursive renaming algorithm

An instance of the recursive pattern implementing adaptive renaming is described in Figure 4. This recursive implementation, inspired from the sketch of an algorithm skeleton succinctly described in [13], has been introduced in [27], where it is proved correct. As for the previous recursive algorithm, the representation adopted here is from [30]. The core of this recursive algorithm is the instantiation of line 4 of the recursive pattern, where appears branching time recursion.

Underlying idea: the case of two processes The base case is when $n = 2$. A process p_i first writes its identity id_i in the shared memory, and then reads the content of the memory.

- If, according to what it has read from the shared memory, a process sees only itself, it adopt the new name 1.
- Otherwise it knows its identity and the one of the other process (id_j). It then compares its identity id_i and id_j, and does the following: if $id_i > id_j$, it adopts the new name 3, if $id_i < id_j$, it adopts the new name 2.

The new name space is consequently $[1..2p-1]$ where p (number of participating processes) is 1 or 2.

The underlying shared memory The shared memory $SM[n..1]$ accessed by processes is now a three-dimensional array $SM[n..1, 1..2n-1, \{up, down\}]$ such that $SM[x, first, dir]$ is a an array of n atomic read/write registers. $SM[x, first, dir][i]$ can be written only by p_i but can be read by all processes.

From a notational point of view $up = 1 = \overline{down}$, and $down = -1 = \overline{up}$.

When more than two processes participate The algorithm is described in Figure 4. A process invokes first new_name$(n, 1, up, id_i)$. It then recursively invokes new_name$(x, 1, up, id_i)$, until the recursion level x is equal to the number of processes that p_i sees as competing for a new name.

As we are about to see, given a pair $(first, dir)$, the algorithm ensures that at most x processes invoke new_name$(x, first, dir, -)$. These processes compete for new names in a space name of size $2x-1$ which is the interval $[first..first+(2x-2)]$ if $dir = up$, and $[first-(2x-2)..first]$ if $dir = down$. Hence, the value up is used to indicate that the concerned processes are renaming "from left to right" (as far as the new names are concerned), while $down$ is used to indicate that the concerned processes are renaming "from right to left" (this is developed below when explaining the splitter behavior of the underlying read/write registers.) Hence, a process p_i considers initially the renaming space $[1..2n-1]$, and then (as far p_i is concerned) this space will shrink at each recursive invocation (going up or going down) until p_i obtains a new name.

The recursive algorithm The lines 1-3 and 6-8 are the same as in the recursive pattern where $SM[x]$ is replaced by $SM[x, first, dir]$. The lines which are specific to adaptive renaming are the statements in the **then** part of the recursive pattern (lines 4-5). These statements are instantiated by the new lines (4+5).1-(4+5).5, which constitute an appropriate instantiation suited adaptive renaming.

For each triple (x, f, d), all invocations new_name$(-, x, f, d)$ coordinate their respective behavior with the help of the size n array of atomic read/write registers $SM[x, f, d][1..n]$. At line (4+5).", $\max(sm_i)$ denotes the greatest process identity present in sm_i. As a process p_i deposits its identity in $SM[x, first, dir][i]$ before reading $SM[x, first, dir][1..n]$, it follows that sm_i contains at least one process identity when read by p_i.

Let us observe that, if only p processes invoke new_name$(n, 1, up, -)$, $p < n$, then all of them will invoke the algorithm recursively, first with new_name$(n-1, 1, up)$, then new_name$(n-2, 1, up)$, etc., until new_name$(p, 1, up, -)$. Only

```
operation new_name(x, first, dir, id_i) is
(1)      SM[x, first, dir][i] ← id_i;
(2)      for each j ∈ {1, ..., n} do sm_i[j] ← SM[x, first, dir][j] end for;
(3)      part_i ← |{sm_i[j] ≠ ⊥}|;
(4+5).1  if (part_i = x) then last ← first = dir(2x − 2);
(4+5).2                       if (id_i = max(sm_i)
(4+5).3                         then res_i ← last
(4+5).4                         else res_i ← new_name(x − 1, last + dir‾, dir‾, id_i))
(4+5).5                       end if
(6)                    else res_i ← new_name(x − 1, first, dir, id_i))
(7)      end if
(8)      return(res_i)
end operation.
```

Figure 4: A recursive adaptive renaming algorithm [13]

at this point, the behavior of a participating process p_i depend on the concurrency pattern (namely, it may or may not invoke the algorithm recursively, and with either up or $down$).

Splitter behavior associated with $SM[x, first, dir]$ (The notion of a splitter has been informally introduced in [21].) Considering the (at most) x processes that invoke new_name$(x, first, dir, -)$, the splitter behavior associated with the array of atomic registers $SM[x, first, dir]$ is defined by the following properties. Let $x' = x - 1$.

- At most $x' = x - 1$ processes invoke new_name$(x - 1, first, dir, -)$ (line 6. Hence, these processes will obtain new names in an interval of size $(2x' - 1)$ as follows:
  - If $dir = up$, the new names will be in the "going up" interval $[first..first + (2x' - 2)]$,
  - If $dir = down$, the new names will be in the "going down" interval $[first - (2x' - 2)..first]$.

- At most $x' = x - 1$ processes invoke new_name$(x - 1, last + \overline{dir}, \overline{dir})$ (line (4 5).4), where $last = first + dir(2x - 2)$ (line (4 5).1). Hence, these $x' = x - 1$ processes will obtain their new names in a renaming space of size $(2x' - 1)$ starting at $last + 1$ and going from left to right if $\overline{dir} = up$, or starting at $last - 1$ and going from right to left if $\overline{dir} = down$. Let us observe that the value $last \pm 1$ is considered as the starting name because the slot $last$ is reserved for the new name of the process (if any) that stops during its invocation of new_name$(x, first, dir)$ (see next item).
- At most one process "stops", i.e., defines its new name as $last = first + dir(2x - 2)$ (lines (4 5).2 and (4 5.3). Let us observe that the only process p_k that can stop is the one such that id_k has the greatest value in the array $SM[x, first, dir][1..n]$ which contains then exactly x identities.

5.2 Example of an execution

A proof of the previous algorithm can be found in [30]. This section presents an example of an execution of this algorithm. It considers four processes p_1, p_2, p_3, and p_4.

First: process p_3 executes alone Process p_3 invokes new_name$(4, 1, up, id_1)$ while (for the moment) no other process invokes the renaming operation. It follows from the algorithm that p_3 invokes recursively new_name$(3, 1, up, id_1)$, then new_name$(2, 1, up, id_1)$, and finally new_name$(1, 1, up, id_1)$. During the last invocation, it obtains the new name 1. This is illustrated in Figure 5. As, during its execution, p_3 sees only $p = 1$ process (namely, itself), it decides consistently in the new name space $[1..2p - 1] = 1$.

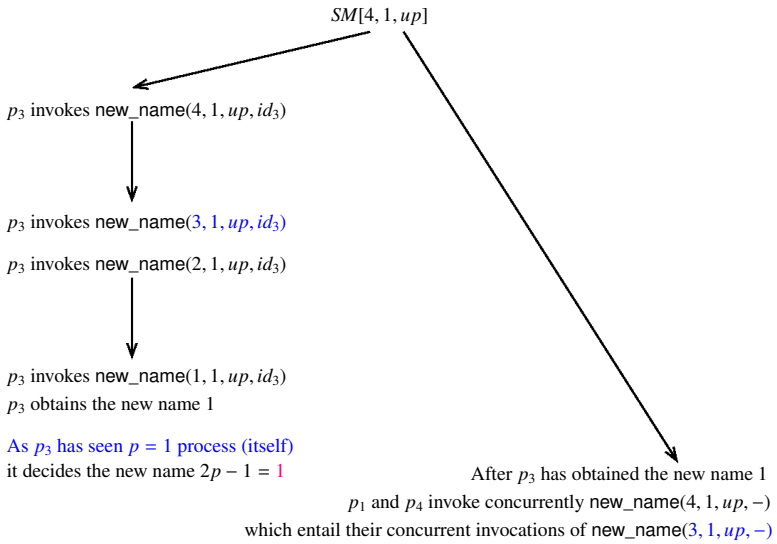


Figure 5: Recursive renaming: first, p_3 executes alone

Then: processes p_1 and p_4 invoke new_name() After p_3 has obtained a new name, both p_1 and p_4 invoke new_name$(4, 1, up, -)$ (See Figure 6). As they see only three processes that have written their identities into $SM[4, 1, up]$, both concurrently invoke new_name$(3, 1, up, -)$ and consequently both compute $last = 1 + (2 * 3 - 2) = 5$. Hence their new name space is $[1..5]$.

Now, let us assume that p_1 stops executing while p_4 executes alone. Moreover, let $id_1, id_4 < id_3$. As it has not the greatest identity among the processes that have accessed $SM[3, 1, up]$ (namely, the processes p_1, p_3 and p_4), p_4 invokes first new_name$(2, 4, down, id_4)$ and then recursively new_name$(1, 4, down, id_4)$, and finally obtains the new name 4.

After process p_4 has obtained its new name, p_1 continues its execution, invokes new_name$(2, 4, down, id_1)$ and computes $last = 4-(2\times 2-2) = 2$. The behavior of p_1 depends then on the values of id_1 and id_4. If $id_4 < id_1$, p_1 decides the name $last = 4-(2\times 2-2) = 2$. If $id_4 > id_1$, p_1 invokes new_name$(1, 3, 1, id_1)$ and finally decides the name 3.

Finally, if later p_2 invokes new_name$(4, 1, up, id_2)$, it sees that the splitter $SM[4, 1, up]$ was accessed by four processes. Hence p_2 computes $last = 1 + (2\times 4-2) = 1$, and consequently invokes recursively new_name$(3, 6, down, id_1)$, new_name$(2, 6, down, id_1)$, new_name$(1, 6, down, id_1)$, at the end of which it computes $last == 6 + (2\times 1-2) = -$ and decides the name 6.

The multiplicity of branching times appears clearly on this example. As an example, the branch of time experienced by p_3 (which is represented by the sequence of accesses to $SM[4, 1, up]$, $SM[3, 1, up]$, $SM[2, 1, up]$, and $SM[1, 1, up]$), is different from the branch of time experienced by p_4 (which is represented by the sequence of accesses to $SM[4, 1, up]$, $SM[3, 1, up]$, $SM[2, 4, down]$, and $SM[1, 4, up]$).

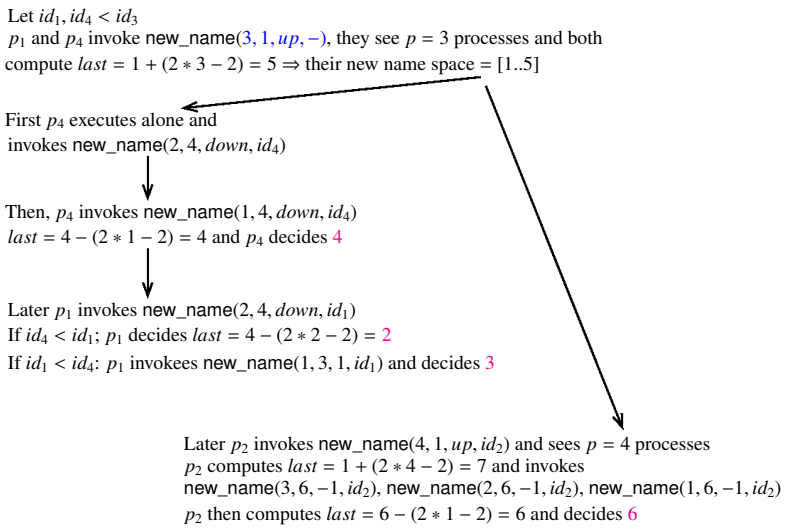


Figure 6: Recursive renaming: p_1 and p_4 invoke new_name$(4, 1, up, -)$

Let us observe that the new name space attributed to the $p = 3$ processes p_1, p_3, and p_4 (the only ones that, up to now, have invoked new_name$(4, 1, up)()$) is $[1..2p-1] = [1..5]$.

Finally process p_2 invokes new_name() Let us now assume that p_2 invokes new_name$(4, 1, up, id_2)$. Moreover, let $id_2 < id-1, id_2, id_3$. Process p_2 sees that $p = 4$ processes have accessed the splitter $SM[4, 1, up]$, and consequently computes $last = 1+(2\times 4-2) = 7$. The size of its new name space is $[1..2p-1] = [1..7]$. As it does not have the greatest initial name among the four processes, p_2

invokes new_name$(3, 6, down, id_2)$, and recursively new_name$(2, 6, down)$ and new_name$(1, 6, down, id - 2)$, and finally obtains 6 as its new name.

6 Conclusion

The aim of this paper is to be an introductory tutorial on concurrency-related recursion in asynchronous read/write systems where any number of processes may crash. The paper has shown that a new type of recursion is introduced by the net effect of asynchrony and failures, namely the recursion parameter is used to allow a process to learn the number of processes with which it has to coordinate to compute its local result. This recursion has been illustrated with two task examples, write-snapshot and adaptive renaming. Interestingly, the first example is related to a linear time notion, while the second one is related to a branching time notion.

Acknowledgments

A special acknowledgment to E. Gafni, whose seminal work on recursive distributed renaming contributed to this presentation.

References

[1] Afek Y., Attiya H., Dolev D., Gafni E., Merritt M. and Shavit N., Atomic snapshots of shared memory. *Journal of the ACM*, 40(4):873-890, 1993.

[2] Akl S.G., *The design and analysis of parallel algorithms*. Prentice-Hall Int'l Series, 401 pages, 1989.

[3] Attiya H., Bar-Noy A., Dolev D., Peleg D. and Reischuk R., Renaming in an asynchronous environment. *Journal of the ACM*, 37(3):524-548, 1990.

[4] Attiya H., Fouren A., and Gafni E., An adaptive collect algorithm with applications. *Distributed Computing*, 15(2): 87-96, 2002.

[5] Attiya H., Herlihy M. and Rachman O., Atomic snapshots using lattice agreement. *Distributed Computing*, 8(3):121-132, 1995.

[6] Attiya H. and Welch J.L., *Distributed computing: fundamentals, simulations and advanced topics*, (2d Edition), Wiley-Interscience, 414 pages, 2004 (ISBN 0-471-45324-2).

[7] Bar-Noy A., Dolev D., Dwork C. and Strong R., Shifting gears: changing algorithms on the fly to expedite Byzantine agreement. *Information and Computation*, 97(2):205-233, 1992.

[8] Borowsky E. and Gafni E., Immediate atomic snapshots and fast renaming. *Proc. 12th ACM Symposium on Principles of Distributed Computing (PODC'93)*, pp. 41-51, 1993.

[9] Castañeda, Rajsbaum S., and Raynal M., The renaming problem in shared memory systems: An introduction. *Computer Science Review*, 5(3):229-251, 2011.

[10] Dahl O.J., Dijkstra E.W., and Hoare C.A.R., *Structured programming*. Academic Press, 220 pages, 1972 (ISBN 0-12-200550-3).

[11] Fischer M.J., Lynch N.A., and Paterson M.S., Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374-382, 1985.

[12] Francez N., Hailpern B., and Taubendfeld G., Script: a communication abstraction mechanism and its verification. *Science of Computer Programming*, 6:35-88, 1986.

[13] Gafni E. and Rajsbaum S., Recursion in distributed computing. *Proc. 12th Int'l l Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS '10)*, Springer LNCS 6366, pp. 362-376, 2010.

[14] Harel D. and Feldman Y., *Algorithmics: the spirit of computing* (third edition). Springer, 572 pages, 2012 (ISBN 978-3-642-27265-3).

[15] Herlihy M.P., Wait-free synchronization. *ACM Transactions on Programming Languages and Systems*, 13(1):124-149, 1991.

[16] Herlihy M.P., Rajsbaum S., and Raynal M., Power and limits of distributed computing shared memory models. To appear *Theoretical Computer Science*, (http://dx.doi.org/10.1016/j.tcs.2013.03.002), 2013.

[17] Herlihy M.P. and Shavit N., The topological structure of asynchronous computability. *Journal ACM*, 46(6):858-923, 1999.

[18] Herlihy M.P. and Wing J.M, Linearizability: a correctness condition for concurrent objects. *ACM Transactions on Programming Languages and Systems*, 12(3):463-492, 1990.

[19] Horowitz E. and Shani S., *Fundamentals of computer algorithms*. Pitman, 626 pages, 1978 (ISBN 0-273-01324-0).

[20] Lamport. L., On Interprocess Communication, Part 1: Basic formalism, Part II: Algorithms. *Distributed Computing*, 1(2):77-101,1986.

[21] Lamport L., Fast mutual exclusion. *ACM Transactions on Computer Systems*, 5(1):1-11, 1987.

[22] Lamport L., Shostak E., and Pease M.C., The Byzantine general problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382-401, 1982.

[23] Loui M. and Abu-Amara H., Memory requirements for agreement among unreliable asynchronous processes. *Advances in Computing Research*, 4:163-183, JAI Press, 1987.

[24] Lynch N.A., *Distributed Algorithms*. Morgan Kaufmann Pub., San Francisco (CA), 872 pages, 1996.

[25] Mehlhorn K. and Sanders P., *Algorithms and data structures*. Springer, 300 pages, 2008 (ISBN 978-3-540-77977-3).

[26] Onofre J.-C., Rajsbaum S., and Raynal M., A topological perspective of recursion in distributed computing. *Tech report*, UNAM (Mexico), 12 pages, 2013.

[27] Rajsbaum S. and Raynal M., A theory-oriented introduction to wait-free synchronization based on the adaptive renaming problem. *Proc. 25th Int'l Conference on Advanced Information Networking and Applications (AINA'11)*, IEEE Press, pp. 356-363, 2011.

[28] Randell B., Recursively structured distributed computing systems. *Proc. 3rd Symposium on Reliability in Distributed Software and Database Systems*, IEEE Press, pp. 3-11, 1983.

[29] Raynal M., *Fault-tolerant agreement in synchronous distributed systems*. Morgan & Claypool, 167 pages, 2010 (ISBN 978-1-608-45525-6).

[30] Raynal M., *Concurrent programming: algorithms, principles and foundations.* Springer, 515 pages, 2013 (ISBN 978-3-642-32026-2).

The Formal Language Theory Column

BY

Giovanni Pighizzini

Dipartimento di Informatica
Università degli Studi di Milano
20135 Milano, Italy
pighizzini@di.unimi.it

Dear Reader, welcome to the Formal Language Theory Column!

Starting from this issue, I am the new editor of this column. First of all, I warmly thank my predecessor, Arto Salomaa, which was responsible of the column for many years. I would like to continue his work, by collecting contributions from the area that can be useful for exchanging new ideas, increasing and sharing the global knowledge in the field, and stimulating new researches. For instance, papers describing new developments, recent results, revisitations of classical topics as well as open problems will be welcome.

This column presents a survey paper written by Martin Kutrib and myself, on some recent trends in descriptional complexity of formal languages.

BEATCS no 111

Recent Trends in Descriptional Complexity of Formal Languages

Martin Kutrib
Institut für Informatik
Arndtstr. 2,
35392 Giessen, Germany
kutrib@informatik.uni-giessen.de

Giovanni Pighizzini
Dipartimento di Informatica
Università degli Studi di Milano
via Comelico 39, 20135 Milano, Italy
pighizzini@di.unimi.it

Abstract

Formal languages can be described by several means. A basic question is how succinctly can a descriptional system represent a formal language in comparison with other descriptional systems? What is the maximal size trade-off when changing from one system to another, and can it be achieved? Here, we select some recent trends in the descriptional complexity of formal languages and discuss the problems, results, and open questions. In particular, we present the main historical development and address the basics concepts of descriptional complexity from a general abstract perspective. Then we consider the representation by two-way finite automata, multi-head finite automata, and limited automata in more detail. Finally, we discuss a few further topics in note form. The results presented are not proved but we merely draw attention to the overall picture and some of the main ideas involved.

1 Introduction

Since the dawn of theoretical computer science the relative succinctness of different representations of formal languages by automata, grammars, equation systems, and other descriptional systems have been a subject of intensive research. The approach to analyze the size of systems as opposed to the computational power seems to originate from Stearns [64] who studied the relative succinctness of regular languages represented by deterministic finite automata and deterministic pushdown automata. He

showed the decidability of regularity for deterministic pushdown automata in a deep proof. The effective procedure revealed the following upper bound for the simulation. Given a deterministic pushdown automaton with $n > 1$ states and $t > 1$ stack symbols that accepts a regular language. Then the number of states which is sufficient for an equivalent DFA is bounded by an expression of the order $t^{n^{n^n}}$. Later this triple exponential upper bound has been improved by one level of exponentiation in [66]. In the levels of exponentiation it is tight. In [50] a double exponential lower bound has been obtained. The precise bound is still an open problem. Probably the best-known result on descriptional complexity is the construction of a DFA that simulates a given nondeterministic finite automaton [60]. By this so-called *powerset construction*, each state of the DFA is associated with a subset of NFA states. Moreover, the construction turned out to be optimal, in general. That is, the bound on the number of states necessary for the construction is tight in the sense that for an arbitrary n there is always some n-state NFA which cannot be simulated by any DFA with strictly less than 2^n states [42, 50, 53]. Let us turn to another cornerstone of descriptional complexity theory in the seminal paper by Meyer and Fischer [50]. In general, a known upper bound for the trade-off answers the question, how succinctly can a language be represented by a descriptor of one descriptional system compared with the representation by an equivalent descriptor of the other descriptional system? In [50] the sizes of finite automata and general context-free grammars for regular languages are compared. The comparison revealed a qualitatively new phenomenon. The gain in economy of description can be arbitrary, that is, there are no recursive functions serving as upper bounds for the trade-off, which is said to be *non-recursive*.

Nowadays, descriptional complexity has become a large and widespread area. Classical main branches not addressed in this summary are automata simulations, state complexity of operations, whose systematic study was initiated in [72], magic numbers, a research field initiated in [26], determinization of nondeterministic finite automata accepting subregular languages [3], transition complexity of NFA [6, 15, 22, 23, 41], and non-recursive trade-offs. Further results and references on these topics can be found, for example, in the surveys [13, 19, 20, 32].

1.1 Basic Concepts of Descriptional Complexity

In order to be more precise, we now turn to present and discuss the very basics of descriptional complexity.

We denote the set of nonnegative integers by $\mathbb{N}$. Let Σ^* denote the set of all words over a finite alphabet Σ. The *empty word* is denoted by λ, and we set $\Sigma^+ = \Sigma^* - \{\lambda\}$. For the *reversal of a word* w we write w^R and for its *length* we write $|w|$. We use $\subseteq$ for *inclusions* and $\subset$ for *strict inclusions*. In general, the family of all languages accepted by a device of some type X is denoted by $L(X)$.

In order to be general, we first formalize the intuitive notion of a representation or description of a family of languages. A *descriptional system* is a collection of

encodings of items where each item *represents* or *describes* a formal language. In the following, we call the items *descriptors*, and identify the encodings of some language representation with the representation itself. More precisely, a descriptional system $\mathcal{S}$ is a set of finite descriptors such that each $D \in \mathcal{S}$ describes a formal language $L(D)$, and the underlying alphabet $\mathrm{alph}(D)$ over which D represents a language can be obtained from D. The *family of languages represented* (or *described*) by $\mathcal{S}$ is $L(\mathcal{S}) = \{ L(D) \mid D \in \mathcal{S} \}$. For every language L, the set $\mathcal{S}(L) = \{ D \in \mathcal{S} \mid L(D) = L \}$ is the set of its descriptors in $\mathcal{S}$. A *complexity measure* for a descriptional system $\mathcal{S}$ is a total recursive mapping $c : \mathcal{S} \to \mathbb{N}$. From the viewpoint that a descriptional system is a collection of encoding strings, the length of the strings is a natural measure for the size. We denote it by length.

For example, nondeterministic finite automata can be encoded over some fixed alphabet such that their input alphabets can be extracted from the encodings. The set of these encodings is a descriptional system $\mathcal{S}$, and $L(\mathcal{S})$ is the family of regular languages.

Apart from length, examples for complexity measures for nondeterministic finite automata are the *number of states* (state) and the *number of transition* (trans).

In fact, we will use length to obtain a rough classification of different complexity measures. We distinguish between measures that (with respect to the size of the underlying alphabet) are recursively related with length and measures that are not. In the following, we only use complexity measures of the former type: If there is a total recursive function $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that, for all $D \in \mathcal{S}$, $\mathsf{length}(D) \leq g(c(D), |\mathrm{alph}(D)|)$, then c is said to be an *s-measure* (a *size* measure). Since for any coding alphabet there are only finitely many descriptors having at most length $g(c(D), |\mathrm{alph}(D)|)$, over the same alphabet there are only finitely many descriptors in $\mathcal{S}$ having the same size as D. If, in addition, for any alphabet Σ, the set of descriptors in $\mathcal{S}$ describing languages over Σ is recursively enumerable in order of increasing size, then c is said to be an *sn-measure*. Clearly, length, state, and trans are sn-measures for finite automata.

Whenever we consider the relative succinctness of two descriptional systems $\mathcal{S}_1$ and $\mathcal{S}_2$, we assume that the intersection $L(\mathcal{S}_1) \cap L(\mathcal{S}_2)$ is non-empty. Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be descriptional systems with complexity measures c_1 and c_2, respectively.

A total function $f : \mathbb{N} \to \mathbb{N}$, is said to be a *lower bound* for the increase in complexity when changing from a descriptor in $\mathcal{S}_1$ to an equivalent descriptor in $\mathcal{S}_2$, if for infinitely many $D_1 \in \mathcal{S}_1$ with $L(D_1) \in L(\mathcal{S}_2)$ there exists a *minimal* $D_2 \in \mathcal{S}_2(L(D_1))$ such that $c_2(D_2) \geq f(c_1(D_1))$.

A total function $f : \mathbb{N} \to \mathbb{N}$ is an *upper bound* for the increase in complexity when changing from a descriptor in $\mathcal{S}_1$ to an equivalent descriptor in $\mathcal{S}_2$, if for all $D_1 \in \mathcal{S}_1$ with $L(D_1) \in L(\mathcal{S}_2)$, there exists a $D_2 \in \mathcal{S}_2(L(D_1))$ such that $c_2(D_2) \leq f(c_1(D_1))$.

It may happen that the upper bound is not effectively computable. If there is no recursive upper bound, then the *trade-off* for changing from a description in $\mathcal{S}_1$ to an equivalent description in $\mathcal{S}_2$ is said to be *non-recursive*. Non-recursive trade-offs are independent of particular sn-measures. That is, whenever the trade-off from one descriptional system to another is non-recursive, one can choose an arbitrarily large

recursive function f but the gain in economy of description eventually exceeds f when changing from the former system to the latter. As an example, we consider nondeterministic pushdown automata that are used to accept regular languages. Clearly, for any such automaton there exists an equivalent finite automaton. However, the trade-off for the conversion of the pushdown automaton into the finite automaton is non-recursive. (See, for example, [13, 16, 19, 32] for more on non-recursive trade-offs.)

2 Recent Trends

Over the years a lot of investigations of descriptional complexity have been done documenting the importance of that field, its valuable concepts, and its vivid development. In the present section, we discuss only a few of the recent topics reflecting our personal view of what constitute the currently most interesting and challenging problems of descriptional complexity theory.

2.1 Two-Way Finite Automata

Since the beginning of automata theory it is known that the possibility of moving the head on the input tape in both directions does not increase the computational power of finite automata, even if nondeterministic transitions are allowed [60, 62]. However, the devices so obtained, which are called *two-way finite automata*, can be smaller than equivalent one-way automata.

As a simple example, for each integer $n > 0$ let us consider the language $L_n = (a + b)^* a(a + b)^{n-1}$. We can easily build a two-way deterministic finite automaton (2DFA) which accepts it by scanning the entire input from left to right and, when the right end is reached, by moving to the left in order to verify whether or not the symbol in position n from the right is an a. This gives a 2DFA with $n + 2$ states, in contrast with the minimal DFA which requires 2^n states. Notice that L_n is accepted by a NFA with $n + 1$ states.

In 1978, Sakoda and Sipser [61] raised the question of the exact costs in states for the simulations of NFAs and 2NFAs by 2DFAs. They conjectured that these costs are not polynomial. To support such a conjecture, Sakoda and Sipser presented a complete analogy with the P versus NP question, by introducing a notion of reducibility between families of regular languages which allows to identify families of languages which are complete for these simulations. (For a detailed discussion on this approach we address the reader to the recent paper [27].) In spite of many attempts to solve it, the problem is still open.

However, in the last decade many progresses have been done by attacking and solving restricted versions of this question. In particular, in the literature three families of restrictions have been considered:

- restrictions on the simulating machines,

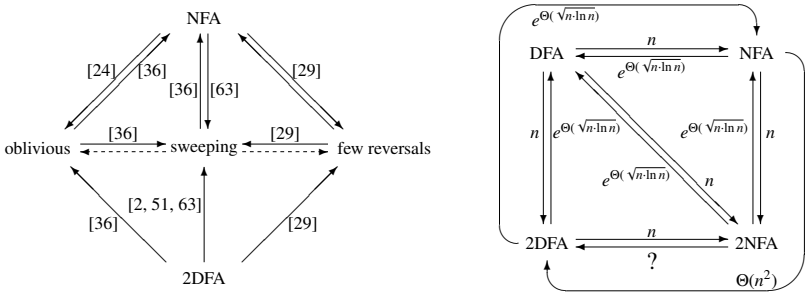


Figure 1: *(Left)* Costs of the simulation between different variants of automata. An arrow from a class A of machines to a class B indicates an exponential separation. A dashed arrow indicates a polynomial simulation. The (trivial) dashed arrow from oblivious, sweeping, and few reversal automata to 2DFAs are not depicted.
(Right) Costs of the *optimal* simulations between different kinds of *unary* automata [4, 49]. An arrow labeled $f(n)$ from a vertex A to a vertex B means that a unary n-state automaton in the class A can be simulated by an $f(n)$-state automaton in the class B.

- restrictions on the class of languages,
- restrictions on the simulated machines.

We now briefly discuss some of these restrictions, refering the reader to [56] for a recent extended survey along these lines.

Exponential separations have been obtained for the simulation of NFAs and 2NFAs by the following restricted classes of 2DFAs:

- *Sweeping automata:* these devices can reverse the head direction only while visiting the endmarkers (two special symbols marking the left and the right ends of the input).
- *Oblivious automata:* the "trajectories" of the head on each two inputs of the same length should coincide, namely, the position of the input head at each step t of the computation does not depend on the input content, but *only on its length*.
- *Few reversal automata:* the number of reversals of the head direction is sublinear with respect to the input length.

However, all these separations cannot solve the general problem. In fact, it has also been proved that all these devices can require exponentially many states with respect to equivalent (unrestricted) 2DFAs. Those separations, with references to the literature, are summarized in Figure 1 *(Left)*.

Concerning the second family of restrictions, interesting results have been found in the case of *unary languages*, namely languages defined over a one letter input alphabet. The state costs of the optimal simulations between different variants of unary automata

have been obtained by Chrobak [4] and by Mereghetti and Pighizzini [49], and they are summarized in Figure 1 *(Right)*. From the picture we can observe that the cost of the optimal simulations in the unary case can be smaller than in the general case. For example, the cost of the simulation of n-state NFAs by DFAs reduces from 2^n to $e^{\Theta(\sqrt{n \cdot \ln n})}$. Quite surprisingly, eliminating at the same time both nondeterminism and two-way motion costs as eliminating only one of them. The question NFAs versus 2DFAs has been solved in the unary case in [4] by showing that the tight costs are polynomial, more precisely $\Theta(n^2)$. This gives also the best known lower bound for the general case.

Despite the unary case looks simpler than the general one, the question of 2NFAs versus 2DFAs not only is still open even in this case, but it seems also to be difficult and, at the same time, very challenging. First of all, we mention that in [11] it has been proved that each unary n-state 2NFA can be simulated by a 2DFA with $e^{O(\ln^2 n)}$ states. This gives a subexponential but still superpolynomial upper bound. Proving the optimality of this upper bound, or proving a smaller but still superpolynomial lower bound for the state cost of the simulation of *unary* 2NFAs by 2DFAs would separate deterministic and nondeterministic logarithmic space (L and NL, respectively). In fact, as showed in [12], if $\mathsf{L} = \mathsf{NL}$ then the state cost of the simulation of unary 2NFAs by 2DFAs is polynomial. The converse implication is also true if the classes are defined in a nonuniform way. For recent developments and further results we address the reader to [30].

Some extensions of the analysis for the unary case have been obtained by considering *outer nondeterministic automata*. These devices are 2NFAs that are restricted to take nondeterministic decisions *only* when the input head is scanning the endmarkers. Hence, transitions on "real" input symbols are deterministic. Notice that there are no restrictions on the head movements as for instance in sweeping automata. These models share several properties with unary 2NFAs [9, 30]. Among them, a subexponential state upper bound for the simulation by 2DFAs have been obtained and relationships with the question L vs. NL have been stated.

2.2 Multi-Head Finite Automata

Before we turn to present the known results, current studies, and open questions of descriptional complexity issues of multi-head finite automata, we informally recall briefly what they are.

Let $k \geq 1$ be a natural number. A *nondeterministic two-way k-head finite automaton* (2NFA(k)) is a nondeterministic finite automaton having a single read-only input tape whose inscription is the input word in between two endmarkers. The k heads of the automaton can move freely on the tape but not beyond the endmarkers. A 2NFA(k) starts with all of its heads on the left endmarker. It halts when the transition function is not defined for the current situation. The input is accepted if and only if the automaton halts in an accepting state.

If in any case the transition function is either undefined or a singleton, then the k-head finite automaton is said to be *deterministic* (2DFA(k)). In case the heads never move to the left, the k-head finite automaton is said to be *one-way*. Nondeterministic and deterministic one-way k-head finite automata are denoted by 1NFA(k) and 1DFA(k).

Obviously, for one-head machines, regardless of whether they work one- or two-way, or of whether they are deterministic or nondeterministic, we obtain a characterization of the regular languages. On the other hand, a simple example is the non-context-free language $\{ wcw \mid w \in \{a, b\}^+ \}$ that can be accepted by a deterministic one-way *two-head* finite automaton.

The power of multi-head finite automata is well studied in the literature (see, for example, [21] for a survey). The interest is also driven by the strong relation to the computational complexity classes L and NL. In fact, in [17] the characterizations $\mathsf{L} = \bigcup_{k\geq 1} L(\text{2DFA}(k))$ and $\mathsf{NL} = \bigcup_{k\geq 1} L(\text{2NFA}(k))$ are shown.

Taking a closer look reveals the natural questions for the descriptional and computational power of the *precise number of heads*. The questions for the computational power have eventually been answered in [52], where it is shown that, for each $k \geq 1$, there is a *unary* language accepted by some deterministic (nondeterministic) two-way finite automaton with $k + 1$ heads which is not accepted by any k-head deterministic (nondeterministic) two-way finite automaton, and in [71], where it is shown that the language

$$L_n = \{ w_1\$w_2\$\cdots\$w_{2n} \mid w_i \in \{a, b\}^* \text{ and } w_i = w_{2n+1-i}, \text{ for } 1 \leq i \leq n \}$$

can be accepted by a 1DFA(k) if and only if $n \leq \binom{k}{2}$. Thus, L_n can be used to separate the computational power of automata with $k + 1$ heads from those with k heads also in the one-way setting.

But how about the descriptional power? The question of determining the trade-offs between the levels of the head hierarchies arises immediately. It was Kapoutsis [28] who solved the problem for two-way machines. In particular, there are non-recursive trade-offs between all levels of the head hierarchies for deterministic and nondeterministic devices (cf. also [31]). Moreover, the enormous descriptional power of heads evolutes already for *unary* languages.

Similarly, for one-way multi-head finite automata it is known [32] that the trade-offs between 1DFA($k + 1$) and 1DFA(k), between 1NFA($k + 1$) and 1NFA(k), and between 1DFA($k + 1$) and 1NFA(k) are all non-recursive. Moreover, non-recursive trade-offs are shown between nondeterministic 2-head and deterministic k-head automata.

So, is the descriptional complexity of multi-head finite automata a fully developed area without major open problems? The answer is a little hidden. While the non-recursive trade-offs for two-way machines are already for unary languages, in the one-way case every accepted unary language boils down to a regular one. In [25, 65] it is shown that every unary language accepted by a one-way multi-head finite automaton

is semilinear and, thus, regular. So, a lot of new questions are arising from the fog. Just to say it with one sentence, all simulations between descriptional systems for unary regular languages and multi-head finite automata, all problems in connection with the descriptional complexity of language operations, as well as the size costs for simulating $k+1$-head by k-head automata are worth studying. First steps have been done in [38], where the following results are from. Here the complexity is measured by the number of states, that is, we use the measure state.

First, we turn to the number of states for the simulation of an n-state k-head finite automaton by a classical (one-head) deterministic or nondeterministic finite automaton. So, we consider the maximal head reduction. As is often the case in connection with unary languages, the function

$$F(n) = \max\{ (c_1, c_2 \ldots, c_l) \mid c_1, c_2, \ldots, c_l \geq 1 \text{ and } c_1 + c_2 + \cdots + c_l = n \},$$

plays a crucial role, where denotes the *least common multiple*. It is well known that the c_i always can be chosen to be relatively prime. The function has been investigated by Landau [39, 40] who proved the asymptotic growth rate $\lim_{n\to\infty} \frac{\ln(F(n))}{\sqrt{n \cdot \ln n}} = 1$. The bounds $F(n) \in \Omega\left(e^{\sqrt{n \cdot \ln n}}\right)$ and $F(n) \in O\left(e^{\sqrt{n \cdot \ln n}(1+o(1))}\right)$ have been derived in [7].

For the simulation by a DFA, an upper bound of $O(n \cdot F(t \cdot n)^{k-1})$ and a lower bound of $n \cdot F(n)^{k-1}$ states is shown, where t is a constant depending only on k. Since both bounds are of order $e^{\Theta(\sqrt{n \cdot \ln n})}$, the trade-off for the simulation is tight in the order of magnitude. It is worth mentioning that for both bounds the number k of heads is a constant. It has been given as part of the bounds to be more precise.

For any constants $k \geq 2$ and prime $n \geq 2$, the lower bound is shown by construction of a unary n-state 1DFA(k) so that every equivalent deterministic or nondeterministic finite automaton has a cycle of at least $\{ nc_i^{k-1} \mid 1 \leq i \leq l \} = n(c_1 c_2 \cdots c_l)^{k-1} = n \cdot F(n)^{k-1}$ states.

Based on investigations of the length of words in unary languages accepted by n-state 1DFA(k), in [37] the upper bound is derived.

These results reveal that the costs for the simulation of 1DFA(k) by DFA are the same (in the order of magnitude) as for the simulation of NFA by DFA. From this point of view the two resources *heads* and *nondeterminism* are equally powerful. So the question for the costs of the mutual simulation of 1DFA(k) and NFA raises immediately. Trading k heads for nondeterminism is known to yield polynomially larger state sets, where the degree of the polynomial depends on k. For constants $k, n \geq 2$ any unary n-state 1DFA(k) can be simulated by some NFA with $O(n^{2k})$ states.

For the lower bound, the singleton languages $L_{k,n} = \{ a^{(k-1)n^k} \}$, for $k, n \geq 2$, are used as witnesses, which are accepted by some n-state 1DFA(k). Clearly, any NFA accepting $L_{k,n}$ needs at least $(k-1)n^k + 1 \in \Omega(n^k)$ states to check that there is no shorter word accepted.

So far, we considered the costs for the head reduction. Next we turn to the converse question whether we can trade nondeterminism for heads, that is, we are interested in the state complexity for the NFA by 1DFA(k) simulation. Naturally, our upper

bound depends highly on the number k of heads available. If k is at least the (on first sight) cryptic number $t = \lfloor \frac{-3+\sqrt{8n+1}}{2} \rfloor$, then the upper bound is quadratic, otherwise superpolynomial. Let $k \geq 1$, $n \geq 2$ be constants, $t = \lfloor \frac{-3+\sqrt{8n+1}}{2} \rfloor$, and M be a unary n-state NFA. Then

$$n' \leq \begin{cases} n^2 - 2 + F(n), & \text{if } k = 1; \\ n^2 - 2 + \left(n - \frac{t^2+t}{2}\right)^{\lceil \frac{t}{k} \rceil}, & \text{if } 1 < k < t/2; \\ 2n^2, & \text{if } k \geq t/2. \end{cases}$$

states are sufficient for any equivalent 1DFA(k).

The lower bound reads as follows. Let $k \geq 1$ be a constant. For any integer $m \geq 1$ there is an integer $n > m$ and a unary n-state NFA, such that $c_2 \cdot \sqrt[k]{e^{\sqrt{\frac{\sqrt{2n}}{c_1 \ln(\sqrt{2n})}}}}$ states are necessary for any equivalent 1DFA(k), where $c_1, c_2 > 0$ are two constants.

So, there is a gap between upper and lower bound. It is currently a challenging open problem how to close this gap. Also open are the descriptional costs for simulations of unary *nondeterministic* one-way multi-head finite automata. Furthermore, how about the trade-offs between devices with $k + 1$ and k heads? Are these trade-offs gradually evolve to the maximal head reduction? Are there jumps from a certain level, say, from two heads to one head?

2.3 Limited Automata

Each class of the *Chomsky hierarchy* is defined in terms of grammars using some special kind of productions, where the form of the productions which are allowed for grammars of type $1 \leq k \leq 3$ is a restriction of the form used for grammars of type $k-1$. From the point of view of language acceptors, each class of the hierarchy is characterized by a family of devices. However, while *linear bounded automata* used to characterize type 1 languages and *finite state automata* used to characterize type 3 languages can be seen as restrictions of (one-tape) Turing machines, which characterize type 0 languages, for type 2 languages, namely context-free, the characterization in terms of *pushdown automata* is usually presented. These devices are very useful to investigate and manipulate context-free languages. They also emphasize the main difference between regular and context-free languages, namely the possibility of representing recursive structures which, in terms of accepting devices, corresponds to increase the power of finite automata by adding a pushdown store. However, in a hierarchical view, pushdown automata do not appear as a special case of linear bounded automata.

Almost half a century ago, Hibbard discovered a different characterization of context-free languages, which uses a restricted version of Turing machines, called *scan limited automata* or, shortly, *limited automata* [18]. For each integer $d \geq 0$, a d-limited automaton (d-LA) is a two-way nondeterministic Turing machine which can rewrite the content of each tape cell *only in the first d visits*. He proved that, for each

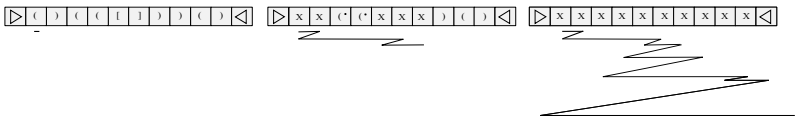


Figure 2: Some steps of the automaton A accepting the Dyck language D_k on input ()(([])).

$d \geq 2$, the class of languages accepted by d-limited automata coincides with the class of context-free languages. Without affecting the computational power, these devices can be allowed to use only the part of the tape containing the input string. Hence, they are restrictions of linear bounded automata while, clearly, they are extensions of finite state automata. This gives a hierarchy of classes of Turing machines corresponding to the classes of the Chomsky hierarchy. Recently, in [57] this hierarchical view has been strengthen by proving that *deterministic* 2-limited automata characterize deterministic context-free language, solving in this way a problem which was left open by Hibbard.

Let us present, as a simple example, how the *Dyck language* D_k over the alphabet $\{(_1,)_1, (_2,)_2, \ldots, (_k,)_k\}$ of $k \geq 1$ types of brackets, namely the set of strings representing well balanced sequences of brackets, can be accepted by a deterministic 2-LA A. The automaton A starts scanning the tape until it finds a closing bracket $)_i$. Then, A substitutes $)_i$ with a symbol X and changes the head direction, moving to the left until it reaches an opening bracket $(_j$. If $i \neq j$ then A rejects. Otherwise, it writes X on the cell and changes again the head direction moving to the right, to search another closing bracket. This procedure is repeated as long as A does not reach one of the endmarkers (see Figure 2.) However, if the left endmarker is reached, then at least one of the closing brackets in the input w does not have a matching opening bracket. Hence, A rejects. On the other hand, if the right endmarker is reached, then A has to make sure that no unmatched opening brackets are left. In order to do this, it scans the entire tape from the right to the left and, if it finds an opening bracket which has not be rewritten, then it rejects. Otherwise, A accepts the input.

In [57] the equivalence between 2-LAs and PDAs has been revisited considering descriptional complexity aspects. In particular, the following results have been obtained:

- Each 2-LA can be transformed into an equivalent PDA with an exponential increasing in the size. This gap cannot be reduced. Furthermore, the transformation preserves determinism.
- Conversely, each PDA can be transformed into an equivalent 2-LA of *polynomial* size. Even in this case, it is possible to preserve determinism.

Concerning determinism in d-limited automata for $d > 2$, it is not hard to see that the language $L = \{ a^n b^n c \mid n \geq 0 \} \cup \{ a^n b^{2n} d \mid n \geq 0 \}$ is accepted by a deterministic 3-LA, which first completely traverses the input from left to right and then, depending

on the last input letter, starts to rewrite one or two bs for each a. However, L is not a deterministic context-free language. Hence, each 2-LA accepting it must be nondeterministic. As a matter of fact, Hibbard proved the existence of an infinite hierarchy of deterministic languages: for each integer $d > 2$ there is a language accepted by a deterministic d-LA which cannot be accepted by any deterministic $(d-1)$-LA. This result is also true for $d = 2$ but, in this case, it is a consequence of the fact that 1-limited automata accept only regular languages [67]. Descriptional complexity aspects of such equivalence have been recently investigated in [58].

Another characterization of context-free languages of a similar flavor has been obtained by Wechsung [68, 69]: for a fixed constant d, the machine is allowed to rewrite each input cell *only in the last d visits*.[1] Even in this case, for each $d \geq 2$ context-free languages are characterized, while for $d = 1$ the model is trivially equivalent to finite automata. The set of palindromes can easily be accepted in this model in a deterministic way with $d = 2$, just implementing the trivial algorithm which compares pair by pair the symbols in the corresponding positions starting from the left and right ends of the input and moving toward the middle. Any d-limited automaton accepting this language must perform nondeterministic choices, for each $d \geq 2$ [18]. On the other hand, the language $\{ a^n b^{n+m} a^m \mid n, m \geq 0 \}$ cannot be accepted in a deterministic way by the model proposed by Wechsung, for any $d \geq 2$ [54].

Finally, it is worth mentioning shortly what happens if we replace the constant d by a function $d(n)$ of the input length. In this case, both the models proposed by Hibbard and by Wechsung are equivalent to one-way auxiliary pushdown automata working in space $d(n)$, namely standard pushdown automata extended with a $d(n)$ space bounded work tape [70].

Probably the reader has noticed that even if this paper is devoted to recent trends in descriptional complexity of formal languages, in the above discussion on limited automata descriptional complexity aspects are very restricted. Actually, the investigation of descriptional complexity of limited automata has been an opportunity for revisiting and inspecting several aspects of these devices which are not so well known. We think that these devices deserve further investigation. Concerning their descriptional complexity, two points under investigations are the cost of the simulation of d-limited automata by pushdown automata for $d > 2$ and the cost of the simulations of d-limited automata in the unary case. It is worth remembering that, since unary context-free languages are regular, in the unary case the cost of the simulation of d-limited automata by finite automata should be also considered. Furthermore, both in the general and in the unary cases, descriptional complexity of d-limited automata with that of $(d-1)$-limited automata could be also compared, for each $d > 2$.

[1] Wechsung introduced the term *return complexity* to indicate the maximum number of visits to a cell beginning with the first rewriting of its initial content. Hence, he considered machines with fixed return complexity d. In contrast, the maximum number of visits to a cell ending with the last rewriting of its content, namely the measure related to limited automata investigated by Hibbard, was called *dual return complexity*.

Both the notions proposed by Hibbard and Wechsung characterize, in different and in some sense dual ways, context-free language. Is it possible to obtain a more general notion of "limited automata" which captures these two notions, without increasing the computational power?

3 More Topics

We selected only a few topics in Section 2 which are close to our recent research interests. Of course, there are many other important "hot" topics that have recently been studied. We briefly mention some of them with references to the literature. However, our list is clearly far from being complete.

Unary Languages. The investigation of descriptional complexity in the case of *unary* languages shows many different bounds with respect to the general case. Tools and properties from number theory have extensively been used. In the previous section we mentioned the unary case several times, see in particular Figure 1 *(Right)*.

Restricted Pushdown Automata. Not so many results concerning descriptional complexity of PDAs have been obtained. In the introduction we mentioned the non-recursive trade-off between context-free grammars, or equivalently PDAs, for regular languages and finite automata, and the double exponential gap between deterministic PDAs accepting regular languages and finite automata. Both conversions have been investigated in the unary case obtaining optimal recursive bounds [59, 55].

Two other connections between PDAs and regular languages have been recently considered with respect to descriptional complexity: PDAs with pushdown stores of constant height [1, 8]; languages consisting of all pushdown contents in accepting computations of PDAs [46, 10] (these languages are known to be regular [14]).

Two-Way Pushdown Automata. While for finite state automata the possibility of moving the input head in both directions does not increase the computational power, it is well-known that this is not true in the case of PDAs. For example, the non-context-free language $\{ a^n b^n c^n \mid n \geq 0 \}$ can easily be accepted by a *two-way pushdown automaton*. Up to now, the computational power of these models has not been clearly identified. For example, it is unknown if they are equivalent to linear bounded automata and if the nondeterministic variant is more powerful than the deterministic one. Recent descriptional complexity results concerning these models and taking into account, besides the size of the devices, the number of the head reversals and the number of turns of the pushdown store, have been presented in [48].

Cellular Automata and Iterative Arrays. These devices are often studied as massively parallel language acceptors [33]. The investigation of their descriptional complexity originates in [43, 45]. It turned out that in many cases the resources given to cellular automata in connection with massively parallelism yield non-recursive trade-offs. In the recent papers [35, 47] it is shown that even very little additional resources

have a big impact on the necessary size of the devices. For example, adding sublinearly more time obtaining time complexities strictly in between real time and linear time, adding dimensions, allow the communication cell to perform a few nondeterministic steps, or increase the number of bits that may be communicated to neighboring cells slightly, allows arbitrary savings in the size of the descriptions of the arrays which cannot be bounded by any computable function. So the challenging tasks are to identify resources that can be added to or modifications of cellular automata that yield *recursive* trade-offs. Examples are the decompositions and generalized presentation of languages gained in language expressions with operations under which the family in question is closed, the so-called operational state complexity (see [34] for results on one-way cellular automata). Another approach which bounds the number of cells available can be found in [44].

References

[1] Bednárová, Z., Geffert, V., Mereghetti, C., Palano, B.: The size-cost of Boolean operations on constant height deterministic pushdown automata. Theoret. Comput. Sci. 449, 23–36 (2012)

[2] Berman, P.: A note on sweeping automata. In: International Colloquium on Automata, Languages and Programming (ICALP 1980). LNCS, vol. 85, pp. 91–97. Springer (1980)

[3] Bordihn, H., Holzer, M., Kutrib, M.: Determinization of finite automata accepting subregular languages. Theoret. Comput. Sci. 410, 3209–3222 (2009)

[4] Chrobak, M.: Finite automata and unary languages. Theoret. Comput. Sci. 47(2), 149–158 (1986), errata: [5]

[5] Chrobak, M.: Errata to "Finite automata and unary languages". Theoret. Comput. Sci. 302, 497–498 (2003)

[6] Domaratzki, M., Salomaa, K.: Lower bounds for the transition complexity of NFAs. In: Mathematical Foundations of Computer Science (MFCS 2006). LNCS, vol. 4162, pp. 315–326. Springer (2006)

[7] Ellul, K.: Descriptional Complexity Measures of Regular Languages. Master's thesis, University of Waterloo, Canada (2004)

[8] Geffert, V., Bednárová, Z., Mereghetti, C., Palano, B.: Boolean language operations on nondeterministic automata with a pushdown of constant height. In: Computer Science Symposium in Russia (CSR 2013). LNCS, vol. 7913, pp. 100–111. Springer (2013)

[9] Geffert, V., Guillon, B., Pighizzini, G.: Two-way automata making choices only at the endmarkers. In: Language and Automata Theory and Applications (LATA 2012), LNCS, vol. 7183, pp. 264–276. Springer (2012)

[10] Geffert, V., Malcher, A., Meckel, K., Mereghetti, C., Palano, B.: A direct construction of finite state automata for pushdown store languages. In: Descriptional Complexity of Formal Systems (DCFS 2013). LNCS, vol. 8031, pp. 90–101. Springer (2013)

[11] Geffert, V., Mereghetti, C., Pighizzini, G.: Converting two-way nondeterministic unary automata into simpler automata. Theoret. Comput. Sci. 295, 189–203 (2003)

[12] Geffert, V., Pighizzini, G.: Two-way unary automata versus logarithmic space. Inform. Comput. 209, 1016–1025 (2011)

[13] Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. J. UCS 8, 193–234 (2002)

[14] Greibach, S.A.: A note on pushdown store automata and regular systems. Proc. Amer. Math. Soc. 18, 263–268 (1967)

[15] Gruber, H., Holzer, M.: On the average state and transition complexity of finite languages. Theoret. Comput. Sci. 387, 155–166 (2007)

[16] Gruber, H., Holzer, M., Kutrib, M.: On measuring non-recursive trade-offs. J. Autom., Lang. Comb. 15, 107–120 (2010)

[17] Hartmanis, J.: On non-determinancy in simple computing devices. Acta Inform. 1, 336–344 (1972)

[18] Hibbard, T.N.: A generalization of context-free determinism. Inform. Control 11, 196–238 (1967)

[19] Holzer, M., Kutrib, M.: Descriptional complexity – An introductory survey. In: Scientific Applications of Language Methods, pp. 1–58. Imperial College Press (2010)

[20] Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata – A survey. Inform. Comput. 209, 456–470 (2011)

[21] Holzer, M., Kutrib, M., Malcher, A.: Complexity of multi-head finite automata: Origins and directions. Theoret. Comput. Sci. 412, 83–96 (2011)

[22] Hromkovič, J., Schnitger, G.: NFAs with and without ϵ-transitions. In: International Colloquium on Automata, Languages and Programming (ICALP 2005). LNCS, vol. 3580, pp. 385–396. Springer (2005)

[23] Hromkovič, J., Seibert, S., Wilke, T.: Translating regular expressions into small ϵ-free nondeterministic finite automata. J. Comput. System Sci. 62, 565–588 (2001)

[24] Hromkovič, J., Schnitger, G.: Nondeterminism versus determinism for two-way finite automata: Generalizations of Sipser's separation. In: International Colloquium on Automata, Languages and Programming (ICALP 2003). LNCS, vol. 2719, pp. 439–451. Springer (2003)

[25] Ibarra, O.H.: A note on semilinear sets and bounded-reversal multihead pushdown automata. Inform. Process. Lett. 3, 25–28 (1974)

[26] Iwama, K., Kambayashi, Y., Takaki, K.: Tight bounds on the number of states of DFAs that are equivalent to n-state NFAs. Theoret. Comput. Sci. 237, 485–494 (2000)

[27] Kapoutsis, C.: Minicomplexity. In: Descriptional Complexity of Formal Systems (DCFS 2012), LNCS, vol. 7386, pp. 20–42. Springer (2012)

[28] Kapoutsis, C.A.: Non-recursive trade-offs for two-way machines. Int. J. Found. Comput. Sci. 16, 943–956 (2005)

[29] Kapoutsis, C.A.: Nondeterminism is essential in small two-way finite automata with few reversals. Inform. Comput. 222, 208–227 (2013)

[30] Kapoutsis, C.A., Pighizzini, G.: Two-way automata characterizations of L/poly versus NL. In: Computer Science Symposium in Russia (CSR 2012). LNCS, vol. 7353, pp. 217–228. Springer (2012)

[31] Kutrib, M.: On the descriptional power of heads, counters, and pebbles. Theoret. Comput. Sci. 330, 311–324 (2005)

[32] Kutrib, M.: The phenomenon of non-recursive trade-offs. Int. J. Found. Comput. Sci. 16, 957–973 (2005)

[33] Kutrib, M.: Cellular automata and language theory. In: Encyclopedia of Complexity and System Science, pp. 800–823. Springer (2009)

[34] Kutrib, M., Lefèvre, J., Malcher, A.: The size of one-way cellular automata. In: Cellular Automata and Discrete Complex Systems (AUTOMATA 2010). pp. 71–90. Discrete Mathematics and Theoretical Computer Science Proceedings, DMTCS (2010)

[35] Kutrib, M., Malcher, A.: The size impact of little iterative array resources. J. Cellular Automata 7, 489–507 (2012)

[36] Kutrib, M., Malcher, A., Pighizzini, G.: Oblivious two-way finite automata: Decidability and complexity. In: Latin 2012: Theoretical Informatics, LNCS, vol. 7256, pp. 208–221. Springer (2012)

[37] Kutrib, M., Malcher, A., Wendlandt, M.: States and heads do count for unary multi-head finite automata. In: Developments in Language Theory (DLT 2012). LNCS, vol. 7410, pp. 214–225. Springer (2012)

[38] Kutrib, M., Malcher, A., Wendlandt, M.: Size of unary one-way multi-head finite automata. In: Descriptional Complexity of Formal Systems (DCFS 2013). LNCS, vol. 8031, pp. 148–159. Springer (2013)

[39] Landau, E.: Über die Maximalordnung der Permutationen gegebenen Grades. Archiv der Math. und Phys. 3, 92–103 (1903)

[40] Landau, E.: Handbuch der Lehre von der Verteilung der Primzahlen. Teubner, Leipzig (1909)

[41] Lifshits, Y.: A lower bound on the size of ϵ-free NFA corresponding to a regular expression. Inform. Process. Lett. 85, 293–299 (2003)

[42] Lupanov, O.B.: A comparison of two types of finite sources. Problemy Kybernetiki 9, 321–326 (1963), (in Russian), German translation: Über den Vergleich zweier Typen endlicher Quellen. Probleme der Kybernetik 6, 328-335 (1966)

[43] Malcher, A.: Descriptional complexity of cellular automata and decidability questions. J. Autom., Lang. Comb. 7, 549–560 (2002)

[44] Malcher, A.: On one-way cellular automata with a fixed number of cells. Fund. Inform. 58, 355–368 (2003)

[45] Malcher, A.: On the descriptional complexity of iterative arrays. IEICE Trans. Inf. Syst. E87-D(3), 721–725 (2004)

[46] Malcher, A., Meckel, K., Mereghetti, C., Palano, B.: Descriptional complexity of pushdown store languages. In: Descriptional Complexity of Formal Systems (DCFS 2012). LNCS, vol. 7386, pp. 209–221. Springer (2012)

[47] Malcher, A., Mereghetti, C., Palano, B.: Sublinearly space bounded iterative arrays. Int. J. Found. Comput. Sci. 21, 843–858 (2010)

[48] Malcher, A., Mereghetti, C., Palano, B.: Descriptional complexity of two-way pushdown automata with restricted head reversals. Theoret. Comput. Sci. 449, 119–133 (2012)

[49] Mereghetti, C., Pighizzini, G.: Optimal simulations between unary automata. SIAM J. Comput. 30, 1976–1992 (2001)

[50] Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: Symposium on Switching and Automata Theory (SWAT 1971). pp. 188–191. IEEE (1971)

[51] Micali, S.: Two-way deterministic finite automata are exponentially more succinct than sweeping automata. Inform. Process. Lett. 12, 103–105 (1981)

[52] Monien, B.: Two-way multihead automata over a one-letter alphabet. RAIRO Inform. Théor. 14, 67–82 (1980)

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

[54] Peckel, J.: On a deterministic subclass of context-free languages. In: Mathematical Foundations of Computer Science (MFCS 1977). LNCS, vol. 53, pp. 430–434. Springer (1977)

[55] Pighizzini, G.: Deterministic pushdown automata and unary languages. Int. J. Found. Comput. Sci. 20(4), 629–645 (2009)

[56] Pighizzini, G.: Two-way finite automata: Old and recent results. Fund. Inform. 126(2-3), 225–246 (2013)

[57] Pighizzini, G., Pisoni, A.: Limited automata and context-free languages. In: Non-Classical Models of Automata and Applications (NCMA 2013). books@ocg.at, vol. 294, pp. 209–223. Austrian Computer Society, Vienna (2013)

[58] Pighizzini, G., Pisoni, A.: Limited automata and regular languages. In: Descriptional Complexity of Formal Systems (DCFS 2013). LNCS, vol. 8031, pp. 253–264. Springer (2013)

[59] Pighizzini, G., Shallit, J., Wang, M.W.: Unary context-free grammars and pushdown automata, descriptional complexity and auxiliary space lower bounds. J. Comput. System Sci. 65, 393–414 (2002)

[60] Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Dev. 3, 114–125 (1959)

[61] Sakoda, W.J., Sipser, M.: Nondeterminism and the size of two way finite automata. In: Symposium on Theory of Computing (STOC 1978), pp. 275–286. ACM Press, New York (1978)

[62] Shepherdson, J.C.: The reduction of two-way automata to one-way automata. IBM J. Res. Dev. 3, 198–200 (1959)

[63] Sipser, M.: Lower bounds on the size of sweeping automata. J. Comput. System Sci. 21, 195–202 (1980)

[64] Stearns, R.E.: A regularity test for pushdown machines. Inform. Control 11, 323–340 (1967)

[65] Sudborough, I.H.: Bounded-reversal multihead finite automata languages. Inform. Control 25, 317–328 (1974)

[66] Valiant, L.G.: Regularity and related problems for deterministic pushdown automata. J. ACM 22, 1–10 (1975)

[67] Wagner, K., Wechsung, G.: Computational Complexity. Reidel, Dordrecht (1986)

[68] Wechsung, G.: Characterization of some classes of context-free languages in terms of complexity classes. In: Mathematical Foundations of Computer Science (MFCS 1975). LNCS, vol. 32, pp. 457–461. Springer (1975)

[69] Wechsung, G.: Kompliziertheitstheoretische Charakterisierung der kontextfreien und linearen Sprachen. J. Inform. Process. Cybern. 12(6), 289–300 (1976)

[70] Wechsung, G., Brandstädt, A.: A relation between space, return and dual return complexities. Theoret. Comput. Sci. 9, 127–140 (1979)

[71] Yao, A.C., Rivest, R.L.: $k + 1$ heads are better than k. J. ACM 25, 337–340 (1978)

[72] Yu, S.: State complexity of regular languages. J. Autom., Lang. Comb. 6, 221–234 (2001)

THE LOGIC IN COMPUTER SCIENCE COLUMN

BY

YURI GUREVICH

Microsoft Research
One Microsoft Way, Redmond WA 98052, USA
gurevich@microsoft.com

WHEN IS A=B?*

Anja Gruünheid
ETH Zürich
ganja@inf.ethz.ch

Donald Kossmann
ETH Zürich
donaldk@ethz.ch

Besmira Nushi
ETH Zürich
nushib@inf.ethz.ch

Abstract

Most database operations such as sorting, grouping and computing joins are based on comparisons between two values. Traditional algorithms assume that machines do not make mistakes. This assumption holds in traditional computing environments; however, it does not hold in several new emerging computing environments. In this write-up, we argue the need for new resilient algorithms that take into account that the result of a comparison might be wrong. The goal is to design algorithms that have low cost (make few comparisons) yet produce high-quality results in the presence of errors.

*This write-up is based on a talk given at the University of Washington and Microsoft Research, Redmond, in August 2013. The slides of that talk are online at http://systems.ethz.ch/talks.

1 Introduction

When we think about computers, we typically assume that they are dumb and make no mistakes. Our software methodology, complexity theory, and algorithmic design are based on these two assumptions. What happens if we drop one of these assumptions? What happens if computers start making mistakes occasionally; even simple mistakes such as getting a comparison between two integers wrong? Will we need new algorithms or are the existing algorithms good enough?

There are a number of research trends that make it worthwhile to think about error-prone computers. The first trend is the emergence of crowdsourcing and the development of hybrid systems that involve machines and humans to compute tasks that neither machines nor humans are capable of computing alone. [4] gives an overview of such systems. Since these systems rely on human input, some of the computations carried out by these systems may be error-prone and algorithms that are designed for these systems need to take this fact into account.

A second trend is the development of new, low-energy processors that trade power for accuracy. That is, these processors might occasionally get an operation wrong in exchange for much lower power consumption. Examples for such designs are [1].

Third, with the advent of Big Data technologies, we are automating an increasing number of tasks based on previous experience. Recommendation systems such as those deployed by Amazon as part of their online shop improve with an increasing amount of data. Putting it differently, these systems might make poor recommendations if only little data is available.

Based on these observations, we believe that it is worthwhile to revisit existing algorithms and start thinking about how to design algorithms for computer systems that occasionally do make errors. It turns out that an algorithm that is optimal in the traditional (error-free) computational model may perform poorly in the presence of error. As an example, this paper reports on some simple observations that we made when studying QuickSort. Furthermore, this paper reports on some observations on how to group objects in a robust way if the machine occasionally misclassifies two objects. These two examples indicate that we might have to rethink complexity theory and algorithm design. As of now, the results of this paper are anecdotal, and we have not been yet able to develop a new theory. The main purpose of this paper is to raise the issue.

The remainder of this paper is organized as follows: Section 2 studies sorting. Section 3 gives an example of how errors impact algorithms for grouping or clustering objects. Section 4 contains conclusions and related work.

2 Example 1: QuickSort

To show how the presence of errors may impact algorithm design, let us start with a discussion of QuickSort. [7] gives a more general discussion of sorting algorithms in the presence of errors. Here and in the remainder of this paper, we assume a computational model in which the computer system might make an error when executing a comparison; however, the logic of the algorithm is executed correctly. Furthermore, we assume that comparisons are the most expensive operation. This computational model matches nicely hybrid systems in which comparisons are crowdsourced (e.g., [10]).

QuickSort is generally perceived as one of the best algorithms for sorting. However, what makes QuickSort great for traditional, error-free computing scenarios, hurts QuickSort in the presence of mistakes. The following example shows why. The task is to sort the following sequence of numbers:

$$7, 24, 2, 13, 51$$

Let us assume that QuickSort chooses 7 as the pivot element of the first partitioning phase and let us furthermore assume that the machine gets all comparisons right in the first partitioning, except for the comparison $7 < 51$. As a consequence, the result of the first iteration of QuickSort are the following two partitions:

$$2, 51$$

$$24, 13$$

Even if the machine is perfect and makes no further mistakes, the best possible outcome to sort the sequence of numbers is:

$$2, 51, 7, 13, 24$$

The key observation is that one wrong comparison (misclassifying $7 < 51$) resulted in three errors in the final result (misclassifying $13 < 51$ and $24 < 51$ in addition to $7 < 51$). The reason is that the QuickSort algorithm aggressively exploits the transitivity of the $<$ relation so that errors propagate. There are many different notions of error and the most appropriate definition depends on the utility function of the application. We use the number of misclassified comparisons in the final result here and in [7] because it is easy to formalize and it is a metric that is highly relevant for many applications that involve sorting or ranking data.

It turns out that it is difficult to fix QuickSort. The most natural way to improve the quality of the result is to avoid misclassifications by repeating the computation. That is, recomputing $7 < 51$ several times and then do a majority vote or accept based on a theshold. That will increase the number of comparisons by a

TransactionId	*Customer*	*Purchase*
1	Jane	$ 1000
2	Bob	$ 500
3	Jane	$ 100
4	Jane	$ 50

Table 1: Example Transactions

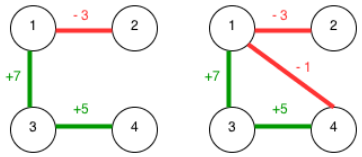
constant factor (i.e., the number of times each comparison is made) so that QuickSort continues to be in the $O(n*\log(n))$ complexity class. The problem is that even with a high number of attempts, the probability of a misclassification is not zero. So, we can never expect perfection with QuickSort. Also, the impact of a wrong comparison grows with the size of the sequence in our particular error model that counts the misclassifications in the final result: In the worst case, it is $n/2$ with n the length of the sequence. The question then is how to best invest additional comparisons and whether new algorithms are more appropriate than traditional algorithms to achieve high quality for lower cost. [7], for instance, shows that iteratively running BubbleSort might be a better a way to invest additional computation for better quality. That is, do an intial sorting with QuickSort and then run BubbleSort once or several times on the result to improve the quality of the result, thereby exploiting that BubbleSort has $O(n)$ complexity if the data is sorted already and the affects of wrong comparisons are always local in BubbleSort.

3 Example 2: Grouping

3.1 Vote Graphs

As a second example of how the cost/quality trade-off of error-prone computer systems impacts algorithm design, consider the list of transactions of Table 1.

The task is to compute the total purchase of each customer; i.e., $ 1150 for Jane and $ 500 for Bob. With SQL, this task can be specified using a simple GROUP BY clause. Depending on the number of customers, the number of transactions, and the skew in the distribution of transactions to customers, modern database systems choose one of three alternative ways to compute this grouping of transactions by customer: sorting, hashing, or nested-loops. For the purpose of this example, we will use the nested-loop variant and discuss alternative ways to compare the *customer* fields of two transactions in order to decide whether they belong to the same customer. Note that hashing and sorting are often more efficient variants, but they suffer from the same kind of error propagation as the QuickSort algorithm in the previous section.



(a) No Conflict (b) With Conflict

Figure 1: Example Vote Graphs for Table 1

Similarly to Section 2, we assume that the comparison of two customer names is the only error-prone and costly operation. Thus, the goal is to minimize the number of such comparisons and minimize the impact of mistakes made when computing these comparisons. Figure 1 illustrates one possible approach to do that. It depicts two example *Vote Graphs*. Such Vote Graphs capture the results of all comparisons carried out between the customer names of the four transactions. The nodes of a Vote Graph are transactions. Edges of a Vote Graph represent the results of comparing the customer names of two transactions. The weight of an edge indicates how often the comparison returned that results; the sign of the weights of an edge indicates the result of the comparison (true or false).

The Vote Graph of Figure 1a, for instance, indicates that we compared three times the customer names of Transactions 1 and 2 (i.e., "Jane = Bob") and all three times the answer was "false" (which happens to be the correct answer in this example). Furthermore, it shows that all seven comparisons between the customer names of Transactions 1 and 3 were positive (which happens to be correct, too, in this example).

3.2 Decision Functions

If minimizing comparisons between two customer names is our main objective (e.g., because they need to be crowdsourced or need to be executed repeatedly on an error-prone machine), then it makes sense to exploit the transitivity of the = relation. So, if the grouping algorithm asks whether Transactions 1 and 4 belong to the same customer in Figure 1a, the answer is *true* and can be inferred from Figure 1a without actually looking at the customer names of these transactions.

Transitivity and anti-transitivity can be applied in a straightforward way in the example of Figure 1a. The situation becomes trickier in Figure 1b because in that Vote Graph there is a conflicting edge: The negative edge between Transactions 1 and 4 conflicts with the positive edges, "1-3" and "1-4".

In the presence of error-prone computations, conflicts in the Vote Graph are inevitable. Therefore, it is important to tolerate these errors and make decisions even in conflict situations. In the example of Figure 1b, it is evident that the

system should conclude that the same customer carried out Transactions 1 and 4 because the weight of the edges "1-3" and "3-4" is much higher than the weight of the negative edge "1-4". In general, we propose the use of a decision function that given a Vote Graph, determines whether two nodes are the same, not the same, or if additional comparisons are needed in order to make the decision.

There are many decisions functions conceivable and [8] contains a more detailed discussion of which properties a decision function should have. For instance, a decision function that always says that two nodes are the same is obviously not good because it will result in poor *quality*. Likewise, a decision function that always says "I do not know" is not good because it will result in high *cost* as it would induce additional comparisons. For the discussion in this paper, let us consider a decision function that is inspired by work on combining scoring functions [5] and that we call the MinMax function.

The MinMax function considers all positive and negative paths between two nodes. A positive path is a path that involves only edges with weight greater than 0. A negative path is a path that has exactly one negative edge. Paths with more than one negative edge are ignored because neither equality nor inequality can be inferred from them. For each path, the MinMax function computes a score: For a positive path, the score is the *minimum* of the weights of the edges of the path. For a negative path, the score is defined as the *minimum* of the *absolute* weights of the edges (i.e., the weight of the only negative edge is multiplied by -1 for this purpose). The intuition behind this scoring function is that a path is as strong as its weakest link. Another way to interpret the *minimum* is that it implements a conjunction (i.e., $\wedge$) along the path, thereby interpreting each edge as a predicate.

Continuing the example of Figure 1b, the score of the positive path '1-3-4' is 5 while the score for the negative path '1-4' is 1.

After computing the scores for all positive and negative paths, the MinMax decision function aggregates these scores into a single positive score, *pScore*, and a single negative score, *nScore*. *pScore* is the *maximum* of the scores of all positive paths. If there is no positive path, then $pScore = 0$. Analogously, *nScore* is the *maximum* of the scores of all negative paths. If there is no negative path, then $nScore = 0$. These values represent the maximum impact that a positive respectively negative path can have within an entity.

Finally, the MinMax function uses a threshold q in order to form a final decision based on the positive and negative scores; e.g., $q = 3$. That is, if the positive score is 3 or more higher than the negative score then the MinMax function decides that the two nodes are the same. More formally, the decision part of MinMax is defined as follows.

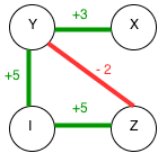


Figure 2: Interesting MinMax Example

$$f(r_1, r_2) = \begin{cases} \text{Yes}, & pScore(r_1, r_2) - nScore(r_1, r_2) \geq q \\ \text{No}, & nScore(r_1, r_2) - pScore(r_1, r_2) \geq q \\ \text{Do-not-know}, & otherwise \end{cases}$$

3.3 Observations

[8] contains a full discussion of this grouping / clustering use case under uncertainty with a series of experiments. The important observation and conclusion of [8] is that maintaining a Vote Graph and doing inference with the MinMax function is much better than doing pairwise comparisons in terms of both quality and cost in order to compute any database operation that is based on equality (e.g., joins, grouping, or clustering). In terms of cost, it is better because of its inference capability; in terms of quality, it is better because it detects inconsistencies and tries to keep the whole graph consistent. The designers of traditional database systems would never consider keeping such a Vote Graph because it is in traditional computing environments it is always cheaper (and as reliable) to recompute a comparison than to infer its result from a Vote Graph.

[8] discusses some of the properties of the MinMax decision function. It turns out that it is not transitive and an example can be seen in Figure 2 with a threshold of 3. In that example, the MinMax rules that "X = Y" (pScore=3, nScore=0) and "Y = Z" (pScore=5, nScore=2)), but it rules that "X = Z" is *unknown* (pScore=3, nScore=2). There are many conceivable decision functions; many which indeed are transitive. For instance, it would be possible to define a decision function by applying the MinCuts algorithm on every instance of the Vote Graph (i.e., after computing every comparison). This decision would indeed be transitive, but its implementation would have high computational cost. [8] proposes the MinMax function because it can be implemented in a highly efficient way.

For the purpose of designing good and robust algorithms for error-prone computer systems, however, we would like to make another important, somewhat surprising observation. Going back to Figure 2 and using the MinMax function, the best way to conclude that "X = Z" is *not* by comparing "X = Z" directly. Doing so would require, in the best case, five calls to the comparison function. Instead,

investing into the "Y = Z" edge is more promising: In the best case, two comparisons that confirm that indeed "Y = Z" are sufficient to finally conclude with the MinMax function that "X = Z".

4 Conclusion and Related Work

The two examples showed some phenomena that may occur if computer systems make mistakes. The examples show that an optimal algorithm for the traditonal (error-free) computing model might result in poor quality when run on error-prone computer systems. It is an open question of what the optimal algorithms to sort a sequence of numbers and to group/cluster objects in the presence of errors are. The main message that we would like to illustrate with these examples is that error should be part of the equation. That is, we need to do two things:

- We need to design algorithms that scale (with the problem size) and tolerate errors. (Traditional algorithms were designed only to scale.)
- We need to optimize for both *cost* and *quality*. (Traditional algorithms were designed to minimize cost only.)

In other words, algorithm designers face two kinds of optimizations:

- Given a problem (e.g., sorting), a problem instance (e.g., 1000 integers), an error model (e.g., 1% of the comparisons are wrong uniformly) and a budget (e.g., 1 million comparisons), maximize the quality of the result.
- Given a problem, a problem instance, an error model, and quality requirements, minimize the cost.

At the moment, we do not even have good abstractions to characterize computational error and result quality.

The examples used in this paper were derived from typical database operators (i.e., sorting, joins, and grouping). Recently, there have a number of papers in the database community that studied how to enhance database with crowdsourcing, a special form of uncertain computation; e.g., [9, 11, 6, 3] to name just a few. It turns out that the topic of error-prone computing has been studied in other communities as well and not only in the context of crowdsourcing. For instance, Busse and Buhmann studied the information gain of a comparison in alternative sorting algorithms [2]. Schulze developed a method to carry out elections, called the Schulze method, which is similar to the MinMax decision function [12]. Furthermore, designers of distributed systems have been developing fault-tolerant algorithms for decades. The fact that several communities are looking into fault-tolerant computation makes it even more important to develop a theory that incorporates error and result quality in algorithm design and complexity.

References

[1] L. Avinash, K. K. Muntimadugu, C. C. Enz, R. M. Karp, K. V. Palem, and C. Piguet. Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling. In J. Feo, P. Faraboschi, and O. Villa, editors, *Conf. Computing Frontiers*, pages 3–12. ACM, 2012.

[2] L. M. Busse, M. H. Chehreghani, and J. M. Buhmann. The information content in sorting algorithms. In *ISIT*, pages 2746–2750. IEEE, 2012.

[3] S. B. Davidson, S. Khanna, T. Milo, and S. Roy. Using the crowd for top-k and group-by queries. In W.-C. Tan, G. Guerrini, B. Catania, and A. Gounaris, editors, *ICDT*, pages 225–236. ACM, 2013.

[4] A. Doan, R. Ramakrishnan, and A. Y. Halevy. Crowdsourcing systems on the world-wide web. *Commun. ACM*, 54(4):86–96, 2011.

[5] R. Fagin and E. L. Wimmers. A formula for incorporating weights into scoring rules. *Theor. Comput. Sci.*, 239(2):309–338, 2000.

[6] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: answering queries with crowdsourcing. In Sellis et al. [13], pages 61–72.

[7] A. Gruenheid and D. Kossmann. Cost and quality trade-offs in crowdsourcing. In R. Cheng, A. D. Sarma, S. Maniu, and P. Senellart, editors, *DBCrowd*, volume 1025 of *CEUR Workshop Proceedings*, pages 43–46. CEUR-WS.org, 2013.

[8] A. Gruenheid, D. Kossmann, S. Ramesh, and F. Widmer. Crowdsourcing entity resolution: When is a=b? Technical Report No. 785, Department of Computer Science, ETH Zurich, Sep 2012.

[9] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Demonstration of qurk: a query processor for humanoperators. In Sellis et al. [13], pages 1315–1318.

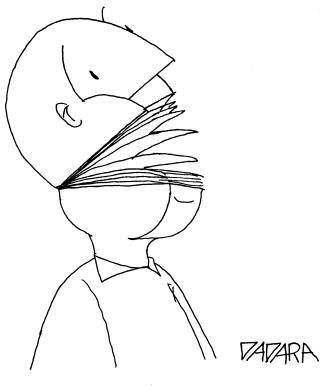
[10] A. Marcus, E. Wu, D. R. Karger, S. Madden, and R. C. Miller. Human-powered sorts and joins. *PVLDB*, 5(1):13–24, 2011.

[11] H. Park, R. Pang, A. G. Parameswaran, H. Garcia-Molina, N. Polyzotis, and J. Widom. Deco: A system for declarative crowdsourcing. *PVLDB*, 5(12):1990–1993, 2012.

[12] M. Schulze. A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method. *Social Choice and Welfare*, 36(2):267–303, 2011.

[13] T. K. Sellis, R. J. Miller, A. Kementsietsidis, and Y. Velegrakis, editors. *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011*. ACM, 2011.

Contributions by EATCS Award Recipients



A Few Lessons I've Learned

Erik D. Demaine
Massachusetts Institute of Technology
edemaine@mit.edu

Abstract

This article summarizes four key lessons I've learned doing research as a theoretical computer scientist. The lessons represent my style and approach to research, and are somewhat nonconventional, centered around fun. This article is based on a talk I gave upon receiving the Presburger Award this year, which can be viewed online.[1]

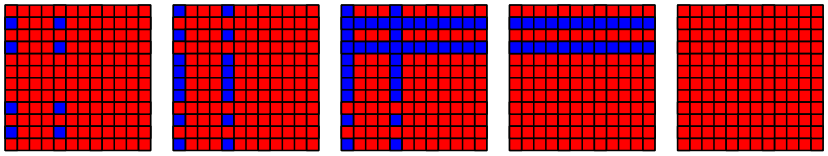
1 Have Fun

We all study theoretical computer science because we enjoy it—the beauty of simple models and algorithms, the certainty of proved theorems, the thrill of solving problems. By working on what we are passionate about, and enjoying what we do, we naturally work harder and are more productive.

I like to embrace "having fun" even further, by studying what I call *recreational computer science* [5] (by analogy to recreational mathematics, an area championed by Martin Gardner). This area is all about studying fun topics, from the perspective of serious theoretical computer science. In this way, we get to play with our work.

For example, a recent paper [7] with my father Martin Demaine, my students Sarah Eisenstat and Andrew Winslow, and my former PhD advisor Anna Lubiw, studied the Rubik's Cube from a theoretical computer science perspective. The Rubik's Cube is probably the most famous puzzle in the world, and its mathematical connections to group theory are well known, but it had not been considered from an algorithmic perspective. The classic $3 \times 3 \times 3$ cube had just been conquered, proving that the worst-case optimal algorithm solves any configuration in 20 moves [13]. But from an algorithmic perspective, the natural question is of scalability: how does the worst-case number of moves to solve an $n \times n \times n$ Cube grow with n? Standard strategies for solving Rubik's Cubes solve $O(1)$ cubies

[1] http://www.youtube.com/watch?v=ROYIVVZ5gvE



(a) Start (b) Cols. flipped (c) Rows flipped (d) Cols. flipped (e) Rows flipped

Figure 1: Simultaneously solving a grid of identically configured cubies in an $n \times n \times 1$ Rubik's Cube. (Figure from [7].) In this example, each cubie in the grid could be solved by flipping its row, then its column, then its row, then its column; but given the shared rows and columns, it is more efficient to solve them all together. The 3D Cube is similar.

in $O(1)$ moves, which suggests an upper bound of $O(n^2)$, the surface area of an $n \times n \times n$ Cube. But the right answer turns out to be $\Theta(n^2/\log n)$—using tricks from data structures, we show you can always kill $\Omega(\log n)$ birds with $O(1)$ stones.

I also love to study the computational complexity of games and puzzles. While solving actual puzzles and playing actual games is fun, the meta-puzzle of figuring out how to solve them algorithmically is even more fun—at least for theoretical computer scientists. In fact, most puzzles are NP-complete or PSPACE-complete, while most 2-player games are PSPACE-complete, EXPTIME-complete, or worse. Proving these results involves constructing portions of puzzles and games ("gadgets") with simple properties, which itself feels like solving a puzzle. The difference is that no one has solved the puzzle before, so it may not have a solution (requiring taking a different approach), but if solved, the result is publishable!

This subject was the thesis topic of my PhD student Bob Hearn, which resulted in a book [12]; see also our survey [9]. Some of my favorite results in the area are that Tetris is NP-complete [2], sliding-block puzzles are PSPACE-complete [11], and Super Mario Bros., Legend of Zelda, and other Nintendo games are all NP-hard [1].

Why study games and puzzles? These results give us a mathematical understanding of what makes games and puzzles challenging, which is probably part of what makes them fun. The results are also accessible to the general public, which helps explain our field and what powerful results it can show about the limitations of computation, and in turn helps attract young students to the field. These problems are also accessible to new researchers: everyone has a favorite game or puzzle, and with relatively little background in the field, can combine their own expertise in how to play with a clear path of searching for gadgets that establish computational complexity. But to me the primary motivation to study these problems is that they are *fun*.

2 Do More Than One Thing

I work primarily in four areas of algorithms: computational geometry (in particular folding [10]), data structures (such as cache-oblivious [4]), recreational algorithms (just described), and graph algorithms and minors (see [8]). I have worked in the first three fields since early on as a graduate student. The recreational algorithms were for fun; the other two areas were inspired by my PhD advisors Anna Lubiw (primarily in computational geometry) and Ian Munro (primarily in data structures). But instead of picking one of the areas to focus on, I chose to focus on all of them.

Why do more than one thing? The primary reason is to avoid getting stuck, which can easily happen on any one problem, for unpredictable lengths. Hedging your bets against many problems helps you smooth out this behavior, and consistently be able to solve something at all times. Beyond just solving problems, each paper or project evolves from problem to solution to write-up to submission to revision and so on. Having multiple projects on the go means that you usually have one at each stage, which lets you adapt what you work on according to your mood. When you're feeling creative, you can brainstorm problems and/or solutions—but when you're not, you can be just as productive by finishing a write-up or revising a paper. And whenever you learn a new technique from reading a paper or watching a talk, you can check whether it sheds new light on any of your problems.

I think everyone has their own capacity to how many problems and fields they can reasonably keep active. But I believe everyone should learn to make it greater than 1 (and perhaps much greater), given the efficiency gain. Plus, it's more of a good thing, so more fun.

A pleasant surprise is when one project you're working on inspires progress on another. This effect is especially cool across different areas. For example, I find geometry offers a useful perspective on traditionally nongeometric problems, like data structures; and in the reverse direction, I find many techniques from data structures are useful in many other areas such as computational geometry (and the Rubik's Cube mentioned above). By knowing both areas, I can solve problems that most others can't.

3 Collaborate

I find collaboration an extremely productive way to do research. I've written papers with 326 people (still a far cry from the famous mathematician Paul Erdős's 511 co-authors). The only papers I've written by myself are surveys, a few papers before I entered theoretical computer science (and discovered collaboration), and this article you're reading.

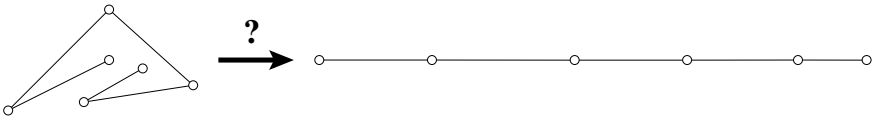


Figure 2: The Carpenter's Rule Problem: can you unfold any polygonal chain while preserving edge lengths and avoiding collisions?

I learned the highly collaborative research model from a series of computational geometry workshops held at the Bellairs Research Institute of McGill University in Holetown, Barbados. These workshops generally have no talks—just lists of open problems to work on, and progress reports on partial results. In the span of one week, a group of 10–30 researchers can solve an impressive number of problems, which lead to several papers. The style of research is constant brainstorming: suggesting ideas, finding counterexamples, drawing pictures, etc. The results that come out typically combine key ideas from many people, and avoid pitfalls observed by many more people, leading to (well-deserved) many-author papers.

My theory of collaboration is that problems decompose into several subproblems, each of which is probably easy for *someone* to solve, but also at least one of which is probably hard for each individual. If you try to solve the whole problem yourself, you'll inevitably get stuck on the subproblem that's hard for you. But if you combine the right mix of collaborators, with a mix of expertise, every subproblem can be easy. In this way, n people can solve a problem much faster than a single person, even working for n times as long.

A personal success for me was the Carpenter's Rule Theorem [3]: any 2D polygonal chain can be continuously unfolded to a straight line while keeping the edge lengths constant and avoiding collisions. Whether this was possible had been open for decades, and it quickly became my favorite open problem, so I carried it wherever I went. Eventually I found the right ideas from the right collaborators. Günter Rote had the counterintuitive idea that the unfolding could simultaneously expand all pairwise vertex distances. Bob Connelly then realized that this put the problem within the domain of his area of expertise, rigidity and tensegrity, so he could quickly identify powerful tools to help answer it. Then together we found a geometric argument that solved the problem, within just a few hours. So ultimately the problem was "easy" with the right collaborators and tools—finding them was the hard part. Without collaboration, the problem might never be solved.

Collaboration also allows us to grow our own toolset by learning each others' tools. What better way to learn a new tool or area than by solving an open problem that needs it, together with an expert who knows it? This is usually a necessary ingredient for interdisciplinary work, which is the next topic.

But let me also mention that collaboration is *fun*. It makes research a social experience. We share ideas over meals, in coffee breaks, and while traveling. These interactions build close personal bonds between researchers that makes the work all the more enjoyable.

4 Cross Disciplines

Interdisciplinary research is quite popular these days, and for good reason. Traditional fields are well-developed, and many of the basic problems have been solved or are known to be difficult. But in between these fields, there are still many new problems to be discovered, with lots of potential for exciting results. I would encourage everyone to explore Theoretical Computer Science + X, for their favorite value of X, especially those that have not yet been explored.

My favorite value of X (at the moment) is art. At first glance, art may seem like a completely different discipline from mathematics. But they are not really so different. Both math and art are all about creativity—about having a clever idea, and then carefully executing that idea within the medium. For mathematics, the medium is proof; for art, it might be sculpture, performance, or purely conceptual. Both math and art have an aesthetic of beauty: mathematicians try not just to prove a theorem, but to find an elegant proof. In this way, mathematics is an art form [6].

I got into art because my father's background is in visual arts, but also because it was helpful for my mathematical research. We built physical models to better understand the geometric structures we were studying, and those models turned out to be nice in their own right. So we decided to pursue the artistic goal of turning them into sculpture.

The most fun for us is when we can pursue the same project from both an artistic and mathematical perspective. This can be a powerful approach, because the two perspectives play off of each other, leading to inspirations on both sides—similar to collaboration or working on more than one thing. Our mathematics inspires new sculptures to build, leading to new art. Our artistic ideas often lead to questions about whether desired structures actually exist, leading to new mathematics. We believe that this makes us more productive, both as artists and as mathematicians, and leads to work on both sides that would not otherwise happen (similar to collaboration). See [6] for some examples.

You can follow these lessons however you see fit, but have fun with it!

Figure 3: Sculpture 0351 by Erik and Martin Demaine, Mi-Teintes watercolor paper, 2013, 8″ × 15″ × 16″ high. For more of our sculpture, see `http://erikdemaine.org/curved/`.

References

[1] Greg Aloupis, Erik D. Demaine, and Alan Guo. Classic Nintendo games are (NP-)hard. arXiv:1203.1895, March 2012.

[2] Ron Breukelaar, Erik D. Demaine, Susan Hohenberger, Hendrik Jan Hoogeboom, Walter A. Kosters, and David Liben-Nowell. Tetris is hard, even to approximate. *International Journal of Computational Geometry and Applications*, 14(1–2):41–68, 2004.

[3] Robert Connelly, Erik D. Demaine, and Günter Rote. Straightening polygonal arcs and convexifying polygonal cycles. *Discrete & Computational Geometry*, 30(2):205–239, September 2003.

[4] Erik D. Demaine. Cache-oblivious algorithms and data structures. In *Lecture Notes from the EEF Summer School on Massive Data Sets*. BRICS, University of Aarhus, Denmark, 2002. `http://erikdemaine.org/papers/BRICS2002/`.

[5] Erik D. Demaine. Recreational computing: Puzzles and tricks from Martin Gardner inspire math and science. *American Scientist*, 98(6):452–456, 2010.

[6] Erik D. Demaine and Martin L. Demaine. Mathematics is art. In *Proceedings of 12th Annual Conference of BRIDGES: Mathematics, Music, Art, Architecture, Culture (BRIDGES 2009)*, pages 1–10, Banff, Canada, July 2009.

[7] Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Anna Lubiw, and Andrew Winslow. Algorithms for solving Rubik's cubes. In *Proceedings of the 19th Annual European Symposium on Algorithms (ESA 2011)*, pages 689–700, September 2011. arXiv:1106.5736.

[8] Erik D. Demaine and MohammadTaghi Hajiaghayi. The bidimensionality theory and its algorithmic applications. *The Computer Journal*, 51(3):292–302, 2008.

[9] Erik D. Demaine and Robert A. Hearn. Playing games with algorithms: Algorithmic combinatorial game theory. In Michael H. Albert and Richard J. Nowakowski, editors, *Games of No Chance 3*, volume 56 of *Mathematical Sciences Research Institute Publications*, pages 3–56. Cambridge University Press, 2009.

[10] Erik D. Demaine and Joseph O'Rourke. *Geometric Folding Algorithms: Linkages, Origami, Polyhedra*. Cambridge University Press, July 2007.

[11] Robert A. Hearn and Erik D. Demaine. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1–2):72–96, October 2005.

[12] Robert A. Hearn and Erik D. Demaine. *Games, Puzzles, and Computation*. A K Peters, July 2009.

[13] Tomas Rokicki, Herbert Kociemba, Morley Davidson, and John Dethridge. God's number is 20, 2010. `http://cube20.org`.

BEATCS no 111

Convergence Issues in Congestion Games

Luca Moscardelli *

Abstract

Congestion games are a widely studied class of non-cooperative games. In fact, besides being able to model many practical settings, they constitute a framework with nice theoretical properties: Congestion games always converge to pure Nash Equilibria by means of improvement moves performed by the players, and many classes of congestion games guarantee a low price of anarchy, that is the ratio between the worst Nash Equilibrium and the social optimum. Unfortunately, the time of convergence to Nash Equilibria, even under best response moves of the players, can be very high, i.e., exponential in the number of players, and in many setting also computing a Nash equilibrium can require a high computational complexity.

Motivated by the above facts, in order to guarantee a fast convergence to Nash Equilibria, in the last decade many computer science researchers focused on special classes of congestion games (e.g., with linear or polynomial delay functions), on simplified structures of the strategy space (e.g., on symmetric games in which all players share the same set of strategies or on matroid congestion games in which the set of strategies constitutes a matroid) and on the relaxation of the notion of Nash Equilibria (e.g., exploiting the notion of ϵ-Nash Equilibria). We survey such attempts that, however, only in some very specific cases have led to satisfactory results on the speed of convergence to Nash Equilibria.

If we relax the constraint of reaching a Nash Equilibrium, and our goal becomes that of reaching states approximating the social optimum by a "low" factor, i.e., a factor being order of the price of anarchy, significantly better results on the speed of convergence under best response dynamics can be achieved.

Interestingly, in the more general asymmetric setting, fairness among players influences the speed of convergence. For instance, considering the fundamental class of linear congestion games, if each player is allowed to play at least once and at most β times every T best responses, states with approximation ratio $O(\beta)$ times the price of anarchy are reached after

*Department of Economic Studies, University of Chieti-Pescara, Viale Pindaro 42, 65127 Pescara, Italy. Email: luca.moscardelli@unich.it

$T\lceil \log \log n \rceil$ best responses, and such a bound is essentially tight also after exponentially many ones. It is worth noticing that the structure of the game implicitly affects its performances in terms of convergence speed: In particular, in the symmetric setting the game always converges to an efficient state after a polynomial number of best responses, regardless of the frequency each player moves with. Most of these results extend to polynomial and weighted congestion games.

1 Introduction

Congestion games are used for modeling non-cooperative systems in which a set of resources are shared among a set of selfish players. In a congestion game we have a set of m resources and a set of n players. Each player's strategy consists of a subset of resources. The delay of a particular resource e depends on its congestion, corresponding to the number of players choosing e, and the cost of each player i is the sum of the delays associated with the resources selected by i. A congestion game is called symmetric if all players share the same strategy set. A state of the game is any combination of strategies for the players and its social cost, defined as the sum of the players' costs, denotes its quality from a global perspective. The social optimum denotes the minimum possible social cost among all the states of the game.

In the weighed version of the game, each player is assigned with a non-negative weight and the congestion of each resource is the sum of the weights of the player using that resource.

In this work we focus on a fundamental classes of congestion games, having particular delay functions for their resources. In particular, we consider polynomial delay function with maximum degree d. We are mainly interested in studying and analyzing the time of convergence to solutions being a good approximation of the optimum.

The paper is organized as follows: In the next section we provide the basic notation and definitions. Section 3 surveys the state of the art on convergence issues in congestion games. Section 4 is devoted to the study of the quality of the solution reached after a move performed by each player, while in Section 5 it is studied the speed of convergence to good solutions. Finally, Section 6 gives some conclusive remarks and lists some interesting open problems.

2 Model and Definitions

A *weighted congestion game* $\mathcal{G} = (N, E, (w_i)_{i\in N}, (\Sigma_i)_{i\in N}, (f_e)_{e\in E}, (c_i)_{i\in N})$ is a non-cooperative strategic game characterized by the existence of a set E of resources

to be shared by the players in $N = \{1, \ldots, n\}$.

Each player i has a weighted demand $w_i \in \mathbb{R}^+$. Since it is always possible to suitably scale all the weights, without loss of generality, we assume that $w_i \geq 1$ for any $i \in N$; furthermore, we denote by W the sum of the weights of all players, i.e. $W = \sum_{i \in N} w_i$. Moreover, let us denote by w_{max} the maximum weight. Σ_i is the strategy space of player i, and any strategy $s_i \in \Sigma_i$ of player i is a subset of resources, i.e. $\Sigma_i \subseteq 2^E$. Given a *strategy profile* (also called *state*) $S = (s_1, \ldots, s_n)$ and a resource e, we define the congestion $\theta_e(S)$ on resource e by $\theta_e(S) = \sum_{i \in N | e \in s_i} w_i$. A delay function $f_e : \mathbb{R}^+ \mapsto \mathbb{R}^+$ associates to resource e a delay depending on its congestion, so that the cost of player i for the pure strategy s_i is given by the weighted sum of the delays associated with resources in s_i, i.e. $c_i(S) = \sum_{e \in s_i} w_i f_e(\theta_e(S))$.

In this paper we will focus on congestion games with *polynomial* delay functions with maximum degree d and non-negative coefficients, that is for every resource $e \in E$ the delay function is of the form $f_e(x) = \sum_{j=1}^{d} a_{e,j} x^j$ with $a_{e,j} \geq 0$ for all $j = 0, \ldots, d$. We call a congestion game *linear* if $d = 1$, that is $f_e(x) = a_e x + b_e$ for every resource $e \in E$, with $a_e, b_e \geq 0$.

We call a congestion game *unweighted* whenever, for each player $i \in N$, $w_i = 1$. In this case, the congestion $\theta_e(S)$ on resource e is also denoted by $n_e(S)$ and it is equal to $|\{i \in N | e \in s_i\}|$, i.e., $n_e(S) = \theta_e(S) = |\{i \in N | e \in s_i\}|$.

Given the strategy profile $S = (s_1, \ldots, s_n)$, the social cost $C(S)$ of a given state S is defined as the sum of all the players' costs, i.e., $C(S) = \sum_{i \in N} c_i(S) = \sum_{e \in E} \theta_e(S) f_e(\theta_e(S))$. An optimal strategy profile $S^* = (s_1^*, \ldots, s_n^*)$ is one having minimum social cost; we denote $C(S^*)$ by Opt. The *approximation ratio* of state S is given by the ratio between the social cost of S and the social optimum, i.e., $\frac{C(S)}{\text{Opt}}$. Moreover, given the strategy profile $S = (s_1, s_2, \ldots, s_n)$ and a strategy $s_i' \in \Sigma_i$, let $(S_{-i}, s_i') = (s_1, s_2, \ldots, s_{i-1}, s_i', s_{i+1}, \ldots, s_n)$, i.e., the strategy profile obtained from S if player i changes its strategy from s_i to s_i'.

A notable special class of congestion games is that of *Network Congestion Games*, in which we are given a communication graph G whose edges are the resources of the congestion game; each player is associated to a source and a destination node of G and her strategies are given by all the resources corresponding to the edges of the simple paths connecting her source with her destination in G.

Finally, a *symmetric* game is a game in which all the players share the same set of strategies.

Each player acts selfishly and aims at choosing the strategy decreasing its cost, given the strategy choices of other players. For a strategy profile $S = (s_1, s_2, \ldots, s_n)$, an *improvement move* of player i is a strategy s_i' such that $c_i(S_{-i}, s_i') < c_i(S)$. A *best response* of player i in S is a strategy $s_i^b \in \Sigma_i$ yielding the minimum possible cost, given the strategy choices of the other players, i.e., $c_i(S_{-i}, s_i^b) \leq c_i(S_{-i}, s_i')$ for any other strategy $s_i' \in \Sigma_i$. Moreover, if no $s_i' \in \Sigma_i$ is

such that $c_i(S_{-i}, s'_i) < c_i(S)$, the best response of i in S is s_i.

A state S is a *Nash equilibrium* if for every player $i \in N$ and all strategy $s'_i \in \Sigma_i$, $c_i(S) \leq c_i(S_{-i}, s'_i)$, i.e., no player in N can improve her individual cost by unilaterally changing her strategy. The *price of anarchy* (PoA) is the ratio $C(S)/\textsc{Opt}$, where S is the worst Nash equilibrium, i.e. it is the approximation ratio of the Nash equilibrium having maximum social cost. Notice that Nash equilibria correspond to sinks in the Nash dynamics graph.

By extending the Nash dynamics and the Nash equilibrium concepts, it is possible to define their approximated versions as follows: Given a strategy profile $S = (s_1, s_2, \ldots, s_n)$, an *$\epsilon$-improvement move* of player i is a strategy s'_i such that $c_i(S_{-i}, s'_i) < (1 - \epsilon)c_i(S)$. S is an ϵ-Nash equilibrium if no player has an ϵ-improvement move. An *ϵ-better response* of player i in S is either an ϵ-improvement move, if it is available, or the current strategy s_i, otherwise. An *ϵ-better response dynamics* any sequence of ϵ-better responses.

In Section 3 we will briefly survey the results achieved in the literature with respect to the speed of convergence to Nash equilibria and ϵ-Nash equilibria. Then, in the remainder of the paper, we will study the approximation ratio of states reached after sequences of best responses, that we call a *best response dynamics*.

The selfish behavior of players performing best responses can be modelled by the *(Best Response) Nash Dynamics Graph*. Formally the Nash Dynamics Graph associated to a congestion game $\mathcal{G}$ is a directed graph $\mathcal{B} = (V, A)$ where each vertex in V corresponds to a strategy profile and there is an edge $(S, S') \in A$ with label i, where $S' = (S_{-i}, s'_i)$ and $s'_i \in \Sigma_i$, if and only if both the following conditions are met: (*i*) s'_i is a best response of i in S; (*ii*) if $S \neq S'$, s'_i is also an improvement move of i in S. Observe that $\mathcal{B}$ may contain loops, corresponding to best response in which a player maintains her current strategy. A *best response walk* is a directed walk in $\mathcal{B}$.

To this aim, we must consider dynamics in which each player performs a best response at least once in a given number T of best responses, otherwise one or more players could be "locked out" for arbitrarily long and we could not expect to bound the social cost of the state reached at the end of the dynamics:

Definition 1 (T-Minimum Liveness Condition)**.** *Given any $T \geq n$, a best response dynamics satisfies the T-Minimum Liveness Condition if each player performs at least a best response every T consecutive responses.*

We consider the following notions of best response walks, that are a refinement of the ones introduced in [10, 21]; notice that all of them satisfy the Minimum Liveness Condition above stated.

1-**round walk**: it's a best response walk $R = \left(\left(S_R^0, S_R^1\right), \left(S_R^1, S_R^2\right), \ldots, \left(S_R^i, S_R^{i+1}\right), \ldots, \left(S_R^{n-1}, S_R^n\right)\right)$ in $\mathcal{B}$ of length n, where the edge (S_R^i, S_R^{i+1})

has label $\pi_R(i)$ for every $0 \leq i \leq n-1$, i.e. $\pi_R(i)$ is the player performing the i-th best response of the walk. π_R is such that every player performs exactly a best response in R. S_R^0 is said the *initial* state of R and S_R^n its *final* state. For simplicity we denote R by a sequence of states, i.e. $R = \left(S_R^0, \ldots, S_R^n\right)$.

(ℓ,β)**-bounded covering walk**: it's a best response walk $R = \left(\left(S_R^0, S_R^1\right), \left(S_R^1, S_R^2\right), \ \ldots, \left(S_R^i, S_R^{i+1}\right), \ldots, \left(S_R^{\ell-1}, S_R^\ell\right)\right)$ in $\mathcal{B}$ of length ℓ, where the edge (S_R^i, S_R^{i+1}) has label $\pi_R(i)$ for every $0 \leq i \leq \ell - 1$, i.e. $\pi_R(i)$ is the player performing the i-th best response of the walk. π_R is such that every player performs at least a best response and at most β best responses in R. S_R^0 is said the *initial* state of R and S_R^ℓ its *final* state. For simplicity we denote R by a sequence of states, i.e. $R = \left(S_R^0, \ldots, S_R^\ell\right)$. For any $i = 1, \ldots, n$, the last best response performed by player i in R is the $\text{last}_R(i)$-th best response of R, leading from state $S^{\text{last}_R(i)-1}$ to state $S^{\text{last}_R(i)}$.

Notice that β is a sort of (un)fairness index: If β is constant, it means that every player plays at most a constant number of times in each T-covering and therefore the dynamics can be considered *fair*.

Definition 2 ((T,β)-Fairness Condition)**.** *Given any $T \geq n$, a best response dynamics satisfies the (T,β)-Fairness Condition if it ca be decomposed in k (ℓ,β)-bounded covering walks such that $\ell \leq T$ for all such coverings.*

When clear from the context, we will drop the index R from the notation, writing S^i, π and last(i) instead of S_R^i, π_R and $last(i)_R$, respectively.

We will consider the following two scenarios:

1. all walks are assumed to start from an arbitrary initial state;
2. all walks are assumed to start from the "empty" state in which no receiver has still selected its communication path from the source; in order to include such a situation in the above framework we include an additional special empty path $\emptyset$ in the strategies sets P_i, assuming that the only possible best response moves from $\emptyset$ are the ones selecting a path of minimum payment actually connecting the source to the receiver; all the notation and definitions are trivially extended accordingly.

3 Related Work and Our Contribution

On the one hand, Rosenthal [23] has shown, by a potential function argument, that for unweighted congestion games the natural decentralized mechanism known as

Nash dynamics, in which at each step some player performs an improvement step switching her strategy to a better alternative, is guaranteed to converge to a pure Nash equilibrium [22], i.e. a fixed point in which no player can perform an improvement step (note that in a Nash dynamics players play their improvement steps sequentially, and not in parallel). On the other hand, weighted congestion games do not necessarily admit a Nash equilibrium [20] unless specific settings are considered (linear delay functions [18], singleton congestion games [17], matroid congestion games [2]). With a little abuse of notation, also in the context of weighted congestion games that do not admit a Nash equilibrium, we call Nash dynamics the dynamics in which each player aims at minimizing its current cost, given the strategic choices of all the other players. More formally, an *exact* real-valued potential function Φ defined on the set of states of the game satisfies the property that for each player i and each strategy $s'_i \in \Sigma_i$ of i in S, it holds that $c_i(S_{-i}, s'_i) - c_i(S) = \Phi(S_{-i}, s'_i) - \Phi(S)$. Unweighted congestion games admit (see [23]) the (exact) potential function

$$\Phi(S) = \sum_{e \in E} \sum_{j=1}^{n_e(S)} f_e(j).$$

It is worth noticing that in unweighted congestion games with polynomial delays with maximum degree d, for any state S, it holds $\Phi(S) \leq C(S) \leq (d+1)\Phi(S)$. Unfortunately, in general, weighted congestion games do not admit a potential function; nevertheless, it is possible to show that weighted congestion games with linear delays do (see [18]), and such a potential function is

$$\Phi(S) = \frac{1}{2}\left(\sum_{e \in E} \theta_e(S) f_e(\theta_e(S)) + \sum_{i=1}^{n} \sum_{e \in s_i} w_i f_e(w_i)\right).$$

It is worth noticing that in weighted linear congestion games, for any state S, it holds $\Phi(S) \leq C(S) \leq 2\Phi(S)$.

Even in the unweighted setting in which Nash equilibria are guaranteed to exist, Fabrikant et al. [12] show that such dynamics may require a number of steps exponential in the number of players n in order to reach such an equilibrium. Their analysis relates congestion games to local search problems by showing that it is PLS-complete [11] to compute a Nash equilibrium for general unweighted congestion games. Moreover from their completeness proof and from previous results about local search problems, it follows that there exist congestion games with initial states such that any improvement sequence starting from these states needs an exponential number of steps in order to reach a Nash equilibrium. More recently, Ackermann et al. [1] show that the previous negative result holds even in the restricted case of linear unweighted congestion games. On the positive side,

Fabrikant et al. [12], by exploiting a reduction to the Min-cost flow problem, show that it is possible to Compute a Nash equilibrium in symmetric Network congestion games with non-decreasing delay function in polynomial time. Such a result can be also extended to network congestion games in which all the players share the same source (multicast games). Unfortunately, such a result does not imply a fast convergence to Nash equilibria, as Ackermann et al. [1] show the existence of network congestion games in which any sequence of improvement moves is exponentially long in n. It is possible to obtain a polynomial convergence by restricting the combinatorial structure of congestion games, in particular by imposing that the strategy set of each player is the basis of a matroid; note that load balancing games, in which every strategy is composed by only one resource, belong to such a class of congestion games.

The negative results on computing equilibria in congestion games have lead to the development of the concept of ϵ-Nash equilibrium, in which no player can decrease its cost by a factor of more than ϵ. Unfortunately, as shown by Skopalik and Vöcking [24], also the problem of finding an ϵ-Nash equilibrium in (unweighted) congestion games is PLS-complete for any ϵ, though, under some restrictions on the delay functions, Chien and Sinclair [8] proved, for some classes of delay functions including the polynomial ones, that in symmetric congestion games the convergence to ϵ-Nash equilibrium is polynomial in the description of the game and the minimal number of steps within which each player has a chance to move.

In order to measure the degradation of social welfare due to the selfish behavior of the players, Koutsoupias and Papadimitriou [19] defined the price of anarchy as the worst-case ratio between the social cost in a Nash equilibrium and that of a social optimum. The price of anarchy for congestion games has been investigated by Awerbuch et al. [4] and Christodoulou and Koutsoupias [9]. They both proved that the price of anarchy for congestion games with linear delays is $5/2$.

The performances of Nash equilibria in unweighted and weighted congestion games with polynomial delay functions have been investigated by Aland et al. [3], who have provided a tight price of anarchy both for the unweighted case and the weighted one. In particular they have improved the previous results due to Awerbuch et al. [4] (for the weighted case) and Christodoulou and Koutsoupias [9] (for the unweighted case).

Since negative results tend to dominate the issues relative to the complexity of computing equilibria, also considering that in the more general setting of weighted congestion games Nash equilibria may not be guaranteed to exist, another natural arising question is whether efficient states (with a social cost constant or at least comparable to the one of any Nash equilibrium) can be reached by best response moves in a reasonable amount of time (e.g., [5, 10, 13, 14]). We measure the efficiency of a state by the ratio among its cost and the optimal one, i.e., its ap-

proximation ratio. We generally say that a state is efficient when its approximation ratio is within a constant factor from the price of anarchy.

Awerbuch et al. [5] have proved that for some classes of delay functions (including the polynomial delay functions), sequences of moves reducing the cost of each player by at least a factor of ϵ, converge to efficient states in a number of moves polynomial in $1/\epsilon$ and the number of players, under the minimal liveness condition that every player moves at least once every polynomial number of moves. Under the same liveness condition, they also proved that exact best response dynamics can guarantee the convergence to efficient states only after an exponential number of best responses [5].

Nevertheless, we are able to show that best response dynamics in congestion games with polynomial delays can actually fast converge to good solutions, provided that the dynamics obeys some mild constraint on the order of the moving players.

Our Contribution. In this paper we first provide some preliminary result [6] on the approximation ratio of the solutions achieved after a one-round walk in linear congestion games. For one-round walks starting from the empty strategy profile, we close the existing gap between the upper bound of $2 + \sqrt{5} \approx 4.24$ given in [10] and the lower bound of 4 derived in [7] by providing a family of instances yielding lower bounds approaching $2 + \sqrt{5}$ as the number of players goes to infinity. The construction and the analysis of these instances require non-trivial arguments.

Then, we focus on the more general setting in which the dynamics start from a generic state.

We show [13, 14] that, for weighted congestion games with polynomial delays, the approximation ratio achieved after a sequence of k $O(1)$-bounded covering walks is $O\left(W^{d\left(\frac{d}{d+1}\right)^{k-1}}\right)$ and $\Omega\left(\frac{W^{d\left(\frac{d}{d+1}\right)^{k-1}}}{k}\right)$ (which is asymptotically matching for constant values of k), where W is the sum of the players' weights. As a consequence, we prove that, for any given d, $\Theta(\log\log W)$ $O(1)$-bounded covering walks are necessary and sufficient to achieve a constant factor approximate solution.

Finally, we completely characterize [15] how the frequency with which each player participates in the game dynamics affects the possibility of reaching efficient states. In particular, we close the most important open problem left open by [5] and [13, 14] for linear congestion games. On the one hand, in [5] it is shown that, even after an exponential number of best responses, states with a very high approximation ratio, namely $\Omega\left(\frac{\sqrt{n}}{\log n}\right)$, can be reached. On the other hand, in [13, 14] it is shown that, under the minimal liveness condition in which every player moves at least once every T steps, if players perform best responses such that each player is allowed to play at most $\beta = O(1)$ times any T steps (notice

that $\beta = O(1)$ implies $T = O(n)$), after $T\lceil \log \log n \rceil$ best responses a state with a constant factor approximation ratio is reached. The more β increases, the less the dynamics is fair with respect to the chance every player has of performing a best response: β measures the degree of unfairness of the dynamics. The important left open question was that of determining the maximum order of β needed to obtain fast convergence to efficient states: We answer this question by proving that, after $T\lceil \log \log W \rceil$ best responses, the dynamics reaches states with an approximation ratio of $O(\beta)$. Such a result is essentially tight since we are also able to show that, for any $\epsilon > 0$, there exist congestion games for which, even for an exponential number of best responses, states with an approximation ratio of $\Omega(\beta^{1-\epsilon})$ are obtained. Therefore, β constant as assumed in [13, 14] is not only sufficient, but also necessary in order to reach efficient states after a polynomial number of best responses. Furthermore, in the special case of symmetric congestion games with linear delays, we show that the unfairness in best response dynamics does not affect the fast convergence to efficient states; namely, we prove that, for any β, after $T\lceil \log \log W \rceil$ best responses efficient states are always reached.

4 One-Round Analysis [6]

Christodoulou et al. [10] proved that for any linear congestion game and 1-round walk $(S^0, S^1, \ldots, S^n)$ with $S^0 = \emptyset$, it holds $C(S^n) \leq (2 + \sqrt{5})\textsc{Opt} \approx 4.24\textsc{Opt}$. Surprisingly enough, the best known lower bound is the one derived by Caragiannis at al. [7] for the restricted case of load balancing on identical servers, which poses $C(S^n) \geq 4\textsc{Opt} - \epsilon$, for any $\epsilon > 0$. We close this gap by proving that, for any $\epsilon > 0$, there always exist a linear congestion game and a 1-round walk $(S^0, S^1, \ldots, S^n)$, with $S^0 = \emptyset$, such that $C(S^n) \geq (2 + \sqrt{5} - \epsilon)\textsc{Opt}$.

Given three positive integers n, k and o, with $n \geq 2k + o - 1$ and $k \geq 2o$, we define the game $\mathcal{G}_{n,k,o}$ in which there are n players, $m = n + 1$ resources and each player $i \in [n]$ possesses exactly two strategies s_i and s'_i defined according to the following scheme.

- $s_i = \{e_i\}$ and $s'_i = \{e_{i+1}\} \cup \bigcup_{j=k+1}^{k+i} \{e_j\}$, for any $i \in [k-1]$;
- $s_k = \{e_k\}$ and $s'_k = \bigcup_{j=k+1}^{2k} \{e_j\}$;
- $s_i = \bigcup_{j=k+1}^{i} \{e_j\}$ and $s'_i = \bigcup_{j=i+1}^{k+i} \{e_j\}$, for any $k+1 \leq i \leq k+o$;

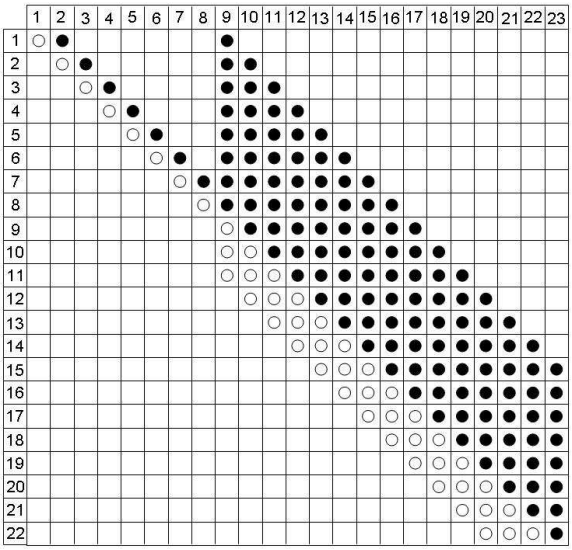


Figure 1: The set of strategies available to each player in the game $\mathcal{G}_{22,8,3}$. Rows are associated with players, while columns with resources. White and black circles represent the first and the second strategy, respectively.

- $s_i = \bigcup_{j=i-o+1}^{i} \{e_j\}$ and $s'_i = \bigcup_{j=i+1}^{\min\{k+i,m\}} \{e_j\}$, for any $k + o + 1 \leq i \leq n$.

A small example in which $n = 22$, $k = 8$ and $o = 3$ is shown in Figure 1.

For any $j \in [m]$ we associate the linear latency function $f_j(x) = a_j \cdot x$ with resource e_j, where each a_j is obtained as a solution of the following system of linear equations.

$$A = \begin{cases} eq_1 \\ eq_2 \\ \dots \\ eq_n \end{cases}$$

where each eq_i is defined as follows:

- $a_1 - a_2 - a_{k+1} = 0$,
- $2a_i - a_{i+1} - \sum_{j=k+1}^{k+i} \Big((k + i - j + 1)a_j\Big) = 0 \;\; \forall\, i = 2, \dots, k-1$,

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1	1	-1							-1														
2		2	-1						-2	-1													
3			2	-1					-3	-2	-1												
4				2	-1				-4	-3	-2	-1											
5					2	-1			-5	-4	-3	-2	-1										
6						2	-1		-6	-5	-4	-3	-2	-1									
7							2	-1	-7	-6	-5	-4	-3	-2	-1								
8								2	-8	-7	-6	-5	-4	-3	-2	-1							
9									9	-8	-7	-6	-5	-4	-3	-2	-1						
10									9	9	-8	-7	-6	-5	-4	-3	-2	-1					
11									9	9	9	-8	-7	-6	-5	-4	-3	-2	-1				
12										9	9	9	-8	-7	-6	-5	-4	-3	-2	-1			
13											9	9	9	-8	-7	-6	-5	-4	-3	-2	-1		
14												9	9	9	-8	-7	-6	-5	-4	-3	-2	-1	
15													9	9	9	-8	-7	-6	-5	-4	-3	-2	-1
16														9	9	9	-8	-7	-6	-5	-4	-3	-2
17															9	9	9	-8	-7	-6	-5	-4	-3
18																9	9	9	-8	-7	-6	-5	-4
19																	9	9	9	-8	-7	-6	-5
20																		9	9	9	-8	-7	-6
21																			9	9	9	-8	-7
22																				9	9	9	-8

Figure 2: The coefficient matrix B generated by the game $\mathcal{G}_{22,8,3}$.

- $2a_k - \sum_{j=k+1}^{2k} \Big((2k-j+1)a_j\Big) = 0,$
- $(k+1)\sum_{j=k+1}^{i} a_j - \sum_{j=i+1}^{m} \Big((k+i-j+1)a_j\Big) = 0 \;\; \forall\, i \in \{k+1,\ldots,k+o\},$
- $(k+1)\sum_{j=i-o+1}^{i} a_j - \sum_{j=i+1}^{\min\{k+i,m\}} \Big((k+i-j+1)a_j\Big) = 0 \;\; \forall\, i \in \{k+o+1,\ldots,n\}.$

Note that the definition of each equality is such that, for any $i \in [n]$, both strategies are equivalent for player i, provided all players $j < i$ have chosen s'_j and all players $j > i$ have not entered the game yet.

Let B be the $n \times m$ coefficient matrix defining system A. The matrix B generated by the game $\mathcal{G}_{22,8,3}$ is shown in Figure 2.

Let $a = (a_1, \ldots, a_m)^T$. In order for our instance to be well defined, we need to prove that there exists at least a strictly positive solution to the homogeneous system $Ba = 0$.

Lemma 1. *The system of linear equations $Ba = 0$ admits a strictly positive solution.*

We claim that the strategy profile in which all players choose the second of their strategies is a possible outcome for a 1-round walk starting from the empty strategy profile.

Lemma 2. *For any game $\mathcal{G}_{n,k,o}$, there exists a 1-round walk $(S^0, S^1, \ldots, S^n)$ such that $S^0 = \emptyset$ and $S^n = (s'_1, \ldots, s'_n)$.*

Proof. The claim is a direct consequence of the definition of system A. □

For our purposes, we do not have to explicitly solve system A, but only need to prove some properties characterizing its set of solutions. We do this in the next two lemmas.

Lemma 3. *In any solution of system A it holds $a_1 \leq 4 \sum_{j=k+1}^{2k} a_j$.*

Lemma 4. *In any solution of system A it holds*

$$(k+1) \sum_{i=m-o+1}^{m} ((i-m+o)a_i) \leq \frac{k^3}{n-2k-o+1} \sum_{i=k+1}^{m-o} a_i.$$

We can now prove our main result.

Theorem 1. *For any $\epsilon > 0$, there exist a linear congestion game $\mathcal{G}_{n,k,o}$ and a 1-round walk $(S^0, S^1, \ldots, S^n)$, with $S^0 = \emptyset$, such that $C(S^n) \geq (2 + \sqrt{5} - \epsilon)$Opt.*

Proof. For a fixed integer $n \gg 0$, set $k = \lfloor \sqrt[4]{n} \rfloor$ and $o = \lfloor \frac{3-\sqrt{5}}{2} k \rfloor$. Note that, for a sufficiently big n, these values are consistent with the definition of $\mathcal{G}_{n,k,o}$ since $n \geq 2k + o - 1$ and $k \geq 2o$.

Consider the sum of all the equations defining system A together with the dummy one $a_1 = a_1$. We obtain the equation

$$\sum_{i=1}^{k} 2a_i + (k+1)o \sum_{i=k+1}^{m} a_i - (k+1) \sum_{i=m-o+1}^{m} ((i-m+o)a_i) = \sum_{i=1}^{k} a_i + \frac{k(k+1)}{2} \sum_{i=k+1}^{m} a_i$$

which yields

$$\sum_{i=1}^{k} a_i = (k+1)\left(\frac{k}{2} - o\right) \sum_{i=k+1}^{m} a_i + (k+1) \sum_{i=m-o+1}^{m} ((i-m+o)a_i). \tag{1}$$

Let $S^* = (s_1 \ldots, s_n)$ be the strategy profile in which all players choose the first of their strategies. Because of Lemma 2, we have that there exists a 1-round walk

$(S^0, S^1, \ldots, S^n)$ such that $S^0 = \emptyset$ and $S^n = (s'_1, \ldots, s'_n)$. By comparing the social costs of S^n and S^*, we obtain

$$\frac{C(S^n)}{\text{Opt}} \geq \frac{C(S^n)}{C(S^*)} \geq \frac{\sum_{i=2}^{k} a_i + k^2 \sum_{i=k+1}^{m} a_i}{\sum_{i=1}^{k} a_i + o^2 \sum_{i=k+1}^{m} a_i},$$

where we have exploited the fact that $\text{Opt} \leq C(S^*) \leq \sum_{i=1}^{k} a_i + o^2 \sum_{i=k+1}^{m} a_i$. By using Equality 1, we get

$$\frac{C(S^n)}{\text{Opt}} \geq \frac{\sum_{i=2}^{k} a_i + k^2 \sum_{i=k+1}^{m} a_i}{\sum_{i=1}^{k} a_i + o^2 \sum_{i=k+1}^{m} a_i} =$$

$$\frac{\left((k+1)\left(\frac{k}{2}-o\right)+k^2\right)\sum_{i=k+1}^{m} a_i + (k+1)\sum_{i=m-o+1}^{m}((i-m+o)a_i) - a_1}{\left((k+1)\left(\frac{k}{2}-o\right)+o^2\right)\sum_{i=k+1}^{m} a_i + (k+1)\sum_{i=m-o+1}^{m}((i-m+o)a_i)} \geq$$

$$\frac{\left((k+1)\left(\frac{k}{2}-o\right)+k^2+\frac{k^3}{n-2k-o+1}-4\right)\sum_{i=k+1}^{m} a_i}{\left((k+1)\left(\frac{k}{2}-o\right)+o^2+\frac{k^3}{n-2k-o+1}\right)\sum_{i=k+1}^{m} a_i},$$

where, in the last inequality, we have used Lemmas 3 and 4 together with the fact that for any four positive numbers α, β, γ and δ such that $\alpha \geq \beta$ and $\gamma \geq \delta$, it holds $\frac{\alpha+\delta}{\beta+\delta} \geq \frac{\alpha+\gamma}{\beta+\gamma}$.

For n going to infinity, by considering only the dominant terms, we obtain $\lim_{k\to\infty} \frac{C(S^n)}{\text{Opt}} \geq \lim_{k\to\infty} \frac{k\left(\frac{k}{2}-o\right)+k^2}{k\left(\frac{k}{2}-o\right)+o^2} = \lim_{k\to\infty} \frac{\frac{\sqrt{5}-2}{2}k^2+k^2}{\frac{\sqrt{5}-2}{2}k^2+\frac{7-3\sqrt{5}}{2}k^2} = \lim_{k\to\infty} \frac{\frac{\sqrt{5}}{2}k^2}{\frac{5-2\sqrt{5}}{2}k^2} = \frac{\sqrt{5}}{5-2\sqrt{5}} = 2+\sqrt{5}$, which implies the claim. □

Some numerical results, obtained on particular games, are shown in Figure 3. It is possible to appreciate there that the value of k may be chosen much higher than the bound $\lfloor \sqrt[4]{n} \rfloor$ fixed in the proof of Theorem 1. This is due to the fact that, for the sake of simplicity, the bound proved in Lemma 4 is really far from being tight.

n	k	o	$\frac{SUM(S^n)}{SUM(S^*)}$
70	8	3	4.001152
100	8	3	4.012482
500	80	30	4.185590
700	80	30	4.208719
1000	100	38	4.216734
1500	100	38	4.220854
2000	200	76	4.224342
3000	300	114	4.226854

Figure 3: Lower bounds on the approximation ratio of the solution achieved after a 1-round walk starting from the empty strategy profile in the games $\mathcal{G}_{n,k,o}$ for some particular values of n, k and o.

5 Convergence to Good Solutions

In this section we provide upper and lower bounds to the approximation ratio of the states reached after a dynamics satisfying the (T,β)-Minimum Liveness Condition, starting from an arbitrary state and composed by a number of best responses polynomial in n.

We first provide (in Subsection 5.1) an upper bound to the the social cost of the state achieved after a best response dynamics satisfying the (T,β)-Fairness condition with $\beta = O(1)$ and starting from an arbitrary state, and then (in Subsection 5.2) we deal with the case of general values of β.

All the results hold for congestion games having polynomial delay functions with non-negative coefficients and maximum degree d, i.e. for every $e \in E$, $f_e(x) = \sum_{j=0}^{d} a_{e,j}x^j$ with $a_{e,j} \geq 0$ for all $j = 0,\ldots,d$. Without loss of generality, we can assume that for every $e \in E$, $f_e(x) = x^{d_e}$ with $0 \leq d_e \leq d$.

In fact, given a congestion game $\mathcal{G}$ having polynomial delays with coefficients being non-negative integers and maximum degree d, it is possible to obtain an equivalent congestion game $\mathcal{G}'$ (i.e, a congestion game having a Nash Dynamics Graph isomorphic to the one of $\mathcal{G}$ and in which any strategy profile S' corresponding to the strategy profile S of $\mathcal{G}$ is such that for every player $i \in N$, $c_i(S) = c_i(S')$), having the same set of players and delay functions of the form $f_e(x) = x^{d_e}$ in the following way. For each resource e in $\mathcal{G}$, we include in $\mathcal{G}'$ a set $A_{e,j}$ of $a_{e,j}$ resources for $j = 0,\ldots,d$ with delay function $f_e(x) = x^j$; moreover, given any strategy set $s_i \in \Sigma_i$ in $\mathcal{G}$, $i = 1,\ldots,n$, we build a corresponding strategy set $s'_i \in \Sigma'_i$ (in $\mathcal{G}'$) by including in s'_i, for each $e \in s_i$, all the resources in the sets $A_e = \cup_{j=0}^{d} A_{e,j}$.

If the coefficients $a_{e,j}$ are rationals (not all being integers) we can perform a similar reduction by exploiting a simple scaling argument.

Finally, if the coefficients are real numbers (not all being rationals), we can obtain from a congestion game $\mathcal{G}$ a new game $\mathcal{G}'$ with coefficients being rationals by approximating the real numbers with a sufficiently high precision so that any best response of $\mathcal{G}$ corresponds to a "quasi"-best response of $\mathcal{G}'$, up to an additive ϵ depending on the precision of the considered approximation.

Moreover, we can also assume without loss of generality that all the player weights are at least 1, by suitably scaling all the weights if the considered game does not satisfy such a property.

Therefore, in the sequel of this section we will consider, for every $e \in E$, $f_e(x) = x^{d_e}$, with $d = \max_e d_e$ and $w_i \geq 1$ for every $i = 1, \ldots, n$.

5.1 $\beta = O(1)$ [14]

The following claim, whose proof can be found in [16] (Lemma 3.6 of [16], with $\gamma = \frac{1}{2}$) will be useful in the sequel.

Claim 1 ([16])**.** *For every pair of reals $x, y \geq 0$ and every integer $d \geq 1$, it holds $x^d \geq \frac{1}{2^{d-1}}(x+y)^d - y^d$.*

Let $R = \left(S^0, \ldots, S^\ell\right)$ be a (T,β)-bounded covering walk. Given the optimal strategy profile S^*, since the i-th moving player $\pi(i)$ before moving can always select the strategy she would use in S^*, $c_{\pi(i)}(S^i)$ (that is the player's cost immediately after her best response) can be suitably upper bounded by $\sum_{e \in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e}$. In order to state our results we define the following function

$$\rho(R) = \sum_{i=1}^{\ell} \sum_{e \in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e},$$

which, by the same argument explained earlier, clearly represents an upper bound to $\sum_{i=1}^{\ell} c_{\pi(i)}(S^i)$.

Lemmas 5 and 6 provide a lower and an upper bound to $\rho(R)$, respectively. From such Lemmas, we can easily derive the approximation achieved after a (T,β)-bounded covering walk.

Lemma 5. *For any $\beta \geq 1$, given a (T,β)-bounded covering walk R ending in S^ℓ, it holds $\rho(R) \geq \frac{C(S^\ell)}{(d+1)}$.*

Proof. Since the players perform best responses, inequality (2) below holds. In order to justify inequality 3, let us consider a resource e. Recall that the cost $c_{\pi(i)}(S^i)$ incurred by a player $\pi(i)$ on e is $w_{\pi(i)} f_e(\theta_e(S^i))$; since f_e is a non decreasing function and $\theta_e(S^i)$ is given by the sum of the players already on e at S^{i-1} plus $w_{\pi(i)}$, the sum of all the cost that players using e incur on e can be lower bounded by

considering the resource used by many players each having infinitesimal weight; thus, the summation $\sum_{e\in E}\sum_{i\in\{1,\ldots,\ell\}|e\in s_{\pi(i)}} w_{\pi(i)}\theta_e^{d_e}(S^i)$ can be replaced by the integral $\int_{x=0}^{\theta_e(S^\ell)} x^{d_e}dx$. Therefore, by recalling the definition of $\rho(R)$,

$$\begin{aligned}
\rho(R) &= \sum_{i=1}^{\ell}\sum_{e\in s^*_{\pi(i)}} w_{\pi(i)}\left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e} \\
&\geq \sum_{i=1}^{\ell} c_{\pi(i)}(S^i) \qquad (2)\\
&= \sum_{e\in E}\sum_{\substack{i\in\{1,\ldots,\ell\}|\\ e\in s_{\pi(i)}}} w_{\pi(i)}\theta_e^{d_e}(S^i) \\
&\geq \sum_{e\in E}\int_{x=0}^{\theta_e(S^\ell)} x^{d_e}dx \qquad (3)\\
&\geq \frac{1}{(d+1)}\sum_{e\in E}\theta_e^{d_e+1}(S^\ell) = \frac{C(S^\ell)}{(d+1)}.
\end{aligned}$$

□

Lemma 6. *For any $\beta \geq 1$, given a (T,β)-bounded covering walk R, it holds $\rho(R) \leq \beta(W + w_{max})^d\textsc{Opt}$.*

Proof. By the definition of $\rho(R)$, it holds

$$\begin{aligned}
\rho(R) &= \sum_{i=1}^{\ell}\sum_{e\in s^*_{\pi(i)}} w_{\pi(i)}\left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e} \\
&\leq \sum_{i=1}^{\ell}\sum_{e\in s^*_{\pi(i)}} w_{\pi(i)}(W + w_{max})^{d_e} \\
&\leq (W + w_{max})^d \sum_{i=1}^{\ell} |s^*_{\pi(i)}| w_{\pi(i)} \\
&\leq \beta(W + w_{max})^d\textsc{Opt}, \qquad (4)
\end{aligned}$$

where (4) holds by observing that $\textsc{Opt} \geq \frac{1}{\beta}\sum_{i=1}^{\ell}|s^*_{\pi(i)}|w_{\pi(i)}$.

□

As an immediate consequence of Lemmas 5 and 6, $\mathrm{Apx}_1^{\beta}(\mathcal{G}) \leq \beta(d+1)(W + w_{max})^d$. Lemmas 7 and 8 will be useful to extend such result to a (T,β)-bounded k-covering walk $P = \langle R_1,\ldots,R_k\rangle$ by exploiting the relationship among two consecutive walks. To this aim, recalling that $R = \left(S^0,\ldots,S^\ell\right)$, we define the following

function

$$H(R) = \sum_{i=1}^{\ell} \sum_{e \in s^*_{\pi(i)}} w_{\pi(i)} \theta_e^{d_e}(S^0).$$

By exchanging the order of the summations, we obtain

$$H(R) = \sum_{e \in E^*} \theta_e^{d_e}(S^0) x_e,$$

where $x_e = \sum_{j \in X_e} w_{\pi(j)}$ and $X_e = \left\{ i \in \{1, \ldots, \ell\} | e \in s^*_{\pi(i)} \right\}$. Informally speaking, $H(R)$ represents the sum over all the moves in the covering walk R of the weighted delay that the moving player $\pi(i)$ would experience in the initial state S^0 of R on her optimal strategy $s^*_{\pi(i)}$.

Clearly, as it can be noticed by their definition, $H(R)$ and $\rho(R)$ are correlated and Lemma 7 (resp. Lemma 8) will provide a lower bound (resp. an upper bound) to $\rho(R)$ in terms of $H(R')$ (resp. $H(R)$), where R and R' are two consecutive covering walks, that is the final state of R coincides with the initial one of R'.

Roughly speaking, on the one hand, Lemma 7 shows that the ratio between $H(R')$ and Opt is significantly less than the one between $\Gamma(R)$ and Opt; in other words, recalling that $\Gamma(R)$ is an upper bound to $\sum_{i=1}^{\ell_R} c_{\pi_R(i)}(S^i_R)$ and that $H(R')$ is the sum over all the moves in R' of the delays that the moving players would experience, in the first state of R', on her optimal strategies, we are able to show that in the first state of the next covering R' the congestion on the resources used in the considered optimal solution is such that $\frac{H(R')}{\text{Opt}} \leq \beta(d+1)\left((d+1)\frac{\rho(R)}{\text{Opt}}\right)^{\frac{d}{d+1}}$. On the other hand, Lemma 8 shows that the ratio between $\Gamma(R')$ and Opt is not much greater than the one between $H(R')$ and Opt. ; in other words, $H(R')$, even if referring to the delay in the first state of R', can be already considered as an "approximated" bound to $\Gamma(R')$, that is in turn an upper bound to $\sum_{i=1}^{\ell_{R'}} c_{\pi_{R'}(i)}(S^i_{R'})$. Therefore, the combination of Lemmata 7 and 8 gives that the ratio between $\Gamma(R')$ and Opt is significantly less than the one between $\Gamma(R)$ and Opt.

In fact, by combining $k-1$ times the results of Lemmata 7 and 8, Theorem 2 finally derives an upper bound to $\text{Apx}_k^{\beta}(\mathcal{G})$.

Lemma 7. *For any $\beta \geq 1$, given two consecutive (ℓ, β)-bounded covering walks R and R' such that the final state of R coincides with the initial one of R', it holds*

$$\frac{H(R')}{\text{Opt}} \leq \beta(d+1)\left((d+1)\frac{\rho(R)}{\text{Opt}}\right)^{\frac{d}{d+1}}.$$

Proof. Since the final state S^{ℓ} of R corresponds to the initial state of R', and recalling that each player can perform at most β best response within the same

(ℓ,β)-bounded covering walk, we obtain

$$\begin{aligned}
H(R') &\leq \beta \sum_{e\in E^*} \theta_e^{d_e}(S^\ell)\theta_e(S^*) \\
&\leq \beta \sum_{j=0}^{d} \sum_{e\in E^*:d_e=j} \theta_e^{j}(S^\ell)\theta_e(S^*) \\
&\leq \beta \sum_{e\in E^*:d_e=0} \theta_e(S^*) + \\
&\quad +\beta \sum_{j=1}^{d} \left(\left(\sum_{e\in E^*:d_e=j} (\theta_e^j(S^\ell))^{\frac{j+1}{j}} \right)^{\frac{j}{j+1}} \cdot \right. \\
&\quad \left. \cdot \left(\sum_{e\in E^*:d_e=j} (\theta_e^{j+1}(S^*)) \right)^{\frac{1}{j+1}} \right) \qquad (5) \\
&= \beta \sum_{e\in E^*:d_e=0} \theta_e(S^*) + \\
&\quad \beta \sum_{j=1}^{d} \left(\left(\sum_{e\in E^*:d_e=j} (\theta_e^{j+1}(S^\ell)) \right)^{\frac{j}{j+1}} \cdot \right. \\
&\quad \left. \cdot \left(\sum_{e\in E^*:d_e=j} (\theta_e^{j+1}(S^*)) \right)^{\frac{1}{j+1}} \right) \\
&\leq \beta \textsc{Opt} + \beta \sum_{j=1}^{d} \left(\left(C(S^\ell) \right)^{\frac{j}{j+1}} \textsc{Opt}^{\frac{1}{j+1}} \right) \\
&= \beta \sum_{j=0}^{d} \left(\left(C(S^\ell) \right)^{\frac{j}{j+1}} \textsc{Opt}^{\frac{1}{j+1}} \right) \\
&\leq \beta \sum_{j=0}^{d} \left(\left((d+1)\frac{\rho(R)}{\textsc{Opt}} \textsc{Opt} \right)^{\frac{j}{j+1}} \textsc{Opt}^{\frac{1}{j+1}} \right) \qquad (6) \\
&= \beta \sum_{j=0}^{d} \left(\left((d+1)\frac{\rho(R)}{\textsc{Opt}} \right)^{\frac{j}{j+1}} \textsc{Opt} \right) \\
&\leq \beta(d+1) \left((d+1)\frac{\rho(R)}{\textsc{Opt}} \right)^{\frac{d}{d+1}} \textsc{Opt}.
\end{aligned}$$

where (5) follows from Hölder's inequality, stating that, for r and s such that $\frac{1}{r} + \frac{1}{s} = 1$,

$$\sum_{j=1}^{q} a_j b_j \leq \left(\sum_{j=1}^{q} a_j^r\right)^{1/r} \left(\sum_{j=1}^{q} b_j^s\right)^{1/s}$$

by replacing r with $\left(\frac{d+1}{d}\right)$ and s with $(d+1)$, and (6) follows from Lemma 5.

□

Lemma 8. *For any $\beta \geq 1$, given a (ℓ, β)-bounded covering walk R, it holds*

$$H(R) \geq \left(\frac{1}{2^{d-1}} - \frac{d}{\alpha}\right)\rho(R) - \beta\left((\alpha\beta)^d + 1\right)\textsc{Opt},$$

for any $\alpha > d\, 2^{d-1}$.

Proof. In order to lower bound $H(R)$ with respect to $\rho(R)$, we define the following suitable potential function $h_i(R) = \sum_{e \in E^*} g_e(S^i) x_e^{>i}$ for $i \in \{0, \ldots, \ell\}$, where for a generic state S, $g_e(S) = \max\{0,\ f_e(\theta_e(S)) - f_e(\alpha\beta\theta_e(S^*))\}$ and $x_e^{>k} = \sum_{j \in X_e^{>k}} w_{\pi(j)}$ where $X_e^{>k} = \left\{i \in \{k+1, \ldots, \ell\} | e \in s^*_{\pi(i)}\right\}$. Informally speaking, such a potential function takes into account the delay due to the congestion of the not yet moving players during walk R above a "virtual" congestion frontier given by all the values $\alpha\beta n_e(S^*)$. Let $\Delta_i(R) = h_{i-1}(R) - h_i(R)$ for $i \in \{1, \ldots, \ell\}$. Notice that by the definition of the potential function $h_i(R)$, since $h_\ell(R) = 0$, $\sum_{i=1}^{\ell} \Delta_i(R) = h_0(R) \leq H(R)$, that is a lower bound for $\sum_{i=1}^{\ell} \Delta_i(R)$ is also a lower bound for $H(R)$; therefore, in the following we focus on lower bounding $\sum_{i=1}^{\ell} \Delta_i(R)$.

Consider a generic step i in walk R, in which player $\pi(i)$ performs a best response by selecting resources in $s^i_{\pi(i)}$ and let us bound from below the value of $\Delta_i(R)$ by evaluating how much player $\pi(i)$ removes from $h_{i-1}(R)$ and how much she adds to $h_i(R)$. Player $\pi(i)$ in order to obtain $h_i(R)$ removes at least $\sum_{e \in s^*_{\pi(i)}} w_{\pi(i)} g_e(S^{i-1})$ from $h_{i-1}(R)$, due to the decrease of the coefficients $x_e^{>(i-1)}$ to $x_e^{>i}$. Let us evaluate how much player $\pi(i)$ adds to $h_i(R)$. Player $\pi(i)$ increases the value of $h_i(R)$ only by resources whose congestion is above the virtual frontier after player $\pi(i)$ plays her best response. Thus for each resource $e \in s^i_{\pi(i)}$ such that $\theta_e(S^i) > \alpha\beta\theta_e(S^*)$, the increase of $h_i(R)$ is equal to $\left(g_e(S^i) - g_e(S^{i-1})\right) x_e^{>i}$ which, by the definition of g_e, is equal to $\left(f_e(\theta_e(S^i)) - f_e(\theta_e(S^{i-1}))\right) x_e^{>i}$. Since f_e is convex, such quantity is at most $\left(\theta_e(S^i) - \theta_e(S^{i-1})\right) f'_e(\theta_e(S^i)) x_e^{>i}$, that is equal to $w_{\pi(i)} f'_e(\theta_e(S^i)) x_e^{>i}$, where f'_e is the derivative of f_e. Moreover since $x_e^{>i} \leq x_e \leq \beta\theta_e(S^*) \leq \theta_e(S^i)/\alpha$, we obtain that the increase for each resource e is at most $w_{\pi(i)} f'_e(\theta_e(S^i))\theta_e(S^i)/\alpha$. Thus, considering the previous quantity as an

upper bound of the increase for all the resources in $s^i_{\pi(i)}$, player $\pi(i)$ in order to obtain $h_i(R)$ adds at most $\frac{1}{\alpha}\sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} f'_e(\theta_e(S^i))\theta_e(S^i)$ to $h_{i-1}(R)$. Therefore,

$$\begin{aligned}\Delta_i(R) \;\geq\; & \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} g_e(S^{i-1}) \\ & -\frac{1}{\alpha}\sum_{e\in s^i_{\pi(i)}\cap E^*} w_{\pi(i)} f'_e(\theta_e(S^i))\theta_e(S^i).\end{aligned}$$

Finally, since $g_e(S^{i-1}) \geq f_e(\theta_e(S^{i-1})) - f_e(\alpha\beta\theta_e(S^*))$ for every $e \in E^*$, it follows that

$$\begin{aligned}\Delta_i(R) \;\geq\; & \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(f_e(\theta_e(S^{i-1})) - f_e(\alpha\beta\theta_e(S^*))\right) \\ & -\frac{1}{\alpha}\sum_{e\in s^i_{\pi(i)}\cap E^*} w_{\pi(i)} f'_e(\theta_e(S^i))\theta_e(S^i). \qquad (7)\end{aligned}$$

Since $f_e(x) = x^{d_e}$ with $d_e \geq 0$, we obtain

$$\begin{aligned}\Delta_i(R) \;\geq\; & \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta^{d_e}_e(S^{i-1}) - (\alpha\beta)^{d_e}\theta^{d_e}_e(S^*)\right) \\ & -\frac{1}{\alpha}\sum_{e\in s^i_{\pi(i)}\cap E^*} d_e w_{\pi(i)}\theta^{d_e}_e(S^i) \\ \geq\; & \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta^{d_e}_e(S^{i-1}) - (\alpha\beta)^{d_e}\theta^{d_e}_e(S^*)\right) \\ & -\frac{d}{\alpha} c_{\pi(i)}(S^i).\end{aligned}$$

Since players perform best responses,

$$c_{\pi(i)}(S^i) \leq \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e}; \qquad (8)$$

moreover, $w_{\pi(i)} \leq \theta_e(S^*)$ for every $e \in s^*_{\pi(i)}$; thus, by using Claim 1 in (9), it follows that

$$\begin{aligned}\Delta_i(R) \;\geq\; & \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta^{d_e}_e(S^{i-1}) - (\alpha\beta)^{d_e}\theta^{d_e}_e(S^*)\right) \\ & -\frac{d}{\alpha}\sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e}\end{aligned}$$

$$
\begin{aligned}
&\geq \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\frac{1}{2^{d_e-1}} \left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^d \right. \\
&\quad \left. - w_{\pi(i)}^{d_e} - (\alpha\beta)^{d_e}\theta_e^{d_e}(S^*)\right) \\
&\quad - \frac{d}{\alpha} \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e} \qquad (9) \\
&\geq \left(\frac{1}{2^{d-1}} - \frac{d}{\alpha}\right) \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e} \\
&\quad - \left((\alpha\beta)^d + 1\right) \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)}\theta_e^{d_e}(S^*).
\end{aligned}
$$

By summing up the values $\Delta_i(R)$, we obtain

$$
\begin{aligned}
&\sum_{i=1}^{\ell} \Delta_i(R) \\
&\geq \left(\frac{1}{2^{d-1}} - \frac{d}{\alpha}\right) \sum_{i=1}^{\ell} \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)} \left(\theta_e(S^{i-1}) + w_{\pi(i)}\right)^{d_e} \\
&\quad - \left((\alpha\beta)^d + 1\right) \sum_{i=1}^{\ell} \sum_{e\in s^*_{\pi(i)}} w_{\pi(i)}\theta_e^{d_e}(S^*) \\
&\geq \left(\frac{1}{2^{d-1}} - \frac{d}{\alpha}\right)\rho(R) - \beta\left((\alpha\beta)^d + 1\right)\textsc{Opt},
\end{aligned}
$$

and thus, since $H(R) \geq \sum_{i=1}^{\ell} \Delta_i(R)$, the claim follows.

□

Theorem 2. *For any $k \geq 1$ and any given $d \geq 1$, in every polynomial congestion game $\mathcal{G}$ with delay functions having maximum degree d, it holds that the approximation ratio of the state reached after k (ℓ,β)-bounded covering walks is $O\left(W^{d\left(\frac{d}{d+1}\right)^{k-1}}\right)$.*

Proof. Let $P = \langle R_1, \ldots, R_k\rangle$ be the sequence of k (ℓ,β)-bounded covering walks. From Lemma 8 we obtain that for each R_j with $j = 2, \ldots, k$ it holds that for any $\alpha > d\ 2^{d-1}$

$$
H(R_j) \geq \left(\frac{1}{2^{d-1}} - \frac{d}{\alpha}\right)\rho(R_j) - \beta\left((\alpha\beta)^d + 1\right)\textsc{Opt}. \qquad (10)
$$

Furthermore, by applying Lemma 7 we obtain that for each R_j with $j = 2, \ldots, k$ it holds

$$H(R_j) \leq \beta(d+1)\left((d+1)\frac{\rho(R_{j-1})}{\textsc{Opt}}\right)^{\frac{d}{d+1}} \textsc{Opt}. \tag{11}$$

By combining (10) and (11) we achieve a relation between $\frac{\rho(R_j)}{\textsc{Opt}}$ and $\frac{\rho(R_{j-1})}{\textsc{Opt}}$ for every $j = 2, \ldots, k$

$$\begin{aligned}\frac{\rho(R_j)}{\textsc{Opt}} \;\leq\; & \beta\left(\frac{\alpha 2^{d-1}}{\alpha - d\,2^{d-1}}\right)\cdot \\ & \cdot\left((d+1)\left((d+1)\frac{\rho(R_{j-1})}{\textsc{Opt}}\right)^{\frac{d}{d+1}} + \right. \\ & \left. + \left((\alpha\beta)^d + 1\right)\right)\end{aligned} \tag{12}$$

for any $\alpha > d\,2^{d-1}$.
From the previous inequalities (12) (for $j = 2, \ldots, k$), since $\beta = O(1)$, we obtain that for constant values of α and d it holds

$$\frac{\rho(R_k)}{\textsc{Opt}} = O\left(\left(\frac{\rho(R_1)}{\textsc{Opt}}\right)^{\left(\frac{d}{d+1}\right)^{k-1}}\right). \tag{13}$$

By applying Lemma 5 to $\rho(R_k)$ and Lemma 6 to $\rho(R_1)$ in (13), since $\beta = O(1)$, by constant value of d we obtain that the cost of the final state of walk P is

$$O\left(\left((W + w_{max})^d\right)^{\left(\frac{d}{d+1}\right)^{k-1}} \textsc{Opt}\right),$$

and thus the approximation is

$$\mathrm{Apx}_k^{O(1)}(\mathcal{G}) = O\left(W^{d\left(\frac{d}{d+1}\right)^{k-1}}\right).$$

□

The following corollary is an immediate consequence of Theorem 2.

Corollary 1. *For any polynomial congestion game $\mathcal{G}$ with $d = O(1)$, a best response dynamics satisfying the (T,β)-Fairness Condition converges from any initial state to a state having approximation ratio $O(1)$ in at most* $\log\log W$ *best responses.*

Now we provide an almost matching lower bound to the approximation ratio achieved after a (ℓ,β)-bounded k-covering walk.

Theorem 3. *For any $d \geq 1$ there exists a congestion game $\mathcal{G}$ with polynomial delay functions having maximum degree d such that the approximation ratio achieved after k (ℓ,β)-bounded covering walks is $\Omega\left(\frac{W^{d}\left(\frac{d}{d+1}\right)^{k-1}}{k}\right)$.*

Proof. Given any integers k and c, we consider a congestion game $\mathcal{G}$ defined on a set of $n = m(k+1)$ players having weight equal to 1 (i.e. $W = n$) and $m(k+2)$ resources, with $m = c^{(d+1)^k}$. The set of players can be partitioned into $k+1$ sets $P_0, \ldots, P_k$, each containing m players, i.e. $P_j = \{p_j^1, \ldots, p_j^m\}$ for $j = 0, \ldots, k$; the set of resources can be partitioned into $k+2$ sets $T_0, \ldots, T_{k+1}$ each containing m resources, i.e. $T_j = \{e_j^1, \ldots, e_j^m\}$ for $j = 0, \ldots, k+1$, and such that each resource has delay function $f(x) = x^d$.

The strategy set of player p_j^i (the i-th player of set P_j) consists of two strategies: the *left* strategy and the *right* strategy. The left strategy for player p_j^i of set P_j ($j = 0, \ldots, k$) consists of the only resource e_j^i belonging to set T_j.

The right strategy of each player in P_0 is T_1; in order to define the right strategies of the remaining players, we need some additional definitions.

Let $b_{i,j} = \left\lfloor \frac{i}{m^{\left(\frac{d}{d+1}\right)^j}} \right\rfloor$; for each $j = 2, \ldots, k$, we partition the set T_j into $\frac{m}{m^{\left(\frac{d}{d+1}\right)^{j-1}}}$ (by the definition of m it is an integer) *blocks* $T_j^0, \ldots, T_j^{\frac{m}{m^{\left(\frac{d}{d+1}\right)^{j-1}}}-1}$. The right strategy for player p_j^i of set P_j ($j = 1, \ldots, k-1$) is $T_{j+1}^{b_{i,j}}$. Notice that, by the definition of $b_{i,j}$, the number of players in P_j having as their right strategy block T_{j+1}^h for a fixed h is $m^{\left(\frac{d}{d+1}\right)^j}$, and for any resource in such a block, the number of players whose right strategy contains it is $m^{\left(\frac{d}{d+1}\right)^j}$.

Let us compute the social cost of the configuration obtained after k walks starting from the state in which all the players select their right strategy.

We assume that, during all the walks, the players of set P_j perform their best response before the ones of set P_{j-1} for $j = 1, \ldots, k$, and for every $i = 0, \ldots, k$ players $p_j^i \in P_j$ perform their best response in increasing order with respect to i.

Let us consider walk R_j, $j = 1, \ldots, k$. We now show by induction on the number of walks that the players belonging to P_x, $x = j, \ldots, k$, choose their right strategy, while the ones belonging to P_x, $x = 0, \ldots, j-1$, choose their left strategy.

Assume by induction that this is true until R_{j-1}; we now show that it holds also for R_j.

- The players belonging to P_x, $x = j, \ldots, k$, do not change their (right) strategy. In fact, on the one hand, by the inductive hypothesis, since each resource in their right strategies is used by $m^{\left(\frac{d}{d+1}\right)^x}$ players, their right strategy has a cost $m^{\left(\frac{d}{d+1}\right)^x} \cdot \left(m^{\left(\frac{d}{d+1}\right)^x}\right)^d = m^{\frac{d^x}{(d+1)^{x-1}}}$. On the other hand, always

by the inductive hypothesis, their left strategies (composed by a unique resource) is used by $m^{\left(\frac{d}{d+1}\right)^{x-1}}$ players and therefore has a cost equal to $\left(m^{\left(\frac{d}{d+1}\right)^{x-1}}\right)^{d} = m^{\frac{d^x}{(d+1)^{x-1}}}$.

- The players belonging to P_{j-1} select their left strategies. In fact, on the one hand, since their right strategy is composed by $m^{\left(\frac{d}{d+1}\right)^{j-1}}$ resources, it has a cost at least $m^{\left(\frac{d}{d+1}\right)^{j-1}}$. On the other hand, by the inductive hypothesis, their left strategies (composed by a unique resource) is free, and therefore has a cost equal to 1.
- The players belonging to P_x, $x = 0, \ldots, j-2$, do not change their (left) strategy where they have a delay equal to 1 (i.e. the minimum possible one in such an instance).

Thus, in the final state of walk R_k, all the players in P_k are using their right strategy consisting of $m^{\left(\frac{d}{d+1}\right)^{k}}$ resources in T_{k+1}. Moreover, each of such resources is used by $m^{\left(\frac{d}{d+1}\right)^{k}}$ players, and it follows that the social cost of the final state of walk R_k is at least $m \cdot \left(m^{\left(\frac{d}{d+1}\right)^{k}}\right)^{d} \cdot m^{\left(\frac{d}{d+1}\right)^{k}} = m \cdot m^{d\left(\frac{d}{d+1}\right)^{k-1}}$. Since the configuration in which each player uses her left strategy costs n,

$$\mathrm{Apx}_k^1(\mathcal{G}) \geq \frac{m \cdot m^{d\left(\frac{d}{d+1}\right)^{k-1}}}{n} = \Omega\left(\frac{W^{d\left(\frac{d}{d+1}\right)^{k-1}}}{k}\right).$$

□

5.2 General β [15]

In this subsection, we focus on the special case of congestion games with linear delays, and completely characterize how the fairness of the dynamics affects the time of convergence to good solutions, approximating an optimal one by a constant factor.

Since the dynamics satisfies the (T,β)-Fairness Condition, we can decompose it into k (ℓ,β)-bounded coverings $R_1, \ldots, R_k$.

Consider a generic (ℓ,β)-bounded covering $R = \left(S^0, \ldots, S^\ell\right)$. Given an optimal strategy profile S^*, since the t-th player $\pi(t)$ performing a best response, before doing it, can always select the strategy she would use in S^*, her *immediate cost* $c_{\pi(t)}(S^t)$ can be suitably upper bounded as $\sum_{e \in s^*_{\pi(t)}} \left(\theta_e(S^{t-1}) + w_{\pi(t)}\right)$.

By extending and strengthening the technique of [13, 14], we are able to prove that the best response dynamics satisfying the (T,β)-Fairness Condition fast converges to states approximating the social optimum by a factor $O(\beta)$. It is worth

noticing that, by exploiting the technique of [13, 14], only a worse bound of $O(\beta^2)$ could be proved. In order to obtain an $O(\beta)$ bound, we need to develop a different and more involved technique, in which also the functions ρ and H, introduced in [13, 14], have to be redefined: roughly speaking, they now must take into account only the last move in R of each player, whereas in [13, 14] they were accounting for all the moves in R.

We now introduce functions ρ and H, defined over the set of all the possible (ℓ,β)-bounded coverings:

- Let $\rho(R) = \sum_{i=1}^{n} w_i \sum_{e \in s_i^*} \left(\theta_e(S^{\text{last}_R(i)-1}) + w_i\right)$;
- let $H(R) = \sum_{i=1}^{n} w_i \sum_{e \in s_i^*} \theta_e(S^0)$.

Notice that $\rho(R)$ is an upper bound to the sum over all the players of the cost that she would experience on her optimal strategy s_i^* just before her last move in R, whereas $H(R)$ represents the sum over all the players of the delay on the moving player's optimal strategy s_i^* in the initial state S^0 of R. Moreover, since players perform best responses, it holds that, for any $i = 1, \ldots, n$,

$$c_i(S^{\text{last}_R(i)}) \leq \sum_{e \in s_i^*} \left(\theta_e(S^{\text{last}_R(i)-1}) + w_i\right) \tag{14}$$

and, any summing over all players, we obtain that $\sum_{i=1}^{n} c_i(S^{\text{last}_R(i)}) \leq \rho(R)$, i.e. $\rho(R)$ is an upper bound to the sum of the immediate costs over the last moves of every players. Finally, it is worth noticing that, by inverting the order of the summations, $H(R) = \sum_{e \in E} \theta_e(S^0)\theta_e(S^*)$.

The upper bound proof is structured as follows. Lemma 9 relates the social cost of the final state S^ℓ of a (ℓ,β)-bounded covering R with $\rho(R)$, by showing that $C(S^\ell) \leq 2\rho(R)$. Let $\overline{R}$ and R be two consecutive (ℓ,β)-bounded coverings; by exploiting Lemmata 10 and 11, providing an upper (lower, respectively) bound to $H(R)$ in terms of $\rho(\overline{R})$ ($\rho(R)$, respectively), Lemma 12 proves that $\frac{\rho}{\text{OPT}}$ rapidly decreases between $\overline{R}$ and R, showing that $\frac{\rho(R)}{\text{OPT}} = O\left(\sqrt{\frac{\rho(\overline{R})}{\text{OPT}}} + \beta\right)$. In the proof of Theorem 1, after deriving a trivial upper bound equal to $O(W)$ for $\rho(R_1)$, Lemma 12 is applied to all the $k-1$ couples of consecutive (ℓ,β)-bounded coverings of the considered dynamics satisfying the (T,β)-Fairness Condition.

Similarly to Lemma 5 and Lemma 7, the following lemmata show that the social cost at the end of any (ℓ,β)-bounded covering R is at most $2\rho(R)$, and that $\frac{H(R')}{\text{OPT}}$ is significantly less than $\frac{\rho(\overline{R})}{\text{OPT}}$ for two consecutive coverings R and R'.

Lemma 9. *For any $\beta \geq 1$, given a (ℓ,β)-bounded covering R, $C(S^\ell) \leq 2\rho(R)$.*

Lemma 10. *For any $\beta \geq 1$, given two consecutive (ℓ, β)-bounded covering walks R and R' such that the final state of R coincides with the initial one of R', it holds, $\frac{H(R')}{\text{Opt}} \leq \sqrt{2\frac{\rho(R)}{\text{Opt}}}$.*

In Lemma 11 we are able to relate $\rho(R)$ and $H(R)$ by strengthening the technique exploited in [13, 14].

Lemma 11. *For any $\beta \geq 1$, given a (ℓ, β)-bounded covering R, $\frac{\rho(R)}{\text{Opt}} \leq 2\frac{H(R)}{\text{Opt}} + 4\beta + 3$.*

Proof. Let $\bar{N}$ be the set of players changing their strategies by performing best responses in R. First of all, notice that if the players in $\bar{N}$ never select strategies used by some player in S^*, i.e. if they select only resources e such that $\theta_e(S^*) = 0$, then, by recalling the definitions of $\rho(R)$ and $H(R)$, we would obtain

$$\begin{aligned}
\rho(R) &= \sum_{i=1}^{n} w_i \sum_{e \in s_i^*} \left(\theta_e(S^{\text{last}_R(i)-1}) + w_i\right) \\
&= \sum_{i=1}^{n} w_i \sum_{e \in s_i^*} \left(\theta_e(S^0) + w_i\right) \\
&\leq \sum_{i=1}^{n} w_i \left(\sum_{e \in\ s_i^*} \theta_e(S^0) + \sum_{e \in s_i^*} w_i\right) \\
&= \sum_{e \in\ E} \theta_e(S^0)\theta_e(S^*) + \sum_{i=1}^{n} \sum_{e \in s_i^*} w_i^2 \\
&\leq \sum_{e \in\ E} \theta_e(S^0)\theta_e(S^*) + \sum_{e \in E} \theta_e^2(S^*) \\
&= H(R) + \text{Opt},
\end{aligned}$$

and the claim would easily follow for any $\beta \geq 1$.

In the following our aim is that of dealing with the generic case in which players moving in R can increase the congestion on resources e such that $\theta_e(S^*) > 0$.

For every resource $e \in E$, we focus on the congestion on such a resource above a "virtual" congestion frontier $g_e = 2\beta\theta_e(S^*)$.

We assume that at the beginning of covering R each resource $e \in E$ has a delay equal to $\delta_{0,e} = \max\left\{\theta_e(S^0) + \theta_e(S^*), g_e\right\}$, and we call $\delta_{0,e}$ the *delay of level* 0 on resource e. $\Delta_0 = \sum_{e \in E} \delta_{0,e} \cdot \theta_e(S^*)$ is an upper bound to $H(R)$. We refer to Δ_0 as the total delay of level 0. Moreover, it holds that

$$\Delta_0 = \sum_{e \in E} \max\left\{\theta_e(S^0) + \theta_e(S^*), 2\beta\theta_e(S^*)\right\} \cdot \theta_e(S^*)$$

$$\begin{aligned} &\leq \sum_{e\in E}\Big(\theta_e(S^0)+\theta_e(S^*)\Big)\cdot\theta_e(S^*)+2\beta\sum_{e\in E}\theta_e^2(S^*) \qquad (15)\\ &= H(R)+(2\beta+1)\textsc{Opt}. \end{aligned}$$

The idea is that the total delay of level 0 can induce on the resources a congestion (over the frontier g_e) contributing to the total delay of level 1, such a delay a congestion contributing (always over the frontier g_e) to the total delay of level 2, and so on.

More formally, for any $p \geq 1$ and any $e \in E$, we define $\delta_{p,e}$ as the delay of level p on resource e; we say that a delay $\delta_{p,e}$ of level p on resource e is *induced* by an amount $x_{p-1,e}$ of delay of level $p-1$ if some players (say, players in $N_{p-1,e}$) moving on e can cause such a delay of level p on e because they are experimenting a delay of level $p-1$ on the resources of their optimal strategies equal to $x_{p-1,e}$. In other words, $x_{p-1,e}$ is the overall delay of level $p-1$ on the resources in the optimal strategies of players in $N_{p-1,e}$ used in order to induce the delay $\delta_{p,e}$ of level p on resource e.

For every $i = 1, \ldots, n$, since players perform best responses, player i, in order to select a strategy s_i, must suffer a delay when selecting her optimal strategy s_i^* at least equal to the cost of the selected strategy s_i. In the following, we will assume that a player i can perform a best response selecting a strategy s_i if her cost for strategy s_i^* computed according to the delays $\delta_{p,e}$ ($e \in s_i^*$, $p \geq 0$) is at least her cost for strategy i. Such a delay is initially induced, for every $e \in s_i^*$, by the congestion $\theta_e(S^0)$; the additive term $\theta_e(S^*)$ in the definition of $\delta_{0,e}$ is due to the fact that, before performing her best response, player i weight might not belong to $\theta_e(S^0)$, while w_i has to be taken into account when computing the delay suffered by i when selecting her optimal strategy s_i^*.

We have to clarify how a total delay belonging to various levels is exploited in order to induce the delay of a given level, say level p, on a resource, say resource e. In fact, only at the beginning we can assume that all the delay is of level 0 and therefore it is entirely used in order to induce a delay of level 1, while at a generic move the delay on a resource can belong to different levels. Consider a best response s_i of a generic player i, with $e \in s_i$; if D is the total amount of delay needed in order to select resource e (delay D is belonging to some resources of s_i^*), we assume that the amount of the induced delay of a given level p on resource e is proportional to the amount of delay of level $p-1$ contained in D.

Assume that resource e has a congestion $\alpha \geq g_e$ and that player i is selecting strategy s_i with $e \in s_i$. The delay of e increases of w_i, and in order to perform such a best response, player i has to suffer a cost equal to $D = w_i(\alpha + w_i)$ on her optimal strategy. We have that $w_i(\alpha + w_i) \geq \alpha w_i$, i.e., the delay on s_i^* used in order to increase the delay on e is at least α times the amount of the increase.

If $\alpha < g_e$ and $\alpha + w_i > g_e$, the delay of e increases of ϵ, with $\epsilon = \alpha + w_i - g_e < w_i$

and in order to perform such a best response, player i has to suffer a cost equal to $D = w_i(g_e + \epsilon) > \epsilon(g_e + \epsilon)$ on her optimal strategy. Therefore, we again obtain that the delay on s_i^* used in order to increase the delay on e is at least α times the amount of the increase.

Since we are assuming that the amount of the induced delay of a given level p on resource e is proportional to the amount of delay of level $p-1$ contained in D, we obtain that

$$\delta_{p,e} \leq \frac{x_{p-1,e}}{\alpha} \leq \frac{x_{p-1,e}}{g_e}. \tag{16}$$

For any p, the total delay of level p is defined as $\Delta_p = \sum_{e \in E} \delta_{p,e} \cdot \theta_e(S^*)$. Moreover, for any $p \geq 1$, we have that

$$\sum_{e \in E} x_{p-1,e} \leq \beta \Delta_{p-1} \tag{17}$$

because each player can move at most β times in R and therefore the total delay of level $p-1$ can be used at most β times in order to induce the total delay of level p.

We also have that $\rho(R) \leq \sum_{p=0}^{\infty} \Delta_p + \textsc{Opt}$, because $\sum_{p=0}^{\infty} \delta_{p,e}$ is an upper bound on the delay of resource e during the whole covering R.

In the following, we bound $\sum_{p=0}^{\infty} \Delta_p$ from above.

$$\Delta_p = \sum_{e \in E} \delta_{p,e} \cdot \theta_e(S^*) \leq \sum_{e \in E} \frac{x_{p-1,e}}{g_e} \cdot \theta_e(S^*) = \sum_{e \in E} \frac{x_{p-1,e}}{2\beta\theta_e(S^*)} \cdot \theta_e(S^*) \leq \frac{\Delta_{p-1}}{2},$$

where the first inequality holds by inequality (16), and the last inequality holds by inequality (17).

We thus obtain that, for any $p \geq 0$, $\Delta_p \leq \frac{\Delta_0}{2^p}$ and $\sum_{p=0}^{\infty} \Delta_p \leq 2\Delta_0$. Therefore,

$$\rho(R) \leq \sum_{p=0}^{\infty} \Delta_p + \textsc{Opt} \leq 2\Delta_0 + \textsc{Opt}.$$

Finally, the claim follows by combining this upper bounds to $\rho(R)$ with the upper bound to Δ_0 derived in (15).

□

By combining Lemmata 10 and 11, the following lemma, showing that $\frac{\rho(\cdot)}{\textsc{Opt}}$ fast decreases between two consecutive coverings, holds.

Lemma 12. *For any $\beta \geq 1$, given two consecutive (ℓ,β)-bounded coverings $\overline{R}$ and R, $\frac{\rho(R)}{\textsc{Opt}} \leq 2\sqrt{2\frac{\rho(\overline{R})}{\textsc{Opt}}} + 4\beta + 3$.*

By applying Lemma 12 to all the couples of consecutive (ℓ,β)-bounded coverings, we are now able to prove the following theorem.

Theorem 4. *Given a weighted congestion game with linear delay functions, any best response dynamics satisfying the (T,β)-Fairness Condition converges from any initial state to a state S such that $\frac{C(S)}{\text{Opt}} = O(\beta)$ in at most $T\lceil \log\log W\rceil$ best responses.*

Proof. Given a best response dynamics satisfying the (T,β)-Fairness Condition, let $R_1,\ldots,R_k$ be the k (ℓ,β)-bounded coverings in which it can be decomposed. By applying Lemma 12 to all the pairs of consecutive (ℓ,β)-bounded coverings R_j and R_{j+1}, for any $j = 1,\ldots,k-1$ we obtain

$$\frac{\rho(R_{j+1})}{\text{Opt}} \leq 2\sqrt{2\frac{\rho(R_j)}{\text{Opt}}} + 4\beta + 3.$$

By combining all the above inequalities for $j = 1,\ldots,k-1$ and by performing some basic algebraic manipulations, for any constant value of d we obtain that $\frac{\rho(R_k)}{\text{Opt}} = O\left(\sqrt[2^{k-1}]{\frac{\rho(R_1)}{\text{Opt}}} + \beta\right)$. Thus, by Lemma 9, the cost of the final state S of walk R_k is such that

$$\frac{C(S)}{\text{Opt}} = O\left(\sqrt[2^{k-1}]{\frac{\rho(R_1)}{\text{Opt}}} + \beta\right).$$

By the definition of $\rho(R)$, since $\sum_{e\in E}\theta_e(S^*) \leq \sum_{e\in E}\theta_e^2(S^*) = \text{Opt}$, for any possible (ℓ,β)-bounded covering R it holds that

$$\begin{aligned}\rho(R) &= \sum_{i=1}^{n} w_i \sum_{e\in s_i^*}\left(\theta_e(S^{\text{last}(i)-1}) + w_i\right) \leq \sum_{i=1}^{n} w_i \sum_{e\in s_i^*}(W+W)\\ &= 2W\sum_{i=1}^{n} w_i|s_i^*| \leq 2W\sum_{e\in E}\theta_e(S^*) \leq 2W\text{Opt}.\end{aligned}$$

Therefore, $\frac{\rho(R_1)}{\text{Opt}} \leq 2W$ and we obtain $\frac{C(S)}{\text{Opt}} = O\left(\sqrt[2^{k-1}]{W} + \beta\right)$.

It is worth noticing that $\log\log W$ (ℓ,β)-bounded coverings are sufficient in order to obtain $\frac{C(S)}{\text{Opt}} = O(\beta)$. Since every (ℓ,β)-bounded covering, by its definition, contains at most T best responses, the claim follows. □

It is also possible to prove the following lower bounds.

Theorem 5. *For any $\epsilon > 0$, there exist a linear congestion game $\mathcal{G}$ and an initial state S^0 such that, for any $\beta = O(n^{-\frac{1}{\log_2\epsilon}})$, there exists a best response dynamics starting from S^0 and satisfying the (T,β)-Fairness Condition such that for a number of best responses exponential in n the cost of the reached states is $\Omega(\beta^{1-\epsilon}\cdot\text{Opt})$.*

By choosing $\beta = \sqrt{n}$ and considering a simplified version of the proof giving the above lower bound, it is possible to prove the following corollary. In particular, it shows that even in the case of best response dynamics verifying an $O(n)$-Minimum Liveness Condition, the speed of convergence to efficient states is very slow; such a fact implies that the T-Minimum Liveness condition cannot precisely characterize the speed of convergence to efficient states because it does not capture the notion of fairness in best response dynamics.

Corollary 2. *There exist a linear congestion game $\mathcal{G}$, an initial state S^0 and a best response dynamics starting from S^0 and satisfying the $O(n)$-Minimum Liveness Condition such that for a number of best responses exponential in n the cost of the reached states is always $\Omega\left(\frac{\sqrt[4]{n}}{\log n} \cdot \textsc{Opt}\right)$.*

In the symmetric case, the unfairness in best response dynamics does not affect the speed of convergence to efficient states. In particular, we are able to show that, for any β, after $T\lceil \log\log W \rceil$ best responses an efficient state is always reached.

Theorem 6. *Given a linear weighted symmetric congestion game, any best response dynamics satisfying the T-Minimum Liveness Condition converges from any initial state to a state S such that $\frac{C(S)}{\textsc{Opt}} = O(1)$ in at most $T\lceil \log\log W \rceil$ best responses.*

6 Conclusions and Future Work

In this work we have surveyed the state of the art about convergence issues in congestion games, with a special focus on the results concerning the speed of convergence of best response dynamics. In particular, we have shown that in congestion games with polynomial delays *fair* dynamics, in which each player is allowed to player at least once and at most a constant number of times every T best responses, fast converges to solutions approximating the optimum by a factor proportion to the price of anarchy.

Moreover, we have completely characterized how, in weighted congestion games with linear delays, the frequency with which each player participates in the game dynamics affects the possibility of reaching states with an approximation ratio within a constant factor from the price of anarchy, within a polynomially bounded number of best responses. We have shown that, while in the asymmetric setting the fairness among players is a necessary and sufficient condition for guaranteeing a fast convergence to efficient states, in the symmetric one the game always converges to an efficient state after a polynomial number of best responses,

regardless of the frequency each player moves with. We conjecture that such results can be extended to broader classes of congestion games, such as congestion games with polynomial delay functions.

It is worth to note that our techniques provide a much faster convergence to efficient states with respect to previous results in the literature. In particular, in the symmetric setting, Theorem 6 shows that best response dynamics leads to efficient states much faster than how ϵ-Nash dynamics (i.e., sequences of moves reducing the cost of a player by at least a factor of ϵ) leads to ϵ-Nash equilibria [8]. Furthermore, also in the more general asymmetric setting, Theorem 4 shows that the same holds for fair best response dynamics with respect to ϵ-Nash ones [5].

An interesting open question is that of studying the time of convergence to solutions approximating by a low factor the optimum with respect to other social function, such as the maximum cost among the players. Finally, considering other kinds of dynamics (such as coalitional responses in which two or more players coordinate in order to decrease their costs) is a left open problem that deserves further research effort.

References

[1] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. On the impact of combinatorial structure on congestion games. *J. ACM*, 55(6), 2008.

[2] Heiner Ackermann, Heiko Röglin, and Berthold Vöcking. Pure Nash equilibria in player-specific and weighted congestion games. *Theor. Comput. Sci.*, 410(17):1552–1563, 2009.

[3] Sebastian Aland, Dominic Dumrauf, Martin Gairing, Burkhard Monien, and Florian Schoppmann. Exact price of anarchy for polynomial congestion games. In *STACS*, volume 3884, pages 218–229. Springer, 2006.

[4] B. Awerbuch, Y. Azar, and A. Epstein. The price of routing unsplittable flow. In *STOC*, pages 57–66. ACM, 2005.

[5] Baruch Awerbuch, Yossi Azar, Amir Epstein, Vahab S. Mirrokni, and Alexander Skopalik. Fast convergence to nearly optimal solutions in potential games. In *ACM Conference on Electronic Commerce*, pages 264–273. ACM, 2008.

[6] Vittorio Bilò, Angelo Fanelli, Michele Flammini, and Luca Moscardelli. Performance of one-round walks in linear congestion games. *Theory Comput. Syst.*, 49(1):24–45, 2011.

[7] I. Caragiannis, M. Flammini, C. Kaklamanis, P. Kanellopoulos, and L. Moscardelli. Tight bounds for selfish and greedy load balancing. In *ICALP (1)*, volume 4051, pages 311–322. Springer, 2006.

[8] S. Chien and A. Sinclair. Convergence to approximate Nash equilibria in congestion games. In *SODA*, pages 169–178. SIAM, 2007.

[9] G. Christodoulou and E. Koutsoupias. The price of anarchy of finite congestion games. In *STOC*, pages 67–73. ACM, 2005.

[10] G. Christodoulou, V. S. Mirrokni, and A. Sidiropoulos. Convergence and approximation in potential games. In *STACS*, volume 3884, pages 349–360. Springer, 2006.

[11] M. Yannakakis D. S. Johnson, C. H. Papadimitriou. How easy is local search? *Journal of Computer and System Sciences*, 37:79–100, 1988.

[12] A. Fabrikant, C. H. Papadimitriou, and K. Talwar. The complexity of pure Nash equilibria. In *STOC*, pages 604–612. ACM, 2004.

[13] Angelo Fanelli, Michele Flammini, and Luca Moscardelli. The speed of convergence in congestion games under best-response dynamics. In *ICALP (1)*, volume 5125, pages 796–807. Springer, 2008.

[14] Angelo Fanelli and Luca Moscardelli. On best response dynamics in weighted congestion games with polynomial delays. *Distributed Computing*, 24:245–254, 2011.

[15] Angelo Fanelli, Luca Moscardelli, and Alexander Skopalik. On the impact of fair best response dynamics. In *MFCS*, volume 7464 of *LNCS*, pages 360–371. Springer, 2012.

[16] Babak Farzad, Neil Olver, and Adrian Vetta. A priority-based model of routing. *Chicago J. Theor. Comput. Sci.*, 2008, 2008.

[17] Dimitris Fotakis, Spyros C. Kontogiannis, Elias Koutsoupias, Marios Mavronicolas, and Paul G. Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *ICALP*, volume 2380, pages 123–134. Springer, 2002.

[18] Dimitris Fotakis, Spyros C. Kontogiannis, and Paul G. Spirakis. Atomic congestion games among coalitions. *ACM Transactions on Algorithms*, 4(4), 2008.

[19] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *STACS*, volume 1563, pages 404–413. Springer, 1999.

[20] Igal Milchtaich. Congestion games with player-specific payoff functions. *Games and Economic Behavior*, 13(1):111 – 124, 1996.

[21] Vahab S. Mirrokni and Adrian Vetta. Convergence issues in competitive games. In *APPROX-RANDOM*, volume 3122 of *LNCS*, pages 183–194. Springer, 2004.

[22] J. F. Nash. Equilibrium points in *n*-person games. In *Proceedings of the National Academy of Sciences*, volume 36, pages 48–49, 1950.

[23] R. W. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2:65–67, 1973.

[24] Alexander Skopalik and Berthold Vöcking. Inapproximability of pure Nash equilibria. In *STOC*, pages 355–364. ACM, 2008.

SECURE COMPUTATION UNDER NETWORK AND PHYSICAL ATTACKS

Alessandra Scafuro
Università degli Studi di Salerno

Abstract

Secure computation enables many parties to jointly compute a function of their private inputs. The security requirement is that the input privacy of any honest party is preserved even if other parties participating in the protocol collude or deviate from the protocol. In concurrent and physical attacks, adversarial parties try to break the privacy of honest parties by exploiting the network connection or physical weaknesses of the honest parties' machine.

This article provides an overview of the results for achieving secure computation in presence of concurrent and physical attacks contained in the phD thesis:"Secure Computation under concurrent and physical attacks", with emphasis to the relation of such results with the state of the art.

1 Introduction

The setting of secure computation is the following. There are many entities, that we call parties. Parties are mutually distrustful, and want to jointly compute a function of their inputs while keeping such inputs secret. As an example of this setting, consider a voting system. Each party has as secret input a preference, i.e., the name of the person that the party wants to vote. The joint function run by the parties takes in input all the secret preferences (the votes) and computes the majority. As it is already apparent from this example, besides the correctness of the result, an important property that we require from a voting system is that the preference of each player remains private.

Informally, a protocol for function evaluation is secure if each player does not learn anything more than what is leaked from the knowledge of the output and the knowledge of its own private input, even if the party actively deviate from the protocol.

How can we formally model the concept of not learning anything in this setting? In the seminal work [39] Goldwasser et al. introduce the concept of the simulator. The idea is that anything that an adversary can learn by interacting

with the actual parties participating in the protocol, it can be learned also by a simulator, that is not interacting with any real party but it simulates the protocol executions in his head using fake inputs (for the honest parties). Very informally, to prove that a protocol is secure, is sufficient to show that for any adversarial party attacking the protocol in a real world execution run with the honest parties, there exists a simulator that successfully performs the very same attack, but executing the protocol in his head, without knowing the input of the honest players, and having access to an oracle that computes the function.

The above definition of secure computation is a good starting point, but it is still very far from capturing real-life setting. Indeed, in the setting above we have considered a very restricted adversary, that participates to one protocol execution only, and that can misbehave by corrupting a subset of parties. Thus, proving security in this setting, implies that a protocol will be secure as long as it will be run in isolation, namely parties cannot be involved in the execution of other protocols.

However, in real-life scenarios many functionalities are run over the internet. Thus, an adversary does not necessarily run one execution of the protocol in isolation, but it can activate and interleave many executions of other (or the same) protocols. Hence, in the formal security definition one should model the adversary as an entity that initiates and actively participates in many protocol executions, that can be run concurrently.

Such general security definition was introduced in [12] and is called Universal Composability. In the Universal Composability framework, besides the simulator and the adversary seen before, a new entity called "environment" is introduced. While the simulator and the adversary still model one execution of the target protocol for which one wants to prove security, the purpose of the environment is to model the concurrent executions of other (possibly different) protocols, that are run together with the target protocol. The environment can of course communicate with the adversary. This models the fact that while attacking the target protocol, the adversary can exploit the information gained from the concurrent executions. The Universal Composability (UC) framework is very general, and thus provides very strong security guarantees. Therefore, is desirable to have protocols that can be proved secure in this model. Unfortunately, it has been shown in [17] that it is impossible to design a protocol that achieves such definition, without the help of some setup assumption. A setup can be seen as some tool that parties can use to run the protocol. A setup is *trusted* if it is assumed that the adversary cannot participate in the generation of such setup. For example, a trusted setup can be that each party is given a smart card, and the adversary cannot participate in the process of generating/delivering such smart cards. Therefore, the behavior of such smart card is never adversarial.

Given that the UC security cannot be achieved without assuming some setup,

two lines of research have been explored. One line of research investigated on the possibility of relaxing the need of trust in the setup assumptions required to achieve UC-security. For example, in the setup that we discussed above, the assumption was that each party receives a trusted smart-card. However, assuming that all the smart cards, even the ones used by the adversary, are trusted seems unrealistic, and it seems indeed natural to ask whether we can *reduce* the amount of trust needed to achieve UC-security. This line of research, started with [46], explores the use of hardware boxes (formally, tamper-proof hardware tokens) in the protocol. The idea is that, each party must trust only its own hardware box, and not the boxes generated/delivered by the other players.

The second line of research instead investigates on relaxing the UC-definition so that it is possible to design protocols without using any trusted setup. Such relaxed definitions capture a restricted scenario in which an adversary activates many executions of the *same* protocol simultaneously, playing always the *same* role (in contrast in the UC-definition the adversary can execute arbitrary protocols concurrently, and play different roles). We refer to this more relaxed definition as security in the concurrent setting, and to the adversary playing in this setting, as a concurrent adversary. Besides the mere feasibility results, this line of research develops on understanding the minimal requirements, such as round and communication complexity, computation complexity, black-box uses of cryptographic primitive, for a protocol which is secure in the concurrent setting.

Overview of the Results

In the thesis we provide contributions for both lines of research.

First, we discuss how to achieve UC-security based on physical setup assumptions while removing the trust in any third party, and the trust on the physical devices used by the adversary. We explore the use of Physically Uncloneable Functions (PUFs) as setup assumption for achieving UC-secure computations. PUF are physical noisy source of randomness. The use of PUFs in the UC-framework has been proposed already in [10]. However, [10] assumes that all PUFs in the system are *trusted*. This means that, each party has to trust the PUFs generated by the other parties. We focus on reducing the trust involved in the use of such PUFs and we introduce the Malicious PUFs model in which only PUFs generated by honest parties are assumed to be trusted. Thus the security of each party relies on its own PUF only and holds regardless of the goodness of the PUFs generated/used by the adversary. We are able to show that, under this more realistic assumption, one can achieve UC-secure computation, under computational assumptions. Moreover, we show how to achieve *unconditional* UC-secure commitments with (malicious) PUFs and with stateless tamper-proof hardware tokens. These results are discussed in more details in Section 2.

Secondly, we restrict our attention to the concurrent setting. We investigate on protocols which are concurrently secure and enjoy *round optimality* and *black-box* access to a cryptographic primitive. We study two fundamental functionalities: commitment scheme and zero knowledge, and we study two concurrent attack models, as explained below.

Commitment scheme is a two-stage (commitment, decommitment stage) functionality run between a committer and a receiver. The committer has a secret value in input, and it commits to such value running the commitment stage. When the committer is ready to reveal the value to the receiver, it runs the decommitment stage, also called opening stage. The security properties of a commitment scheme are hiding and binding. Hiding preserves the security of the committer, and is the property of the commitment stage. A commitment stage is hiding if the secrecy of the input committed is preserved against any adversarial receiver. Binding is a correctness property of the commitment scheme, and it requires that for a given transcript of the commitment stage, there exists only one value that can be revealed by any, possible adversarial, committer. We consider the following attack model for commitment schemes. An adversarial receiver can interleave arbitrarily many commitment sessions, and once the commitment stages are over, ask for the decommitment (the opening) of some of them, adaptively on the transcript observed from the sessions played so far and in any order. Note that this adversary is not purely concurrent, since it can ask to open a session only if the commitment stage of any other sessions is already completed. Namely, there is a barrier between the commitment phase and the openings. We refer to this type of concurrency as "concurrency with barrier". This attack model is referred in literature as Selective Opening Attack (SOA, in short) and was introduced in [32]. A commitment scheme secure in this model is said to be SOA-secure.

A Zero Knowledge protocol is run between two parties, a prover and a verifier. Both parties have as common input an instance x of an **NP** language L. The prover has as input a witness w for the statement $x \in L$, and he wants to use knowledge of the witness to convince the verifier that the statement is true, however, he does not want to reveal any information about the witness. Security here means that any (concurrent) adversarial verifier running a protocol execution with the prover on input x should not gain any information besides the fact that $x \in L$. The correctness requirement establishes that if the statement is false, i.e., $x \notin L$, any adversarial (possibly concurrent) prover should not be able to convince a honest verifier of the truthfulness of the statement. In this thesis we consider the following attack model. An adversarial verifier can initiate any polynomial number of protocol sessions in concurrency, but it is bounded on the number of identities that it can play with. More precisely, we consider a setting in which any verifier that wishes to run the zero knowledge protocol with the prover, has to register its identity in a public file before any proof begins. Thus, there is a registration phase,

in which no interaction between prover and verifier takes place, but each verifier upload its public identity in a common public file. After the registration has been completed, the "proof phase" starts. In this phase prover and registered verifiers run concurrently many proofs. The restriction is that the proof phase can be run only by registered verifiers. This model is called Bare Public Key (BPK, for short) model, and was introduced in [15].

Our results concern some of the round-optimal constructions and lower bounds presented in the literature for both SOA-secure commitments and zero-knowledge protocols in the BPK model. We show that some of the proposed constructions present subtle issues. Then we propose new protocols that meet the security guarantees promised by protocols proposed in literature. We describe these results in more details in Section 3.

Physical Attacks

So far we have discussed about network attacks. Namely, we have considered an adversary which exploits the network to run concurrent executions of different protocols. We now turn to physical attacks. Namely, we consider an adversary that can physically tamper with the machine of the honest party while running the protocol. In particular, in this thesis we consider a very specific attack in which the adversary is able to *reset* the memory of the machine of an honest party, in fact forcing the machine to run many executions of the same protocol *reusing* the same randomness. At first sight, this attack might seem too unrealistic, as one can imagine that the honest party can physically protect its own machine from the adversary. Therefore, considering such attack when designing a protocol, might seem an overkill. However, as nowadays tiny and weak computational devices are used in cryptographic protocols, such attack has become a real threat. Smart cards are the canonical example of tiny devices computing (sensitive) cryptographic protocol. Indeed, we use smart card everyday to perform bank transactions or for identification purposes. The way computations with smart card work, is that we put our smart card in some more powerful computing device, like a smart-card reader, or a PC, and they run a protocol. In this scenario resetting attacks seem very plausible, since a smart card is a very weak device compared to a reader or to a computer.

The security definition that formally captures this kind of attack has been introduced in [15]. This definition focuses only on the zero knowledge functionality (and the weaker notion of witness indistinguishability), and considers only the case in which only one party is reset. More precisely, the party which is subject to reset is always the prover. Thus, the constructions shown in [15] are secure if the prover is subject to reset attack, but is not secure if the verifier is instead the victim of reset. Later in [6] resettable security has been formulated for the oppo-

site case. Namely, [6] provides constructions that are secure if the party that is subject to reset is only the verifier. Clearly, an interesting question is whether one can design a protocol in which the party that might be subject to reset attack is not known in advance. Therefore, such protocol should be secure in both cases. Such notion is called security under *simultaneous resettability*. The work of [22] provides a protocol that is secure in such setting. Namely, it presents a construction for simultaneously resettable zero knowledge system. However this result does not close the gap between what we are able to achieve in presence of a fixed resetting party, and what we can achieve when both parties can reset. Indeed, in case of one-side resetting (either only the prover can reset, or only the verifier can reset), we know how to achieve protocols that are *arguments of knowledge*. Roughly, a protocol is an argument of knowledge, if the prover can convince the verifier of a truthfulness of a statement, if and only if it knows the witness.

In the thesis we provide the first construction of a witness indistinguishable argument system that is *simultaneous resettable* and *argument of knowledge*. We discuss about this contribution in Section 4.

2 UC-security from Malicious PUFs

In this section we discuss our results in the Universal Composability framework. Such results can be found in the articles [59] and [82]. In the following, we first discuss the state of the art of Universally Composable secure computation and then we outline our contribution.

Universal Composability using Physical Assumptions. The universal composability framework was introduced by Canetti in [14] and captures the most general security notion considering an adversary that run many concurrent execution of arbitrary protocols. Such general security notion is, unfortunately, impossible to achieve in the plain model, as it was first proved in Canetti and Fischlin [14] and then strengthened by Canetti et al. in [17]. As a consequence, several setup assumptions, and relaxations of the UC framework have been proposed to achieve UC security [18, 5, 64, 45].

In recent years, researchers have started exploring the use of secure hardware in protocol design. The idea is to achieve protocols with strong security guarantees (like UC) by allowing parties to use hardware boxes that have certain security properties. An example of the kind of security required from such a hardware box is that of *tamper-proofness*; i.e., the receiver of the box can only observe the input/output behaviour of the functionality that the box implements. This property was formalized by Katz in [46], and it was shown that UC security is possible by relying on the existence of tamper-proof programmable hardware tokens, and

computational assumptions. Smart cards are well understood examples of such tokens, since they have been used in practice in the last decades. Several improvements and variations of Katz's model have been then proposed in follow up papers (e.g., [19, 56, 38, 40, 29, 21, 30]).

Spurred by technological advances in manufacturing, recently a new hardware component has gained a lot of attention: Physically Uncloneable Functions (PUFs) [61, 60]. A PUF is a hardware device generated through a special physical process that implements a "random" function[1] that depends upon the physical parameters of the process. These parameters can not be "controlled", and producing a clone of the device is considered infeasible. Once a PUF has been constructed, there is a physical procedure to query it, and to measure its answers. The answer of a PUF depends on the physical behavior of the PUF itself, and it is assumed to be unpredictable, or to have high min-entropy. Namely, even after obtaining many challenge-response pairs, it is infeasible to predict the response to a new challenge.

Since their introduction by Pappu in 2001, PUFs have gained a lot of attention for cryptographic applications like anti-counterfeiting mechanisms, secure storage, RFID applications, identification and authentication protocols [71, 43, 69, 34, 50]. More recently PUFs have been used for designing more advanced cryptographic primitives. In [67] Rührmair shows the first construction of Oblivious Transfer, the security proof of which is later provided in [68]. In [4], Armknecht et al. deploy PUFs for the construction of memory leakage-resilient encryption schemes. In [51] Maes et al. provide construction and implementation of PUFKY, a design for PUF-based cryptographic key generators. There exist several implementations of PUFs, often exhibiting different properties. The work of Armknecht et al. [3] formalizes the security features of physical functions in accordance to existing literature on PUFs and proposes a general security framework for physical functions. A survey on PUF implementations is given in [52]. Very recently in [47] Katzenbeisser et al. presented the first large scale evaluation of the security properties of some popular PUFs implementations (i.e., intrinsic electronic PUFs).

Modeling PUFs in the UC framework. Only very recently, Brzuska et al. [10] suggested a model for using PUFs in the UC setting that aims at abstracting real-world implementations. The unpredictability and uncloneability properties are modeled through an ideal functionality. Such functionality allows only the creation of trusted PUFs. In [10] PUFs are thought as non-PPT setup assumptions. As such, a PPT simulator cannot simulate a PUF, that is, PUFs are non-

[1] Technically, a PUF does not implement a function in the mathematical sense, as the same input might produce different responses.

programmable. Although non-programmable, PUFs are not modeled as global setup [13]. [10] shows how to achieve unconditional UC secure Oblivious Transfer, Bit Commitment and Key Agreement with trusted PUFs.

PUFs vs tamper-proof hardware tokens. The apparent similarity of PUFs with programmable tamper-proof tokens [46] vanishes immediately when one compares in detail the two physical devices. Indeed, PUFs are non programmable and thus provide unpredictability only. Instead tokens are programmable and can run sophisticated code. Moreover, PUFs are stateless, while tokens can be stateful. When a PUF is not physically available, it is not possible to know the output of new queries it received. Instead the answer of a stateless token to a query is always known to its creator[2], since it knows the program embedded in the token. Tamper-proof tokens are realized through ad-hoc procedures that model them as black boxes, their internal content is protected from physical attacks and thus the functionalities that they implement can be accessed only through the prescribed input/output interface provided by the token designer. Instead, PUFs do not necessarily require such a hardware protection and their design is associated to recommended procedures to generate and query a PUF, guaranteeing uncloneability and unpredictability. Finally, in contrast to tokens that correspond to PPT machines, PUFs are not simulatable since it is not clear if one can produce an (even computationally) indistinguishable distribution.

2.1 Our contribution

We continue the line of research started by Brzuska et al. investigating more on the usability of PUFs to obtain UC-secure computation. We observe that the UC formulation of PUFs proposed by Brzuska et al. considers only *trusted* PUFs. This means that, it is assumed that an adversary is be unable to produce fake/malicious PUFs. We believe that making such assumption might be unrealistic. Given that the study of PUFs is still in its infancy, it is risky to rely on assumptions on the impossibility of the adversaries in generating PUFs adversarially.

Our main contribution consists in studying the security of protocols in presence of adversaries that can create malicious PUFs. We present a modification of the model of Brzuska et al. that formalizes security with respect to such stronger adversary and we give positive answers to the question of achieving universally composable secure computation with PUFs. More in details, our contributions are listed below.

[2] This is true for stateful tokens too, provided that one knows the sequence of inputs received by the token.

The Malicious PUFs Model. We generalize the model of Brzuska et al. so to enable the adversary to create untrusted (malicious) PUFs. But what exactly are malicious PUFs? In real life, an adversary could tamper with a PUF in such a way that the PUF loses any of its security properties. Or the adversary may introduce new behaviours; for example, the PUF could keep a state. To keep the treatment of malicious behaviour as general as possible, we allow the adversary to send as PUF any hardware token that meets the syntactical requirements of a PUF. Thus, an adversary is assumed to be able to even produce fake PUFs that might be stateful and programmed with malicious code. We assume that a malicious PUF however cannot interact with its creator once is sent away to another party. If this was not the case, then we are back in the standard model, where UC security is impossible to achieve has argued below.

The impossibility is straight forward. Consider any functionality that protects the privacy of the input of a player P_1. Comparing to the plain model (where UC is impossible), the only advantage of the simulator to extract the input of the real-world adversary P_1^*, is to read the challenge/answer pairs generated by P_1^* when using the honest PUF created by the simulator that plays on behalf of P_2. If such a simulator exists, then an adversary P_2^* can generate a malicious PUF that just plays as proxy and forwards back and forth what P_2^* wishes to play. P_2^* can locally use one more honest PUF in order to compute the answers that the (remote) malicious PUF is supposed to give. Clearly P_2^* will have a full view of all challenge/response pairs generated by honest P_1 and running the simulator's code, P_2^* will extract the input of P_1, therefore contradicting input privacy.

General UC-secure computation in the Malicious PUFs Model. The natural question is whether UC security can be achieved in such a much more hostile setting. We give a positive answer to this question by constructing a *computational* UC-secure commitment scheme in the malicious PUFs model. Our commitment scheme needs two PUFs that are transferred only once (PUFs do not go back-and-forth), at the beginning of the protocol and it requires computational assumptions. We avoid that PUFs go back-and-forth by employing a technique that requires OT. The results of Canetti, et al. [18] shows how to achieve general UC computation from computational UC commitments.

Hardness assumptions with PUFs. Notice that as correctly observed in [10], since PUFs are not PPT machines, it is not clear if standard complexity-theoretic assumptions still hold in presence of PUFs. We agree with this observation. However the critical point is that even though there can exist a PUF that helps to break in polynomial time a standard complexity-theoretic assumptions, it is still unlikely that a PPT adversary can find such a PUF. Indeed a PPT machine can only generate a polynomial number of PUFs, therefore obtaining the one that allows to

break complexity assumptions is an event that happens with negligible probability and thus it does not effect the concrete security of the protocols. In light of the above discussion, only one of the following two cases is possible. 1) Standard complexity-theoretic assumptions still hold in presence of PPT adversaries that generate PUFs; in this case our construction is secure. 2) There exists a PPT adversary that can generate a PUF that breaks standard assumptions; in this case our construction is not secure, but the whole foundations of complexity-theoretic cryptography would fall down (which is quite unlikely to happen) with respect to real-world adversaries.

Unconditional UC-secure Commitment Scheme with Malicious PUFs. We furthermore provide a tool for constructing UC-secure commitments given any straight-line extractable commitment. This tool allows us to prove feasibility of unconditional UC-secure protocols (for a non-trivial functionality) in the malicious PUF model. More precisely, we provide a compiler that transforms any ideal extractable commitment – a primitive that we define – into a UC-secure commitment. An ideal extractable commitment is a statistically hiding, statistically binding and straight-line extractable commitment. The transformation uses the ideal extractable commitment as black-box and is unconditional. The key advantage of such compiler is that, one can implement the ideal extractable commitment with the setup assumption that is more suitable with the application and the technology available.

We then construct an extractable commitment scheme in the malicious PUFs model. By plugging such scheme into the compiler we obtain the first unconditional UC-secure commitment with malicious PUFs. However, whether *general* secure computation with unconditional UC security is possible with (malicious) PUFs, remains an interesting open question.

3 Round-optimal Concurrently Secure Protocols

In this section we discuss our results concerning achieving round-optimal protocols for concurrent Zero Knowledge and Commitment Scheme secure under Selective Opening Attacks in the concurrent setting. Such results can be found in the articles [70] and [58] respectively. In the following, we first discuss the state of the art of the respective problems and then we outline our contribution.

3.1 Round-Optimal Concurrent ZK in the Bare Public Key model

The notion of concurrent zero knowledge (*cZK*, for short) introduced in [33] deals with proofs given in asynchronous networks controlled by the adversary.

In [15] Canetti et al. studied the case of an adversary that can reset the prover, forcing it to re-use the same randomness in different executions. They defined as resettable zero knowledge (r*ZK*, for short) the security of a proof system against such attacks. Very interestingly, r*ZK* is proved to be stronger than *cZK*.

Motivated by the need of achieving round-efficient r*ZK*, in [15] the Bare Public-Key (BPK, for short) model has been introduced, with the goal of relying on a setup assumption that is as close as possible to the standard model. Indeed, round-efficient *cZK* and r*ZK* are often easy to achieve in other models (e.g., with trusted parameters) that unfortunately are hard to justify in practice.

The BPK model. The sole assumption of the BPK model is that when proofs are played, identities of (polynomially many) verifiers interacting with honest provers are fixed. For instance, identities could be posted to a public directory so that players can download the content of the directory before proofs start. This registration phase is non-interactive, does not involve trusted parties or other assumptions, and can be fully controlled by any adversary. When proofs start, it is assumed that honest provers interact with registered verifiers only.

The BPK model is very close to the standard model, indeed the proof phase does not have any requirement beyond the availability of the directory to all provers, and for each verifier, of a secret key associated to his identity. Moreover, in both phases the adversary has full control of the communication network, and of corrupted players.

Round-optimal *cZK* in the BPK model from r*ZK*. The first constant-round r*ZK* (and thus *cZK*) argument for **NP** in the BPK model has been given in [15]. Then in [54] it is pointed out the subtle separations among soundness notions in the BPK model. Indeed, in contrast to the standard model, the notions of one-time, sequential and concurrent soundness, are distinct in the BPK model. In [54] it is then showed that the proof of [15] is actually sufficient for sequential soundness only. Moreover in [54] it is proved that 4 rounds are necessary for concurrent soundness and finally, they showed a 4-round r*ZK* (and thus *cZK*) argument with sequential soundness. The protocol is "conversation based", i.e., by simply observing the transcript one can compute the output of the verifier. In light of the impossibility proved by [1] (i.e., there exists no 3 round sequentially sound *cZK* conversation-based argument in the BPK model for non-trivial languages)

the above 4-round rZK (and thus cZK) argument is round optimal.

Concurrent soundness along with rZK (and thus cZK) was achieved in [24], requiring 4 rounds. Further improvements on the required complexity assumptions have been showed in [79] where a 4-round protocol under generic assumptions and an efficient 5-round protocol under number-theoretic assumptions are shown.

The above mentioned results on constant-round rZK/cZK in the BPK model rely on the assumptions that some cryptographic primitives are secure against sub-exponential time adversaries (i.e., complexity leveraging) and obtained black-box simulation.

Round-optimal cZK in the BPK model under standard assumptions. The question of achieving a constant-round black-box cZK in the BPK model without relying on complexity leveraging has been first addressed in [80] and then in [26]. The protocol of [80] needs 4 rounds and achieves sequential soundness only. The protocol given in [26] also needs 4 rounds and achieves concurrent soundness. A follow up result of [72] showed an efficient transformation that starting from a language admitting a Σ-protocol produces a cZK argument with concurrent soundness needing only 4 rounds and adding only a constant number of modular exponentiations. A more recent result [23] obtains both round optimality and optimal complexity assumptions (i.e., the need of One-way Functions only) for concurrently sound cZK. The notion of "knowledge extraction" has been studied in [27] and in [77, 76] where in particular concurrent knowledge extraction (under different formulations) is considered.

All above results achieve cZK and are based on hardness assumptions with respect to polynomial-time adversaries.

3.1.1 Our Contribution

In the thesis we show subtle problems concerning security proofs of various cZK and rZK arguments in the BPK model [54, 80, 24, 26, 72, 79, 23, 77], including *all* round-optimal constructions published so far.

The source of the problem: parallel execution of different sub-protocols. In order to achieve round efficiency, various known protocols, including all round-optimal protocols, consist in parallel executions of sub-protocols that are useful in different ways in the proofs of soundness and cZK/rZK. Roughly speaking, there is always a sub-protocol π_0 where in 3 rounds the verifier is required to use a secret related to its identity. Then there is a 3-round sub-protocol π_1 in which the prover convinces the verifier about the validity of the statement and the simulator can do the same by using knowledge of a secret information obtained by rewinding π_0

(in the current session or in other sessions corresponding to the same identity). To obtain a 4-round protocol[3], π_1 starts during the second round of π_0. Such round combination yields one of the following two cases.

The first case is when the simulator needs the secret information already to compute the first message of π_1 so that such a message can appear in the final transcript of the simulation. In this case when the simulator runs protocol π_1 for the first time with a given identity, it first needs to extract the secret related to such identity, used in π_0 by the verifier. The use of look-ahead threads (i.e., trying to go ahead with a virtual simulation with the purpose of obtaining the required information needed in the main thread of the simulation) would not help here since only a limited polynomial amount of work can be invested for them, and there is always a non-negligible probability that look-ahead threads fail, while still in the main thread the verifier plays the next message. Given the above difficulty, the simulator needs to play a *bad* first round in π_1 so that later, when the needed secret information is extracted from π_0, the simulator can play again such first round of π_1, this time with a *good* message. However, this approach suffers of a problem too. Indeed, aborting the main thread and starting a new thread leads to a detectable deviation in the final transcript that the simulator will output. Indeed, the fact that the simulator gives up with a thread each time it is stuck, and then starts a new one, skews the distribution of the output of the simulator, since the output will then include with higher probability threads that are "easier" to complete (e.g., where the simulator does not get stuck because new sessions for new identities do not appear). Notice that this issue motivates the simulation strategies adopted in previous work on *cZK* (e.g., [66, 63]) where the main thread corresponds to the construction of the view that will be given in output, while other threads are started with the sole purpose of extracting secrets useful to complete the main thread.

We now consider the second case where the simulator does not need any secret to compute the first round of π_1. We observe that this approach could hurt the proof of concurrent soundness, when the latter is proved by means of witness extraction[4] from π_1. Indeed, a malicious concurrent prover can exploit the execution of π_0 in a session j, for completing the execution of π_1 in another concurrent session $j' \neq j$ by playing a man-in-the-middle attack such that, when (in the proof of concurrent soundness) one tries to reach a contradiction by extracting the witness from the proof π_1 given in session j', it instead obtains the secret used to run π_0 in session j. Instead, if the secret to be extracted from π_1 is fixed from the very first round of π_1, then one can show that it is either independent from the one used in session

[3]Similar discussions hold for some 5-round protocols when π_0 requires 4 rounds.

[4]We note that all constructions of *cZK* in the BPK model under standard assumptions prove soundness by means of witness extraction.

j (this happens when the secret is used in π_0 of session j after the first round of π_1 in session j' is played, and the secret used by the verifier can not be predicted with non-negligible probability), or is dependent but not affected by the rewind of session j' (this happens when the secret is used in π_0 of session j before the first round of π_1 in session j' is played).

The use of the secret in the last round of π_1 only, could instead be helpful in the following three cases: I) when one is interested in rZK since in this case soundness is proved through a reduction based on complexity leveraging (no need for rewinding); II) when cZK with sequential soundness only is desired; III) when the secret needed by the simulator when running π_1 in a session j' is different from the witness used by the verifier in the execution of π_0 in other sessions. In these three cases the above discussion does not necessarily apply. Indeed some proposed round-optimal protocols that fall in one of such cases, might still be secure even though their security proofs seem to ignore at least in part the problems that we are pointing out.

Because of the above case I), we believe that achieving 4-round cZK with concurrent soundness in the BPK model under standard assumptions is definitively harder than obtaining 4-round rZK with concurrent soundness in the BPK model through complexity leveraging. Therefore, in this thesis we focus on achieving cZK and this will require a new technique.

We stress that in all previous constructions, one could obtain a different protocol that satisfies the desired soundness and zero-knowledge properties by simply running π_0 and π_1 sequentially. Indeed, in this case the simulator can complete π_0 in the main thread, then run the extractor in another thread, and finally continue the main thread running π_1 having the secret information. We also stress that all papers that we revisit in the thesis, achieved also other results that are not affected by our analysis when round optimality is not desired.

We finally note that we did not investigate other round-efficient results in variations of the BPK model [53, 81, 25], and other results in the BPK model that do not focus on optimal round complexity [57, 78, 20].

New techniques for round-optimal cZK in the BPK model. In this thesis we show a protocol and a security proof that close the gap in between lower and upper bounds for the round complexity of concurrently sound cZK in the BPK model under standard assumptions. The result is achieved by using a new technique where in addition to the (permanent) secret associated to the identity of the verifier, there is a *temporary* secret per session, that enables the simulator to proceed in two modes as follows. Knowledge of the permanent secret of the verifier allows the simulator to proceed in straight-line in the main thread in sessions started after the extraction of the permanent secret. Knowledge of the temporary

secret allows the simulator to solve the sessions started before the extraction of the permanent secret, by launching rewinding threads but without changing the main thread. The temporary key is extracted through rewinding threads, and it is used only when computing the last prover message of a session of the main thread, i.e., only after the extraction has been completed. This allows to keep the main thread unchanged. In the rewinding threads the simulator is always straight-line. We implement both the permanent and the temporary keys by means of trapdoor commitments. The proof of *cZK* will be tricky since it requires the synergy of the two above simulation modes. In our case the number of extraction procedures required to carry on the simulation is not bounded by the number of identities registered in the directory (in contrast with the main technique used in the past in the BPK model), but by the number of sessions. The proof of concurrent soundness also requires special attention. Indeed while the interplay of temporary and permanent secrets helps the simulator, it could also be exploited by the malicious prover.

Finally, we show that our *cZK* protocol admits a transformation that starting from any language admitting a perfect Σ-protocol, produces round-optimal concurrently-sound *cZK* protocol. Such transformation requires a constant number of modular exponentiations only, and the final protocol is secure under standard number-theoretic assumptions.

3.2 Round-Optimal SOA-secure Commitments

Commitment schemes are a fundamental building block in cryptographic protocols. By their usual notion, they satisfy two security properties, namely, hiding and binding. While the binding property guarantees that a committed message can not be opened to two distinct messages, the hiding property ensures that before the decommitment phase begins, no information about the committed message is revealed. Binding and hiding are preserved under concurrent composition, in the sense that even a concurrent malicious sender will not be able to open a committed message in two ways, and even a concurrent malicious receiver will not be able to detect any relevant information about committed messages as long as only commitment phases have been played so far.

In [32], Dwork et al. pointed out a more subtle definition of security for hiding where the malicious receiver is allowed to ask for the opening of only some of the committed messages, with the goal of breaking the hiding property of the remaining committed messages. This notion was captured in [32] via a simulation-based security definition, and is referred to as hiding in presence of selective opening attack (SOA, for short). [32] shows that, in a *trusted setup* setting, it is possible to construct a non-interactive SOA-secure commitment scheme from a trapdoor commitment scheme. Indeed, in the trusted setup the simulator sets the parameters

of the trapdoor commitment, thus obviously it knows the trapdoor. However, the fundamental question of whether there exist SOA-secure commitment schemes in the *plain model*, is left open in [32]. We stress that the question is particularly important since commitments are often used in larger protocols, where often only some commitments are opened but the security of the whole scheme still relies on hiding the unopened commitments. For instance, the importance of SOA-secure commitments for constructing zero-knowledge sets is discussed in [36][5].

The SOA-security experiment put forth in [32] considers a one-shot commitment phase, in which the receiver gets all commitments in one-shot, picks adaptively a subset of them, and obtains the opening of such subset. Such definition implicitly considers non-interactive commitments and only parallel composition. Subsequent works have explored several extensions/variations of this definition showing possibility and impossibility results. Before proceeding to the discussion of the related work, it is useful to set up the dimensions that will be considered. One dimension is *composition*. As commitment is a two-phase functionality, other than parallel composition, one can consider two kinds of concurrent composition. Concurrent-with-barrier composition (considered in [9, 44]), refers to the setting in which the adversarial receiver can interleave the execution of several commitments, and the execution of decommitments, with the restriction that all commitment phases are played before any decommitment phase begins. Thus, there is a barrier between commitment and decommitment stage. Fully-concurrent composition (considered in [73]) refers to the setting in which the adversary can arbitrarily interleave the execution of the commitment phase of one session with the decommitment of another session (and vice-versa).

Next dimension is the *access to primitive*, namely, if the construction uses a cryptographic primitive as a black-box (in short, BB), or in a non black-box way (in short, NBB).

Another dimension is *simulation*. In this discussion we consider always black-box simulators (if not otherwise specified).

The question of achieving SOA-secure commitments without any set-up was solved affirmatively in [9] by Bellare, Hofheinz, and Yilek, and in Hofheinz [44], who presented an interactive SOA-secure scheme based on non-black-box use of any one-way permutation and with a commitment phase requiring a super-constant number of rounds. The security of such construction is proved in the concurrent-with-barrier setting. [9, 44] also shows that non-interactive SOA-secure commitments which use cryptographic primitives in a black-box way do not exist. The same work introduces the notion of *indistinguishability* under selective opening attacks, that we do not consider in the thesis. The results of [9, 44]

[5] In [36] some forms of zero-knowledge sets were proposed, and their strongest definition required SOA-secure commitments.

left open several other questions on round optimality and black-box use of the underlying cryptographic primitives.

In TCC 2011 [73], Xiao addressed the above open questions and investigated on how to achieve nearly optimal schemes where optimality concerns both the round complexity and the black-box use of cryptographic primitives. In particular, Xiao addressed SOA-security of commitment schemes for both parallel composition and fully-concurrent composition and provided both possibility and impossibility results, sticking to the simulation-based definition. Concerning positive results, [73] shows a 4-round (resp., $(t+3)$-round for a t-round statistically-hiding commitment) computationally binding (resp., statistically binding) SOA-secure scheme for parallel composition. Moreover, [73] provides a commitment scheme which is "strong" (the meaning of strong is explained later) SOA-secure in the fully-concurrent setting and requires a logarithmic number of rounds. All such constructions are fully black-box. Concerning impossibility results, [73] shows that 3-round (resp., 4-round) computationally binding (resp., statistically binding) parallel SOA-secure commitment schemes are impossible to achieve. As explained later, in our work we present some issues affecting the proof of security of the constructions shown in [73]. We also show that, the strong security claimed for the construction suggested for the fully-concurrent setting, is actually impossible to achieve, regardless of the round complexity. We contradict the lower bounds claimed in [73] by providing a 3-round fully black-box commitment scheme which is SOA-secure under concurrent-with-barrier composition, which implies parallel composition.

In a subsequent work [75] Xiao showed a black-box construction of 4-round statistically binding commitment which is SOA-secure under parallel composition.

The work [74] provides an updated version of [73]. Concerning positive results, [74] includes the $(t+3,1)$-round construction of [73] and shows a new simulation strategy for it. Concerning impossibility results, [74] includes the lower bounds of [73] that are still valid for 2-round (resp., 3 round) computationally hiding and computationally (resp., statistically) binding, parallel SOA-secure commitment scheme with black-box simulators. [74] contains also other contributions of [73] that are not contradicted by our results.

In [7], Bellare et al. proves that existence of CRHFs implies impossibility of non-interactive SOA-secure commitments (regardless of the black-box use of the cryptographic primitives). In fact, they show something even stronger; they show that this impossibility holds even if the simulator is non-black-box and knows the distribution of the message space. An implication of such results is that, standard security does not imply SOA-security. Previous results in [9, 44] only showed the impossibility for the case of black-box reductions.

In [62], Pass and Wee provide several black-box constructions for two-party

protocols. Among other things, they provide constructions for look-ahead trapdoor commitments (in a look-ahead commitment, knowledge of the trapdoor is necessary already in the commitment phase in order for the commitment to be equivocal), and trapdoor commitments. Such constructions have not been proved to be SOA-secure commitment schemes, as SOA-security is proven in presence of (at least) parallel composition, while security of the trapdoor commitment of [62] is proved only in the stand-alone setting.

3.2.1 Our Contribution

We focus on simulation-based SOA-secure commitment schemes, and we restrict our attention to black-box simulation, and (mainly) black-box access to cryptographic primitives (like in [73]). Firstly, we point out various issues in the claims of [73]. These issues essentially re-open some of the open questions that were claimed to be answered in [73]. We next show how to solve (in many cases in a nearly optimal way) all of them. Interestingly, our final claims render quite a different state-of-the-art from (and in some cases also in contrast to) the state-of-the-art set by the claims of [73].

In detail, by specifying as (x, y) the round complexity of a commitment scheme when the commitment phase takes x rounds and the decommitment phase takes y rounds, we revisit some claims of [73] and re-open some challenging open questions as follows.

1. The proof in [73] of the non-existence of $(3, 1)$-round schemes assumes implicitly that the sender sends the last message during the commitment phase. We show that surprisingly this assumption is erroneous, and that one round might be saved in the commitment phase if the receiver goes last. This re-opens the question of the achievability of $(3, 1)$-round SOA-secure schemes, even for just parallel composition.

2. The proof of binding and SOA-security of the $(4, 1)$-round scheme of [73] for parallel composition presents a sublet issue, and it is currently unknown whether the scheme is secure. The same issue in the SOA-security proof exists for the $(t + 3, 1)$-round statistically binding scheme of [73] which is based on any t-round statistically-hiding commitment. Indeed, for both constructions, SOA-security is claimed to follow from the simulation technique of Goldreich-Kahan [37]. The problem is that the simulator of [37] was built for a *stand-alone* zero-knowledge protocol where an atomic sub-protocol is repeated several times in parallel, and the verifier cannot *selectively* abort one of the sub-protocols. Instead in the SOA-setting the adversarial receiver interacts with multiple senders and can decide to abort only a subset of the sessions of its choice adaptively based on the commitment-phase transcript.

3. The proof of security of the fully-concurrent SOA-secure commitment proposed in [73] presents a subtle issue. The security of such construction is claimed even for the case in which the simulator cannot efficiently sample from the distribution of messages committed to by the honest sender (but needs to query an external party for it). This notion is referred in [73] as "strong" security. This issue in [73] re-opens the possibility of achieving schemes that are strong SOA-secure under fully concurrent composition (for any round complexity).

In the thesis we solve the above open problems (still sticking to the notion of black-box simulation as formalized in [73]) as follows.

1. We present a $(3, 1)$-round scheme based on BB use of any trapdoor commitment (TCom, for short), which contradicts the lower bound claimed in [73].

   We also show a $(4, 1)$-round scheme based on BB use of any weak trapdoor commitment (wTCom, for short)[6].

2. We show that when the simulator does not know the distribution of the messages committed to by the honest sender, there exists no scheme that achieves fully concurrent SOA-security, regardless of the round complexity and of the BB use of cryptographic primitives. Thus contradicting the claimed security of the construction given in [73].

3. As a corollary of our $(3, 1)$-round scheme based on BB use of any TCom, there exists a $(3, 1)$-round scheme based on NBB use of any one-way function (OWF). This improves the round complexity in [9] from logarithmic in the security parameter to only 3 rounds and using minimal complexity-theoretic assumptions. Moreover, we observe that (as a direct consequence from proof techniques in [73]) a $(2, 1)$-round SOA-secure scheme is impossible regardless of the use of the underlying cryptographic primitive (for black-box simulation only). Thus, our $(3, 1)$-round scheme for black-box simulation is essentially round-optimal.

Notice that both our $(3, 1)$-round protocols – the one based on BB use of TCom and the other based on NBB use of OWFs – contradict the impossibility given in [73], that was claimed to hold regardless of the access to the cryptographic primitives.

All the constructions that we present are secure under concurrent-with-barrier composition, which obviously implies parallel composition. Our simulators work

[6]This result indeed requires a relaxed definition of trapdoor commitment where the trapdoor is required to be known already during the commitment phase in order to later equivocate. We call it "weak" because any TCom is also a wTCom.

for any message distributions, and do not need to know the distribution of the messages committed to by the honest sender. In light of our impossibility for the fully concurrent composition (see Item 2 of the above list), the concurrency achieved by our schemes seems to be optimal for this setting.

4 Simultaneously Resettable Arguments of Knowledge

Interaction and private randomness are the two fundamental ingredients in Cryptography. They are crucial for achieving zero-knowledge proofs [39]. In [15] Canetti, Goldreich, Goldwasser and Micali showed that when private randomness is limited and re-used in multiple instances of a proof system, it is still possible to preserve the zero-knowledge requirement. The setting proposed by [15] is of a malicious verifier that resets the prover, therefore forcing the prover to run several protocol executions using the same randomness. This setting applies to protocols where the prover is implemented by a stateless device. Therefore, a prover can only count on the limited (hardwired) randomness while it can be adaptively reset any polynomial number of times. The resulting security notion against such powerful verifiers is referred to as *resettable zero knowledge* (rZK) and is provably harder to achieve than concurrent zero knowledge. Feasibility results have been achieved in [15, 49] in the standard model with the following round-complexity: polylogarithmic for rZK and constant for resettable witness indistinguishability (rWI, in short). Since then, it was also shown how to achieve resettable zero knowledge in the Bare Public-Key (BPK) model, introduced by Canetti et al. [15], where one can obtain better round complexity and assumptions [54, 24, 1, 79]. Very recently, it has been shown [35] that resettable statistical zero knowledge for non-trivial languages is possible.

The "reverse" of the above question has been considered by Barak, Goldreich, Goldwasser and Lindell [6] where a malicious prover resets a verifier, called *resettable soundness*. In [6], it has been shown how to obtain resettable soundness along with ZK in a constant number of rounds.

Barak et al. [6] proposed the challenging *simultaneous resettability conjecture*, where one would like to prove that a protocol is secure against both a resetting malicious prover and a resetting malicious verifier. The existing machinery turned out to be insufficient, and a definitive answer required almost a decade. In the work of Deng, Goyal and Sahai [22] they showed a resettably sound rZK argument for **NP** with polynomial round complexity. Very recently, results in the BPK model for simultaneous resettability have been obtained in [78, 2] with a constant number of rounds.

Arguments of knowledge under simultaneous resettability. Argument systems are often used with a different goal than proving membership of an instance in a language. Indeed, it is commonly required to prove knowledge (possession) of a witness instead of the truthfulness of a statement. Since arguments of knowledge serve as major building blocks in Cryptography (e.g., in identification schemes[7]), it is an interesting question whether the previous results for arguments of membership extend to arguments of knowledge. Unfortunately, arguments of knowledge have been achieved so far only when one party can reset. That is, we have rZK arguments of knowledge [15] and, separately, resettably sound ZK arguments of knowledge [6]. Instead, when reset attacks are possible in both directions, no result is known even when only rWI with resettable argument of knowledge is desired.

In [31] Dwork and Naor present ZAPs, which are simultaneously resettable WI proofs. It is important to note that resettable security for ZAPs comes almost for free because of the minimal round complexity (1 or 2 rounds). However, it is not known how to accommodate for knowledge extraction, unless one relies on non-standard (e.g., non-falsifiable) assumptions. For the case of resettably sound rZK, all the above results [22, 78, 2] critically use an instance-dependent technique along with ZAPs: when the statement is true (i.e., when proving rZK), the prover/simulator can run ZAPs which allow the use of multiple witnesses. Such use of multiple witnesses gives some flexibility that turns out to be very useful to prove resettable zero knowledge. Instead, when the statement is false, the protocols are designed so that adversarial malicious prover must stick with some fixed messages during the execution of protocol. Therefore, rewinding capabilities do not help the resetting malicious prover since he can not change those fixed messages. This is critically used in the proofs of resettable soundness in order to reach a contradiction when a prover proves a false statement. It is easy to see that the above approach fails when arguments of knowledge are considered. Indeed, when the malicious resetting prover proves a true statement, the same freedom that allows one to prove rZK/rWI, also gives extra power to the malicious prover. Consequently, designing an extractor appears problematic and new techniques seem to be needed so that the simultaneous resettability conjecture is resolved even when we consider knowledge extraction.

Our main result is the first construction of a constant-round simultaneously resettable witness-indistinguishable argument of knowledge for any **NP** language. Our protocol is based on the novel use of ZAPs and resettably sound zero knowledge arguments, which improves over the techniques previously used in [22, 78] as well as concurrent and independent work[8] of [42].

[7] Bellare et al. in [8] gave various definitions for identification schemes when the adversary can also reset the proving device.

[8] In an independent work [42], Goyal and Maji achieved simultaneously resettable secure com-

As application of our main protocol, we also consider the question of secure identification under simultaneous resettability and show how to use the above simultaneous resettable WI argument of knowledge to obtain the first simultaneously resettable identification scheme which follows the knowledge extraction paradigm.

Acknowledgment

The results presented in this survey are contained in the phD thesis: "Secure computation under concurrent and physical attacks", supervised by Prof. Ivan Visconti.

References

[1] Joël Alwen, Giuseppe Persiano, and Ivan Visconti. Impossibility and feasibility results for zero knowledge with public keys. In *Advances in Cryptology – Crypto '05*, volume 3621 of *Lecture Notes in Computer Science*, pages 135–151. Springer Verlag, 2005.

[2] Seiko Arita. A constant-round resettably-sound resettable zero-knowledge argument in the bpk model. Cryptology ePrint Archive, Report 2011/404, 2011. `http://eprint.iacr.org/`.

[3] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, François-Xavier Standaert, and Christian Wachsmann. A formalization of the security features of physical functions. In *IEEE Symposium on Security and Privacy*, pages 397–412. IEEE Computer Society, 2011.

[4] Frederik Armknecht, Roel Maes, Ahmad-Reza Sadeghi, Berk Sunar, and Pim Tuyls. Memory leakage-resilient encryption based on physically unclonable functions. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 685–702. Springer, 2009.

[5] Boaz Barak, Ron Canetti, Jesper B. Nielsen, and Rafael Pass. Universally composable protocols with relaxed set-up assumptions. In *Foundations of Computer Science (FOCS'04)*, pages 394–403, 2004.

[6] Boaz Barak, Oded Goldreich, Shafi Goldwasser, and Yehuda Lindell. Resettably-sound zero-knowledge and its applications. In *FOCS*, pages 116–125, 2001.

[7] Mihir Bellare, Rafael Dowsley, Brent Waters, and Scott Yilek. Standard security does not imply security against selective-opening. In David Pointcheval and

putation. Their work achieves (with simulation-based security) simultaneous resettability with polynomial round complexity assuming also the existence of lossy trapdoor encryption.

Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 645–662. Springer, 2012.

[8] Mihir Bellare, Marc Fischlin, Shafi Goldwasser, and Silvio Micali. Identification protocols secure against reset attacks. In Birgit Pfitzmann, editor, *EUROCRYPT*, volume 2045 of *Lecture Notes in Computer Science*, pages 495–511. Springer, 2001.

[9] Mihir Bellare, Dennis Hofheinz, and Scott Yilek. Possibility and impossibility results for encryption and commitment secure under selective opening. In *EUROCRYPT*, pages 1–35, 2009.

[10] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 51–70. Springer, 2011.

[11] Christina Brzuska, Marc Fischlin, Heike Schröder, and Stefan Katzenbeisser. Physically uncloneable functions in the universal composition framework. *IACR Cryptology ePrint Archive*, 2011:681, 2011.

[12] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Foundations of Computer Science (FOCS'01)*, pages 136–145, 2001.

[13] Ran Canetti, Yevgeniy Dodis, Rafael Pass, and Shabsi Walfish. Universally composable security with global setup. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 61–85. Springer, 2007.

[14] Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 2001.

[15] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *STOC*, pages 235–244, 2000.

[16] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-Box Concurrent Zero-Knowledge Requires $\omega(\log n)$ Rounds. In *33st ACM Symposium on Theory of Computing (STOC '01)*, pages 570–579. ACM, 2001.

[17] Ran Canetti, Eyal Kushilevitz, and Yehuda Lindell. On the limitations of universally composable two-party computation without set-up assumptions. In Eli Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 68–86. Springer, 2003.

[18] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In John H. Reif, editor, *STOC*, pages 494–503. ACM, 2002.

[19] Nishanth Chandran, Vipul Goyal, and Amit Sahai. New constructions for uc secure computation using tamper-proof hardware. In Nigel P. Smart, editor, *EUROCRYPT*, volume 4965 of *Lecture Notes in Computer Science*, pages 545–562. Springer, 2008.

[20] Chongwon Cho, Rafail Ostrovsky, Alessandra Scafuro, and Ivan Visconti. Simultaneously resettable arguments of knowledge. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 530–547. Springer, 2012.

[21] Seung Geol Choi, Jonathan Katz, Dominique Schröder, Arkady Yerukhimovich, and Hong-Sheng Zhou. (efficient) universally composable two-party computation using a minimal number of stateless tokens. *IACR Cryptology ePrint Archive*, 2011:689, 2011.

[22] Yi Deng, Vipul Goyal, and Amit Sahai. Resolving the simultaneous resettability conjecture and a new non-black-box simulation strategy. In *FOCS*, pages 251–260. IEEE Computer Society, 2009.

[23] Giovanni Di Crescenzo. Minimal assumptions and round complexity for concurrent zero-knowledge in the bare public-key model. In *COCOON*, pages 127–137, 2009.

[24] Giovanni Di Crescenzo, Giuseppe Persiano, and Ivan Visconti. Constant-round resettable zero knowledge with concurrent soundness in the bare public-key model. In *Advances in Cryptology – Crypto '04*, volume 3152 of *Lecture Notes in Computer Science*, pages 237–253. Springer-Verlag, 2004.

[25] Giovanni Di Crescenzo, Giuseppe Persiano, and Ivan Visconti. Improved Setup Assumptions for 3-Round Resettable Zero Knowledge. In *Asiacrypt '04*, volume 3329 of *Lecture Notes in Computer Science*, pages 530–544. Springer-Verlag, 2004.

[26] Giovanni Di Crescenzo and Ivan Visconti. Concurrent zero knowledge in the public-key model. In *Proc. of ICALP '05*, volume 3580 of *Lecture Notes in Computer Science*, pages 22–33. Springer, 2005.

[27] Giovanni Di Crescenzo and Ivan Visconti. On defining proofs of knowledge in the bare public key model. In *ICTCS*, pages 187–198, 2007.

[28] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[29] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. Unconditional and composable security using a single stateful tamper-proof hardware token. In Ishai [**?**], pages 164–181.

[30] Nico Döttling, Daniel Kraschewski, and Jörn Müller-Quade. David & goliath oblivious affine function evaluation - asymptotically optimal building blocks for universally composable two-party computation from a single untrusted stateful tamper-proof hardware token. Cryptology ePrint Archive, Report 2012/135, 2012. `http://eprint.iacr.org/`.

[31] Cynthia Dwork and Moni Naor. Zaps and their applications. In *In 41st FOCS*, pages 283–293. IEEE, 2000.

[32] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions. In *Foundations of Computer Science (FOCS'99)*, pages 523–534, 1999.

[33] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *Proceedings of the 20th annual ACM symposium on Theory of computing*, STOC '98, pages 409–418. ACM, 1998.

[34] Ilze Eichhorn, Patrick Koeberl, and Vincent van der Leest. Logically reconfigurable pufs: memory-based secure key storage. In *Proceedings of the sixth ACM workshop on Scalable trusted computing*, STC '11, pages 59–64, New York, NY, USA, 2011. ACM.

[35] Sanjam Garg, Rafail Ostrovsky, Ivan Visconti, and Akshay Wadia. Resettable statistical zero knowledge. In *TCC*, Lecture Notes in Computer Science. Springer-Verlag, 2012.

[36] Rosario Gennaro and Silvio Micali. Independent zero-knowledge sets. In *ICALP*, volume 4052 of *Lecture Notes in Computer Science*, pages 181–234. Springer, 2006.

[37] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996.

[38] Shafi Goldwasser, Yael T. Kalai, and Guy. N. Rothblum. One-time programs. In *Advances in Cryptology – CRYPTO'08*, volume 5157 of *Lecture Notes in Computer Science*, pages 39–56. Springer, Berlin, Germany, 2008.

[39] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In Robert Sedgewick, editor, *STOC*, pages 291–304. ACM, 1985.

[40] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 308–326. Springer, 2010.

[41] Vipul Goyal, Chen-Kuei Lee, Rafail Ostrovsky, and Ivan Visconti. Constructing non-malleable commitments: A black-box approach. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012*, pages 51–60. IEEE Computer Society, 2012.

[42] Vipul Goyal and Hemanta K. Maji. Stateless cryptographic protocols. In *FOCS*, 2011.

[43] Jorge Guajardo, Sandeep S. Kumar, Geert Jan Schrijen, and Pim Tuyls. Fpga intrinsic pufs and their use for ip protection. In Pascal Paillier and Ingrid Verbauwhede, editors, *CHES*, volume 4727 of *Lecture Notes in Computer Science*, pages 63–80. Springer, 2007.

[44] Dennis Hofheinz. Possibility and impossibility results for selective decommitments. *J. Cryptology*, 24(3):470–516, 2011.

[45] Yael Tauman Kalai, Yehuda Lindell, and Manoj Prabhakaran. Concurrent general composition of secure protocols in the timing model. In *37th Annual ACM Symposium on Theory of Computing*, pages 644–653, 2005.

[46] Jonathan Katz. Universally composable multi-party computation using tamper-proof hardware. In Moni Naor, editor, *Advances in Cryptology – EUROCRYPT 2007*, volume 4515 of *Lecture Notes in Computer Science*, pages 115–128, Barcelona, Spain, May 20–24, 2007.

[47] Stefan Katzenbeisser, Ünal Koçabas, Vladimir Rozic, Ahmad-Reza Sadeghi, Ingrid Verbauwhede, and Christian Wachsmann. Pufs: Myth, fact or busted? a security evaluation of physically unclonable functions (pufs) cast in silicon. In Prouff and Schaumont [65], pages 283–301.

[48] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *Proceedings of the twenty-fourth annual ACM symposium on Theory of computing*, STOC '92, pages 723–732, New York, NY, USA, 1992. ACM.

[49] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In *Proceedings of the 33rd annual ACM symposium on Theory of computing*, STOC '01, pages 560–569. ACM, 2001.

[50] Ünal Koçabas, Ahmad-Reza Sadeghi, Christian Wachsmann, and Steffen Schulz. Poster: practical embedded remote attestation using physically unclonable functions. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 797–800. ACM, 2011.

[51] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. Pufky: A fully functional puf-based cryptographic key generator. In Prouff and Schaumont [65], pages 302–319.

[52] Roel Maes and Ingrid Verbauwhede. Physically unclonable functions: A study on the state of the art and future research directions. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 3–37. Springer Berlin Heidelberg, 2010.

[53] Silvio Micali and Leonid Reyzin. Min-round resettable zero-knowledge in the public-key model. In *Advances in Cryptology – Eurocrypt '01*, volume 2045 of *Lecture Notes in Computer Science*, pages 373–393. Springer-Verlag, 2001.

[54] Silvio Micali and Leonid Reyzin. Soundness in the public-key model. In *Advances in Cryptology – Crypto '01*, volume 2139 of *Lecture Notes in Computer Science*, pages 542–565. Springer-Verlag, 2001.

[55] Daniele Micciancio and Erez Petrank. Simulatable commitments and efficient concurrent zero-knowledge. In *Advances in Cryptology – Eurocrypt '03*, volume 2045 of *Lecture Notes in Computer Science*, pages 140–159. Springer-Verlag, 2003.

[56] Tal Moran and Gil Segev. David and Goliath commitments: UC computation for asymmetric parties using tamper-proof hardware. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 527–544, Istanbul, Turkey, April 13–17, 2008. Springer, Berlin, Germany.

[57] Rafail Ostrovsky, Giuseppe Persiano, and Ivan Visconti. Constant-round concurrent non-malleable zero knowledge in the bare public-key model. In *Proc. of ICALP '08*, volume 5126 of *Lecture Notes in Computer Science*, pages 548–559. Springer, 2008.

[58] Rafail Ostrovsky, Vanishree Rao, Alessandra Scafuro, and Ivan Visconti. Revisiting lower and upper bounds for selective decommitments. In submission to TCC 2013.

[59] Ostrovsky, R., Scafuro, A., Visconti, I., Wadia, A.: Universally composable secure computation with (malicious) physically uncloneable functions. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT. Lecture Notes in Computer Science, vol. 7881, pp. 702–718. Springer (2013)

[60] Ravikanth S. Pappu, Ben Recht, Jason Taylor, and Niel Gershenfeld. Physical one-way functions. *Science*, 297:2026–2030, 2002.

[61] Ravikanth Srinivasa Pappu. *Physical One-Way Functions*. PhD thesis, MIT, 2001.

[62] Rafael Pass and Hoeteck Wee. Black-box constructions of two-party protocols from one-way functions. In *TCC*, pages 403–418, 2009.

[63] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *In 43rd FOCS*, pages 366–375, 2002.

[64] Manoj Prabhakaran and Amit Sahai. New notions of security: achieving universal composability without trusted setup. In *36th Annual ACM Symposium on Theory of Computing*, pages 242–251, 2004.

[65] Emmanuel Prouff and Patrick Schaumont, editors. *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*. Springer, 2012.

[66] Ransom Richardson and Joe Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In *Advances in Cryptology – Eurocrypt '99*, volume 1592 of *Lecture Notes in Computer Science*, pages 415–431. Springer-Verlag, 1999.

[67] Ulrich Rührmair. Oblivious transfer based on physical unclonable functions. In Alessandro Acquisti, Sean W. Smith, and Ahmad-Reza Sadeghi, editors, *TRUST*, volume 6101 of *Lecture Notes in Computer Science*, pages 430–440. Springer, 2010.

[68] Ulrich Rührmair, Stefan Katzenbeisser, and H. Busch. Strong pufs: Models, constructions and security proofs. In A. Sadeghi and P. Tuyls, editors, *Towards Hardware Intrinsic Security: Foundations and Practice*, pages 79–96. Springer, 2010.

[69] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. Enhancing rfid security and privacy by physically unclonable functions. In Ahmad-Reza Sadeghi and David Naccache, editors, *Towards Hardware-Intrinsic Security*, Information Security and Cryptography, pages 281–305. Springer Berlin Heidelberg, 2010.

[70] Alessandra Scafuro and Ivan Visconti. On round-optimal zero knowledge in the bare public-key model. In *EUROCRYPT*, Lecture Notes in Computer Science. Springer-Verlag, 2012.

[71] Pim Tuyls and Lejla Batina. Rfid-tags for anti-counterfeiting. In David Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 115–131. Springer, 2006.

[72] Ivan Visconti. Efficient zero knowledge on the internet. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP (2)*, volume 4052 of *Lecture Notes in Computer Science*, pages 22–33. Springer, 2006.

[73] David Xiao. (Nearly) round-optimal black-box constructions of commitments secure against selective opening attacks. In Ishai [**?**], pages 541–558.

[74] David Xiao. On the round complexity of black-box constructions of commitments secure against selective opening attacks. Cryptology ePrint Archive, Report 2009/513 - Revision May 29, 2012, 2012. `http://eprint.iacr.org/`.

[75] David Xiao. Round-optimal black-box statistically binding selective-opening secure commitments. In *Progress in Cryptology - AFRICACRYPT 2012 - 5th International Conference on Cryptology in Africa, Ifrance, Morocco, July 10-12, 2012. Proceedings*, volume 7374 of *Lecture Notes in Computer Science*, pages 395–411. Springer, 2012.

[76] Andrew Chi-Chih Yao, Moti Yung, and Yunlei Zhao. Concurrent knowledge-extraction in the public-key model. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(002), 2007.

[77] Andrew Chi-Chih Yao, Moti Yung, and Yunlei Zhao. Concurrent knowledge extraction in the public-key model. In *ICALP (1)*, pages 702–714, 2010.

[78] Deng Yi, Dengguo Feng, Vipul Goyal, Dongdai Lin, Amit Sahai, and Moti Yung. Resettable cryptography in constant rounds - the case of zero knowledge. In *ASIACRYPT*, 2011.

[79] Moti Yung and Yunlei Zhao. Generic and practical resettable zero-knowledge in the bare public-key model. In *EUROCRYPT*, pages 129–147, 2007.

[80] Yunlei Zhao. Concurrent/resettable zero-knowledge with concurrent soundness in the bare public-key model and its applications. Cryptology ePrint Archive, Report 2003/265, 2003. `http://eprint.iacr.org/`.

[81] Yunlei Zhao, Xiaotie Deng, Chan H. Lee, and Hong Zhu. Resettable zero-knowledge in the weak public-key model. In *Advances in Cryptology – Eurocrypt '03*, volume 2045 of *Lecture Notes in Computer Science*, pages 123–139. Springer-Verlag, 2003.

[82] Damgård, I., Scafuro, A.: Unconditionally secure and universally composable commitments from physical assumptions. To appear in Asiacrypt 2013.

Expressive power of Constraint Handling Rules extensions and fragments

Jacopo Mauro
University of Bologna / INRIA
jmauro@cs.unibo.it

Abstract

Constraints can be used in concurrency theory to increase the conciseness and the expressive power of concurrent languages from a pragmatic point of view. In this work we study the expressive power of a concurrent language, namely Constraint Handling Rules, that supports constraints as a primitive construct. We show what features of this language make it Turing powerful and what happens to its expressive power if priorities are introduced.

1 Introduction

Constraint is a ubiquitous concept: in every day life there are a lot of rules (physical, chemical, economical, and legal) that restrict, limit or regulate the way we operate and what decisions we take. In computer science constraints can be very useful not only to model the world but also to discover or verify if instances satisfy a model. For these reasons the notion of constraints gave birth to a new field called Constraint Programming that has attracted wide attention since it provides a concise and elegant way to describe problems and also efficient tools to compute solutions. Concurrency is a universal concept too. In every second of our life there are thousands of events or tasks occurring simultaneously and interacting with each other. With the evolution of the networks a lot of connected computers are available and nowadays more and more people think that we are inevitably going towards a world full of interconnected devices. This network of devices is a concurrent system and has peculiarities and characteristics that an environment constituted by only one processing unit does not have. On one hand a concurrent system is usually hard to use since, in such a system, problems like deadlocks, resources conflicts, security emerge. On the other hand a concurrent system can be the only mean to solve problems requiring huge amounts of resources or modeling complex scenarios in a simple and clear way.

The idea of approaching hard combinatorial optimization problems through a combination of search and constraint solving appeared first in logic programming. Despite the continued support of logic programming for constraint programmers, research efforts were initiated to import constraint technologies into other paradigms. Nowadays constraints can for instance be easily used in imperative languages. However constraints are equally well suited for modeling and supporting concurrency. In particular, concurrent computation can be seen for instance as agents that communicate and coordinate through a shared constraint store [21]. Importing constraint in existing languages raises some concerns: how easy is to use the new language, how expressive, and how extensible? Each concern is intrinsically linked to the host language and has a direct impact on potential end-users. It is desirable to obtain a declarative reading of a high-level model statement that exploits the facilities of the host language. Extensibility is also crucial since it is important to support the addition of user-defined constraints and user-defined search procedures.

Among all the concurrent languages having constraints as primitive building blocks [9, 23, 26–28, 30, 33] we focus our attention to the language called Constraint Handling Rule (CHR) [15, 16], a committed-choice declarative language. In the last few years, several papers have been devoted to investigate the expressivity of CHR that, as a language, is Turing powerful. When looking for decidable properties it is natural to consider restrictions of CHR which allow for instance the use of only the equality constraint = (interpreted in the usual way as equality on the Herbrand universe) and which, similarly to Datalog, is defined over a signature which contains no function symbols of arity > 0. In this work we first present, as an example, a decidability result obtained for a fragment of CHR that does not allow the introduction of new variables. Then we give an overview of the state of the art concerning the study of the decidability result of some fragments of CHR.

In the second part of the work we consider an extension of CHR dubbed CHR^{ω_p} [22] that was proposed for supporting an high-level, explicit form of execution control which is more flexible and declarative than the one offered by the usual ω_r semantics [12] of CHR. This extension is obtained by introducing explicitly in the syntax of the language rule annotations which allow to specify the priority of each rule to execute. Priorities can be either static, when the annotations are completely defined at compile time, or dynamic, when the annotations contain variables which are instantiated at run-time. In this work we show several expressivity results relating CHR and CHR^{ω_p} by using the notion of acceptable encoding, i.e. a notion of expressivity coming from the field of concurrency theory. In fact, in this field the issue of the expressive power of a language has received a considerable attention in the last years and several techniques and formalisms have been proposed for separating the expressive power of different languages which are Turing powerful (and therefore can not be properly compared by using

the standard tools of computability theory). Such a separation is meaningful both from a theoretical and a pragmatic point of view, since different (Turing complete) languages can provide quite different tools for implementing our algorithms. We show in particular an example of acceptable encoding that proves that dynamic priorities does not improve the expressive power of static priorities while instead priorities do improve the expressivity of CHR.

Structure of the paper. In Section 2 we introduce the CHR language and the notion of acceptable encoding used to discriminate between two Turing powerful languages. In Section 3 we study the decidability fragments of CHR while in Section 4 we investigate the expressive power of CHR with priorities. We conclude in Section 5.

Note. This work is an extract of the results presented in the PhD thesis of the author. For more details we defer the readers attention to [24]. I would like to thank my coauthors and my PhD advisor, Prof. Maurizio Gabbrielli, for their help and guidance during these last years.

2 Constraint Handling Rules

Constraint Handling Rule (CHR) [4, 14–16] is a committed-choice declarative language which has been originally designed for writing constraint solvers and which is nowadays a general purpose language. In this section, after introducing the used notation, we give an overview of the CHR syntax and semantics. We then define what is an acceptable encoding between two CHR languages.

2.1 Notation

Following [12, 16], we distinguish the constraints handled by an existing solver, called built-in (or predefined) constraints, from those defined by the CHR program, called user-defined (or CHR) constraints. Therefore we assume a signature Σ on which program terms are defined and two disjoint sets of predicate symbols Π_b for built-in and Π_u for user-defined constraints.

Definition 1 (Built-in constraint)**.** *A* built-in constraint $p(t_1, \ldots, t_n)$ *is an atomic predicate where* p *is a predicate symbol from* Π_b *and* $t_1, \ldots, t_n$ *are terms over the signature* Σ*. For built-in constraints we assume a (first order) theory* $\mathcal{CT}$ *which describes their meaning.*

Definition 2 (User-defined constraint)**.** *A* user-defined (or CHR) constraint $p(t_1, \ldots, t_n)$ *is an atomic predicate where* p *is a predicate symbol from* Π_u *and* $t_1, \ldots, t_n$ *are terms over the signature* Σ*.*

We use c, d to denote built-in constraints, h, k to denote CHR constraints and a, b, f, g to denote both built-in and user-defined constraints (we will call these generally constraints). The capital versions of these notations will be used to denote multisets of constraints. We also denote by $false$ any inconsistent conjunction of constraints and with *true* the empty multiset of built-in constraints.

We will use "," rather than $\wedge$ to denote conjunction and we will often consider a conjunction of atomic constraints as a multiset of atomic constraints. We prefer to use multisets rather than sequences (as in the original CHR papers) because our results do not depend on the order of atoms in the rules. In particular, we will use this notation based on multisets in the syntax of CHR.

The notation $\exists_V \phi$, where V is a set of variables, denotes the existential closure of a formula ϕ w.r.t. the variables in V, while the notation $\exists_{-V} \phi$ denotes the existential closure of a formula ϕ with the exception of the variables in V which remain unquantified. $Fv(\phi)$ denotes the free variables appearing in ϕ. Finally, we denote by $\bar{t}$ and $\bar{X}$ a sequence of terms and of distinct variables, respectively.

In the following, if $\bar{t} = t_1, \ldots t_m$ and $\bar{t'} = t'_1, \ldots t'_m$ are sequences of terms then the notation $p(\bar{t}) = p'(\bar{t'})$ represents the set of equalities $t_1 = t'_1, \ldots, t_m = t'_m$ if $p = p'$, and it is undefined otherwise. This notation is extended in the expected way to multiset of constraints. Moreover we use $+\!\!+$ to denote sequence concatenation and $\uplus$ for multi-set union.

We follow the logic programming tradition and indicate the application of a substitution σ to a syntactic object t by σt.

To distinguish between different occurrences of syntactically equal constraints a CHR constraints can be labeled by a unique identifier. The resulting syntactic object is called identified CHR constraint and is denoted by $k\#i$, where k is a CHR constraint and i is the identifier. We also use the functions defined as $\texttt{chr}(k\#i) = k$ and $\texttt{id}(k\#i) = i$, possibly extended to sets and sequences of identified CHR constraints in the obvious way.

2.2 CHR program

A CHR program is defined as a sequence of three kinds of rules: simplification, propagation and simpagation rules. Intuitively, simplification rewrites constraints into simpler ones, propagation adds new constraints which are logically redundant but may trigger further simplifications, simpagation combines in one rule the effects of both propagation and simplification rules. For simplicity we consider simplification and propagation rules as special cases of a simpagation rule. The general form of a simpagation rule is:

$$r \; @ \; H^k \backslash H^h \iff D \mid B$$

reflexivity	$\texttt{leq}(X,Y) \Longleftrightarrow X = Y \mid true$
antisymmetry	$\texttt{leq}(X,Y), \texttt{leq}(Y,X) \Longleftrightarrow X = Y$
transitivity	$\texttt{leq}(X,Y), \texttt{leq}(Y,Z) \Rightarrow \texttt{leq}(X,Z)$

Figure 1: A program for defining ≤ in CHR

where r is a unique identifier of a rule, H^k and H^h (the heads) are multi-sets of CHR constraints, D (the guard) is a possibly empty multi-set of built-in constraints and B is a possibly empty multi-set of (built-in and user-defined) constraints. If H^k is empty then the rule is a simplification rule. If H^h is empty then the rule is a propagation rule. At least one of H^k and H^h must be non empty.

In the following when the guard D is empty or *true* we omit $D\,|$. Also the names of rules are omitted when not needed. For a simplification rule we omit $H^k \backslash$ while we write a propagation rule as $H^k \Rightarrow D \mid B$. A CHR *goal* is a multi-set of (both user-defined and built-in) constraints. An example of a CHR program is shown in Figure 1. This program implements the less or equal predicate, assuming that we have only the equality predicate in the available built-in constraints. The first rule, a simplification, deletes the constraint $leq(X,Y)$ if $X = Y$. Analogously the second rule deletes the constraints $leq(X,Y)$ and $leq(Y,X)$ adding the built-in constraint $X = Y$. The third rule of the program is a propagation rule and it is used to add a constraint $leq(X,Z)$ when the two constraints $leq(X,Y)$ and $leq(Y,Z)$ are found.

2.3 Traditional operational semantics

The theoretical operational semantics of CHR, denoted by ω_t, is given in [12] as a state transition system $T = (Conf, \overset{\omega_t}{\rightarrow}_P)$: configurations in $Conf$ are tuples of the form $\langle G, S, B, T\rangle_n$, where G is the goal (a multi-set of constraints that remain to be solved), S is the CHR store (a set of identified CHR constraints), B is the built-in store (a conjunction of built-in constraints), T is the propagation history (a set of sequence of identifiers used to store the rule instances that have fired) and n is the next free identifier (it is used to identify new CHR constraints). The propagation history is used to avoid trivial non termination that could be introduced by repeated application of the same propagation rule. The transitions of ω_t are shown in Table 1.

Given a program P, the transition relation $\overset{\omega_t}{\rightarrow}_P \subseteq Conf \times Conf$ is the least relation satisfying the rules in Table 1. The **Solve** transition allows to update the constraint store by taking into account a built-in constraint contained in the goal. The **Introduce** transition is used to move a user-defined constraint from the goal

Solve	$\langle (c, G), S, C, T\rangle_n \xrightarrow{\omega_t}_P \langle G, S, c \wedge C, T\rangle_n$ where c is a built-in constraint
Introduce	$\langle (k, G), S, C, T\rangle_n \xrightarrow{\omega_t}_P \langle G, \{c\#n\} \cup S, C, T\rangle_{n+1}$ where k is a CHR constraint
Apply	$\langle G, H_1 \cup H_2 \cup S, C, T\rangle_n \xrightarrow{\omega_t}_P \langle (B, G), H_1 \cup S, \theta \wedge D \wedge C, T \cup \{t\}\rangle_n$ where P contains a (renamed apart) rule $$r \ @ H'_1 \backslash H'_2 \iff D \mid B$$ and there exists a matching substitution θ s.t. $\mathtt{chr}(H_1) = \theta H'_1$, $\mathtt{chr}(H_2) = \theta H'_2$, $\mathcal{CT} \models C \rightarrow \exists_{-Fv(C)}(\theta \wedge D)$ and $t = \mathtt{id}(H_1) +\!\!+ \mathtt{id}(H_2) +\!\!+ [r] \notin T$

Table 1: Transitions of ω_t

to the CHR constraint store, where it can be handled by applying CHR rules. The **Apply** transition allows to rewrite user-defined constraints (which are in the CHR constraint store) by using rules from the program. As usual, in order to avoid variable name clashes, this transition assumes that all variables appearing in a program clause are fresh ones. The **Apply** transition is applicable when the current store (B) is strong enough to entail the guard of the rule (D), once the parameter passing has been performed. Note also that, as previously mentioned, the condition $\mathtt{id}(H_1) +\!\!+ \mathtt{id}(H_2) +\!\!+ [r] \notin T$ avoids repeated application of the same propagation rule and therefore trivial non-termination.

An *initial configuration* has the form $\langle G, \emptyset, true, \emptyset\rangle_1$ while a *final configuration* has either the form $\langle G, S, false, T\rangle_k$, when it is *failed*, or the form $\langle \emptyset, S, B, T\rangle_k$, when it is successfully terminated because there are no applicable rules.

Given a goal G, the operational semantics that we consider observes the non failed final stores of terminating computations. This notion of observable is the most used in the CHR literature and is captured by the following.

Definition 3. *[Qualified answers [16]] Let P be a program and let G be a goal. The set $\mathcal{QA}_P(G)$ of qualified answers for the query G in the program P is defined as:*

$$\mathcal{QA}_P(G) = \{\exists_{-Fv(G)}(K \wedge d) \mid \ \mathcal{CT} \not\models d \leftrightarrow false, \ \langle G, \emptyset, true, \emptyset\rangle_1 \xrightarrow{\omega_t}{}^*_P \langle \emptyset, K, d, T\rangle_n \not\xrightarrow{\omega_t}_P\}$$

We also consider the following different notion of answer, obtained by computations terminating with a user-defined constraint which is empty. We call these

observables *data sufficient answers* slightly deviating from the terminology of [16] (a goal which has a data sufficient answer is called a data-sufficient goal in [16]).

Definition 4. *[Data sufficient answers] Let P be a program and let G be a goal. The set $\mathcal{SA}_P(G)$ of data sufficient answers for the query G in the program P is defined as:*

$$\mathcal{SA}_P(G) = \{\exists_{-Fv(G)}(d) \mid \quad \mathcal{CT} \not\models d \leftrightarrow false, \\ \langle G, \emptyset, true, \emptyset\rangle_1 \overset{\omega_t}{\rightarrow}_P^* \langle \emptyset, \emptyset, d, T\rangle_n\}$$

Both previous notions of observables characterize an input/output behaviour, since the input constraint is implicitly considered in the goal. Clearly in general $\mathcal{SA}_P(G) \subseteq \mathcal{QA}_P(G)$ holds, since data sufficient answers can be obtained by setting $K = \emptyset$ in Definition 3.

2.4 Abstract operational semantics

The first CHR operational semantics defined in [16] differs from the traditional semantics ω_t. Indeed this original, so called, abstract semantics denoted by ω_a, allows the firing of a propagation rule an infinite number of times. For this reason ω_a can be seen as the abstraction of the traditional semantics where the propagation history is not considered. In Table 2 we have reported the transaction of the ω_a semantics following the structure of the theoretical semantics using configurations without a propagation history set.

Given a program P, the transition relation $\overset{\omega_a}{\rightarrow}_P \subseteq Conf \times Conf$ is the least relation satisfying the rules in Table 2.

Initial and final configurations can be defined analogously to those of ω_t semantics. In the same way we can define the observables: qualified and data sufficient answers.

2.5 CHR with priorities

De Koninck et al. [22] extended CHR with user-defined priorities. This new language, denoted by CHR^{ω_p}, provides an high level alternative for controlling program execution, that is more appropriate to needs of CHR programmers than other low level approaches.

The syntax of CHR with priorities is compatible with the syntax of CHR. A simpagation rule has now the form

$$p :: r\ @\ H^k \backslash H^h \iff D \mid B$$

Solve	$\langle (c, G), S, C, T\rangle_n \xrightarrow{\omega_a}_P \langle G, S, c \wedge C, T\rangle_n$ where c is a built-in constraint
Introduce	$\langle (k, G), S, C, T\rangle_n \xrightarrow{\omega_a}_P \langle G, \{c\#n\} \cup S, C, T\rangle_{n+1}$ where k is a CHR constraint
Apply	$\langle G, H_1 \cup H_2 \cup S, C, T\rangle_n \xrightarrow{\omega_t}_P \langle (B, G), H_1 \cup S, \theta \wedge D \wedge C, T \cup \{t\}\rangle_n$ where P contains a (renamed apart) rule $$r \ @ H'_1 \backslash H'_2 \iff D \mid B$$ and there exists a matching substitution θ s.t. $\mathtt{chr}(H_1) = \theta H'_1$, $\mathtt{chr}(H_2) = \theta H'_2$, $\mathcal{CT} \models C \rightarrow \exists_{-Fv(C)}(\theta \wedge D)$

Table 2: Transitions of ω_a

$$1 :: \mathtt{source}(V) \Longrightarrow \mathtt{dist}(V, 0)$$
$$1 :: \mathtt{dist}(V, D_1) \backslash \mathtt{dist}(V, D_2) \Longleftrightarrow D_1 \leq D_2 | true$$
$$D + 2 :: \mathtt{dist}(V, D), \mathtt{edge}(V, C, U) \Longrightarrow \mathtt{dist}(U, D + C)$$

Figure 2: A program for computing the shortest path in CHR^{ω_p}

where r, H^k, H^h, D, B are defined as in the CHR simpagation rule in Section 2.2, while p is an arithmetic expression, with $Fv(p) \subseteq (Fv(H^k) \cup Fv(H^h))$, which expresses the priority of rule r. If $Fv(p) = \emptyset$ then p is a static priority, otherwise it is called dynamic.

The formal semantics of CHR^{ω_p}, defined by [22], is an adaptation of the traditional semantics to deal with rule priorities. Formally this semantics, denoted by ω_p, is a state transition system $T = (Conf, \xrightarrow{\omega_p}_P)$ where P is a CHR^{ω_p} program while configurations in $Conf$, as well as the initial and final configurations, are the same as those introduced for the traditional semantics in Section 2.3. The transition relation $\xrightarrow{\omega_p}_P \subseteq Conf \times Conf$ is the least relation satisfying the rules in Table 3. The **Solve** and **Introduce** transitions are equal to those defined for the traditional semantics. The **Apply** transition instead is modified in order to take into account priorities. In fact, a further condition is added imposing that a rule can be fired only if no other rule that can be applied has a smaller value for the priority annotation (as usual in many systems, smaller values correspond to higher priority; For simplicity in the following we will use the terminology "higher" or "lower" priority rather than considering the values).

Solve	$\langle (c, G), S, C, T\rangle_n \overset{\omega_p}{\rightarrow}_P \langle G, S, c \wedge C, T\rangle_n$ where c is a built-in constraint
Introduce	$\langle (k, G), S, C, T\rangle_n \overset{\omega_p}{\rightarrow}_P \langle G, \{c\#n\} \cup S, C, T\rangle_{n+1}$ where k is a CHR constraint
Apply	$\langle \emptyset, H_1 \cup H_2 \cup S, C, T\rangle_n \overset{\omega_p}{\rightarrow}_P \langle B, H_1 \cup S, \theta \wedge D \wedge C, T \cup \{t\}\rangle_n$ where P contains a (renamed apart) rule $$p :: r \ @ H_1' \backslash H_2' \iff D \mid B$$ and there exists a matching substitution θ s.t. $\mathtt{chr}(H_1) = \theta H_1'$, $\mathtt{chr}(H_2) = \theta H_2'$, $\mathcal{CT} \models C \rightarrow \exists_{-Fv(C)}(\theta \wedge D)$, θp is a ground arithmetic expression and $t = \mathtt{id}(H_1) +\!\!+ \mathtt{id}(H_2) +\!\!+ [r] \notin T$. Furthermore no rule of priority p' and substitution θ' exists with $\theta' p' < \theta p$ for which the above conditions hold

Table 3: Transitions of ω_p

An example of a CHR^{ω_p} program (from [22]) is shown in Figure 2. This program can be used to compute the length of the shortest path between a source node and all the other nodes in the graph. We assume that the source node n is defined by using the constraint $source(n)$ and that the graph is represented by using the constraints $edge(V, C, U)$ for every edge of length C between two nodes V, U. When the program terminates we obtain a constraint $dist(U, C)$ iff the length of the shortest path between the source node and U is C.

The qualified and data sufficient answers for CHR^{ω_p} can be defined analogously to those of the standard language:

Definition 5. *[Qualified answers] Let P be a CHR^{ω_p} program and let G be a goal. The set $\mathcal{QA}_P(G)$ of qualified answers for the query G in the program P is defined as:*

$$\mathcal{QA}_P(G) = \{\exists_{-Fv(G)}(K \wedge d) \mid \ \mathcal{CT} \not\models d \leftrightarrow false, \ \langle G, \emptyset, true, \emptyset\rangle_1 \overset{\omega_p}{\rightarrow}_P{}^{*} \langle \emptyset, K, d, T\rangle_n \overset{\omega_p}{\not\rightarrow}_P\}$$

Definition 6. *[Data sufficient answers] Let P be a CHR^{ω_p} program and let G be a goal. The set $\mathcal{SA}_P(G)$ of data sufficient answers for the query G in the program*

P is defined as:

$$\mathcal{SA}_P(G) = \{\exists_{-Fv(G)}(d) \mid \ \mathcal{CT} \not\models d \leftrightarrow false, \\ \langle G, \emptyset, true, \emptyset\rangle_1 \overset{\omega_p}{\rightarrow}{}^*_P \ \langle \emptyset, \emptyset, d, T\rangle_n\}$$

2.6 Language encoding

In order to compare the expressive power of two languages we use the notion of language encoding, first formalized in [10, 29, 34].[1] Intuitively, a language $\mathcal{L}$ is more expressive than a language $\mathcal{L}'$ or, equivalently, $\mathcal{L}'$ can be encoded in $\mathcal{L}$, if each program written in $\mathcal{L}'$ can be translated into an $\mathcal{L}$ program in such a way that: (1) the intended observable behavior of the original program is preserved, under some suitable decoding; (2) the translation process satisfies some additional restrictions.

In this work we impose two requirements on the translation. First we require that the translation of the goal (in the original program) and the decoding of the results (in the translated program) are homomorphic w.r.t. the conjunction of atoms. This assumption essentially means that our encoding and decoding functions respect the structure of the original goal and of the results (recall that for CHR programs these are constraints, that is, conjunction of atoms). Next we assume that the results to be preserved are the, so called, qualified answers. Also this is a rather natural assumption, since these are the typical CHR observables for many CHR reference semantics. To simplify the treatment we assume that both the source and the target language use the same built-in constraints, semantically described by a theory $\mathcal{CT}$, which is not changed in the translation process.

We formally define a *program encoding* as any function $\mathcal{PROG} : \mathcal{P}_{\mathcal{L}} \rightarrow \mathcal{P}_{\mathcal{L}'}$ which translates a $\mathcal{L}$ program into a (finite) $\mathcal{L}'$ program ($\mathcal{P}_{\mathcal{L}}$ and $\mathcal{P}_{\mathcal{L}'}$ denote the set of $\mathcal{L}$ and $\mathcal{L}'$ programs, respectively). To simplify the treatment we assume that both the source and the target language use the same built-in constraints semantically described by a theory $\mathcal{CT}$.

In order to define when an encoding is acceptable, we have to define how the initial goal and the observables should be translated by the encoding and the decoding functions, respectively. We require that these translations are compositional w.r.t. the conjunction of atoms. This assumption essentially means that the encoding and the decoding respect the structure of the original goal and of the observables. Moreover, since the source and the translated programs use the same constraint theory, it is natural to assume also that these two functions do not modify or add built-in constraints (in other words, we do not allow to simulate the behaviour and the effects of the constraint theory).

[1] The original terminology of these papers was "language embedding".

We do not impose any restriction on the program translation, hence we have the following definition.

Definition 7 (Acceptable encoding)**.** *Suppose that C is the class of all the possible multisets of constraints. An* acceptable encoding *(of $\mathcal{L}$ into $\mathcal{L}'$) is a tern of mappings $(\mathcal{PROG}, \mathcal{INP}, \mathcal{OUT})$ where $\mathcal{PROG} : \mathcal{P}_{\mathcal{L}} \to \mathcal{P}_{\mathcal{L}'}$ is the program encoding, $\mathcal{INP} : C \to C$ is the goal encoding, and $\mathcal{OUT} : C \to C$ is the output decoding which satisfy the following conditions:*

1. *the goal encoding function is compositional, that is, for any goal $(A, B) \in C$, $\mathcal{INP}(A, B) = \mathcal{INP}(A), \mathcal{INP}(B)$ holds. We also assume that the built-ins present in the goal are left unchanged and no new built-ins can be added;*

2. *the output decoding function is compositional, that is, for any qualified answer $(A, B) \in C$, $\mathcal{OUT}(A, B) = \mathcal{OUT}(A), \mathcal{OUT}(B)$ holds. We also assume that the built-ins present in the answer are left unchanged and no new built-ins can be added;*

3. *Qualified answers are preserved for the class C, that is, for all $P \in \mathcal{P}_{\mathcal{L}}$ and $G \in C$, $\mathcal{QA}_P(G) = \mathcal{OUT}(\mathcal{QA}_{\mathcal{PROG}(P)}(\mathcal{INP}(G)))$ holds.*

Moreover we define an acceptable encoding for data sufficient answers *of $\mathcal{L}$ into $\mathcal{L}'$ exactly as an acceptable encoding, with the exception that the third condition above is replaced by the following:*

3'. *Data sufficient answers are preserved for the class C, that is, for all $P \in \mathcal{P}_{\mathcal{L}}$ and $G \in C$, $\mathcal{SA}_P(G)$ is equal to the data sufficient answers in $\mathcal{OUT}(\mathcal{QA}_{\mathcal{PROG}(P)}(\mathcal{INP}(G)))$.*[2]

Further weakening these conditions and requiring for instance that the translation of A, B is some form of composition of the translation of A and B does not seem reasonable, as conjunction is the only form for goal composition available in CHR.

3 Non Turing powerful fragments of CHR

The computational power of CHR depends on several aspects, including the number of atoms allowed in the heads, the underlying signature Σ on which programs are defined, and the constraint theory $\mathcal{CT}$, defining the built-ins.

[2] Note that in 3. and in 3′. the function $\mathcal{OUT}()$ is extended in the obvious way to sets of qualified answers.

In this section, as an example, we give the proof that the fragment of CHR that does not introduce new variables is non Turing powerful. Then we provide an overview of the decidability results of CHR fragments.

The language under consideration in this section is the CHR defined over a signature which contains no function symbol of arity > 0 and interpreted using the ω_a semantics. We denote this language as $CHR^{\omega_a}(C)$. We also use the notation $CHR^{\omega_a}(P)$ to denote the language where all constraints have arity zero (i.e. $\Sigma = \emptyset$). Finally $CHR^{\omega_a}(F)$ indicates the CHR language which allows functor symbols and the = built-in.[3]

3.1 Range-restricted $CHR^{\omega_a}(C)$

We consider the (multi-headed) range-restricted $CHR^{\omega_a}(C)$ language. We call a CHR rule range-restricted if all the variables which appear in the body and in the guard appear also in the head of a rule. More formally, if $Var(X)$ denotes the variables used in X, the rule $r\ @H^k \backslash H^h \iff D \mid B$ is range-restricted if $Var(B) \cup Var(D) \subseteq Var(H^k, H^h)$ holds. A CHR language is called range-restricted if it allows range-restricted rules only.

We prove that in range-restricted $CHR^{\omega_a}(C)$ the existence of an infinite computation is a decidable property. This shows that this language is less expressive than Turing Machines and than $CHR^{\omega_a}(C)$. Our result is based on the theory of well-structured transition systems (WSTS) and we refer to [1, 13] for this theory. Here we only provide the basic definitions on WSTS, taken from [13].

Recall that a *quasi-order* (or, equivalently, preorder) is a reflexive and transitive relation. A *well-quasi-order* (wqo) is defined as a quasi-order $\leq$ over a set X such that, for any infinite sequence $x_0, x_1, x_2, \ldots$ in X, there exist indexes $i < j$ such that $x_i \leq x_j$. Thus well-quasi-orders exclude the possibility of having infinite strictly decreasing sequences.

A *transition system* is defined as usual, namely it is a structure $TS = (S, \rightarrow)$, where S is a set of *states* and $\rightarrow \subseteq S \times S$ is a set of *transitions*. We define $Succ(s)$ as the set $\{s' \in S \mid s \rightarrow s'\}$ of immediate successors of s. We say that TS is *finitely branching* if, for each $s \in S$, $Succ(s)$ is finite. Hence we have the key definition.

Definition 8 (Well-structured transition system with strong compatibility)**.** *A well-structured transition system with strong compatibility is a transition system $TS = (S, \rightarrow)$, equipped with a quasi-order $\leq$ on S, such that the two following conditions hold:*

1. *$\leq$ is a well-quasi-order;*

[3]Note that this last language is the signature used in most of the current CHR implementation. Indeed the host language of the majority of CHR implementations is Prolog and therefore the usual signature supports arbitrary Herbrand terms and unification.

2. *$\leq$ is strongly (upward) compatible with $\rightarrow$, that is, for all $s_1 \leq t_1$ and all transitions $s_1 \rightarrow s_2$, there exists a state t_2 such that $t_1 \rightarrow t_2$ and $s_2 \leq t_2$ holds.*

The next theorem is a special case of a result in [13] and will be used to obtain our decidability result.

Theorem 1. *Let $TS = (S, \rightarrow, \leq)$ be a finitely branching, well-structured transition system with strong compatibility, decidable $\leq$ and computable $Succ(s)$ for $s \in S$. Then the existence of an infinite computation starting from a state $s \in S$ is decidable.*

Consider a given goal G and a (CHR) program P and consider the transition system $T = (Conf, \overset{\omega_a}{\rightarrow}_P)$ defined in Section 2.4. Obviously the number of constants and variables appearing in G or in P is finite. Moreover, observe that since we consider range-restricted programs, the application of the transitions $\overset{\omega_a}{\rightarrow}_P$ does not introduce new variables in the computations. In fact, even though rules are renamed (in order to avoid clash of variables), the definition of the Apply rule (in particular the definition of θ) implies that in a transition $s_1 \overset{\omega_a}{\rightarrow}_P s_2$ we have that $Var(s_2) \subseteq Var(s_1)$ holds. Hence an obvious inductive argument implies that no new variables arise in computations. For this reason, given a goal G and a program P, we can assume that the set $Conf$ of all the configurations uses only a finite number of constants and variables. In the following we implicitly make this assumption. We define a quasi-order on configurations as follows.

Definition 9. *Given two configurations $s_1 = \langle G_1, S_1, B_1 \rangle_i$ and $s_2 = \langle G_2, S_2, B_2 \rangle_j$ we say that $s_1 \leq s_2$ if*

- *for every constraint $c \in G_1$ $|\{c \in G_1\}| \leq |\{c \in G_2\}|$*
- *for every constraint $c \in \{d \,.\, d\#i \in S_1\}$ $|\{i \,.\, c\#i \in S_1\}| \leq |\{i \,.\, c\#i \in S_2\}|$*
- *B_1 is logically equivalent to B_2*

It is possible to prove that $\leq$ is a well-quasi-order on $Conf$ and that given a $CHR^{\omega_a}(C)$ program P, $(Conf, \overset{\omega_a}{\rightarrow}_P, \leq)$ is a well-structured transition system with strong compatibility. For more details please see [20].

Exploiting the WSTS we have the desired result.

Theorem 2. *Given a range-restricted $CHR^{\omega_a}(C)$ program P and a goal G, the existence of an infinite computation for G in P is decidable.*

Proof. First observe that, due to our assumption on range-restricted programs, $T = (Conf, \overset{\omega_a}{\rightarrow}_P)$ is finitely branching. In fact, as previously mentioned, the use of rule Apply can not introduce new variables (and hence new different states). The thesis follows immediately from the strong compatibility of $(Conf, \overset{\omega_a}{\rightarrow}_P, \leq)$ and Theorem 1. □

The previous Theorem implies that range-restricted $CHR^{\omega_a}(C)$ is strictly less expressive than Turing Machines, in the sense that there can not exist a termination preserving encoding of Turing Machines into range-restricted $CHR^{\omega_a}(C)$. To be more precise, we consider an encoding of a Turing Machine into a CHR language as a function f which, given a machine Z and an initial instantaneous description D for Z, produces a CHR program and a goal. This is denoted by $(P, G) = f(Z, D)$. Hence we have the following.

Definition 10 (Termination preserving encoding)**.** *An encoding f of Turing Machines into a CHR language is termination preserving*[4] *if the following holds: the machine Z starting with D terminates iff the goal G in the CHR program P has only terminating computations, where $(P, G) = f(Z, D)$. The encoding is weak termination preserving if: the machine Z starting with D terminates iff the goal G in the CHR program P has at least one terminating computation.*

Since termination is undecidable for Turing Machines, we have the following immediate corollary of Theorem 2.

Corollary 1. *There exists no termination preserving encoding of Turing Machines into range-restricted $CHR^{\omega_a}(C)$.*

Note that the previous result does not exclude the existence of weak encodings. For example, in [8] it is shown that the existence of an infinite computation is decidable in CCS!, a variant of CCS, yet it is possible to provide a weak termination preserving encoding of Turing Machines in CCS! (essentially by adding spurious non-terminating computations). We conjecture that such an encoding is not possible for $CHR^{\omega_a}(C)$. Note also that previous results imply that range-restricted $CHR^{\omega_a}(C)$ is strictly less expressive than $CHR^{\omega_a}(C)$: in fact there exists a termination preserving encoding of Turing Machines into $CHR^{\omega_a}(C)$ [11, 31].

3.2 Overview

We proved that range-restricted $CHR^{\omega_a}(C)$ is strictly less expressive than Turing Machines (and therefore than $CHR^{\omega_a}(C)$). In [20] a similar result is proven for

[4] For many authors the existence of a termination preserving encoding into a non-deterministic language L is equivalent to the Turing completeness of L, however there is no general agreement on this, since for others a weak termination preserving encoding suffices.

$CHR^{\omega_a}_1(C)$, the language defined over a signature which does not allow function symbols but only one constraint in the head. These results are not immediate. Indeed, $CHR^{\omega_a}(C)$, without further restrictions and with any of the two semantics, is a Turing complete language [11,31]. It remains quite expressive also with these restrictions: for example, $CHR^{\omega_a}_1(C)$ allows an infinite number of different states, hence, for example, it can not be translated to Petri Nets.

Several papers have considered the expressive power of CHR in the last few years. In particular, [31] showed that a further restriction of $CHR^{\omega_a}_1(C)$, which does not allow built-ins in the body of rules (and which therefore does not allow unification of terms) is not Turing complete. This result is obtained by translating $CHR^{\omega_a}_1(C)$ programs (without unification) into propositional CHR and using the encoding of propositional CHR intro Petri Nets provided in [2]. The translation to propositional CHR is not possible for the language (with unification) $CHR^{\omega_a}_1(C)$. [2] also provides a translation of range-restricted $CHR^{\omega_a}(C)$ to Petri nets. However in this translation, differently from our case, it is also assumed that no unification built-in can be used in the rules, and only ground goals are considered. Related to this work is also [11], where it is shown that $CHR^{\omega_a}(F)$ is Turing complete and that restricting to single-headed rules decreases the computational power of CHR. However, these results are based on the theory of language embedding, developed in the field of concurrency theory to compare Turing complete languages, hence they do not establish any decidability result. Another related study is [32], where the authors show that it is possible to implement any algorithm in CHR in an efficient way, i.e. with the best known time and space complexity. Earlier works by Frühwirth [17, 18] studied the time complexity of simplification rules for naive implementations of CHR. In this approach an upper bound on the derivation length, combined with a worst-case estimate of (the number and cost of) rule application attempts, allows to obtain an upper bound of the time complexity.

A summary of the existing results concerning the computational power of several dialects of CHR is shown in Table 4. In this table, "no" and "yes" refer to the existence of a termination preserving encoding of Turing Machines into the considered language.

4 Expressive power of priorities in CHR

In this section we first show that dynamic priorities do no augment the expressive power of the language w.r.t. static priorities. This result is obtained by providing an acceptable encoding of CHR with priorities into the fragment of CHR that uses static priorities only. We then prove that (static) priorities augment the expressive power of CHR in the sense that there exists no acceptable encoding of CHR^{ω_p} into

Signature	*Operational semantics*	$k = 1$	$k > 1$
P (propositional)	ω_a	No	No
range-restricted C (constants)	ω_a	No	No
C (constants), without =	ω_a and ω_t	No	Yes
C (constants)	ω_a and ω_t	No	Yes
F (functors)	ω_a and ω_t	Yes	Yes

Table 4: Summary of termination preserving encoding of Turing Machines

CHR (with the ω_t semantics). To conclude we give an overview of the state of the art related to the expressive power equivalence between different CHR languages.

We consider the following languages and semantics:

- CHR^{ω_t}: this is standard CHR, where the theoretical semantics is used,
- CHR^{ω_p}: this is CHR with priorities, where both dynamic and static priorities can be used, the semantics is ω_p defined in Section 2.5;
- *static* CHR^{ω_p}: this is CHR with static priorities only, with the ω_p semantics;
- *static* $CHR_2^{\omega_p}$: this is CHR with static priorities only, with the ω_p semantics, where we allow at most two constraints in the head of a rule.

In the following, given a program P, we denote by $Pred(P)$ and $Head(P)$ the set of all the predicate symbols p s.t. p occurs in P and in the head of a rule in P, respectively.

4.1 Encoding CHR^{ω_p} into *static* CHR^{ω_p}

We prove that the CHR^{ω_p} language, which allows dynamic priorities, is not more expressive than *static* CHR^{ω_p}, which allows static priorities only. This result is obtained by providing an (acceptable) encoding of CHR^{ω_p} into *static* CHR^{ω_p}.

We assume that P is a CHR^{ω_p} program composed by m rules and we also assume that the i-th rule (with $i \in \{1, \ldots, m\}$) has the form:

$$p_i :: rule_i \ @ \ H_i \backslash H_i' \Leftrightarrow G_i | B_i$$

Moreover, given a multiset of CHR constraints $\bar{H} = h_1(\bar{t}_1), \ldots, h_n(\bar{t}_n)$ and a sequence of (distinct) variables $\bar{V} = V_1, \ldots, V_n$, we denote by $new'(\bar{H}, \bar{V})$ the multiset of atoms $new_{h_1}(V_1, \bar{t}_1), \ldots, new_{h_n}(V_n, \bar{t}_n)$.

First, we require that the goal encoding is a non surjective function. The reason for this requirement is that the program encoding needs to use, in the translated program, some fresh constraints which do not appear in the initial (translated) goal. A simple goal encoding that satisfies this requirement is the one that does not change built-in constraints and adds a letter, say "a", at the beginning of the other constraints, as shown below

$$\mathcal{INP}(b(\bar{t})) = \begin{cases} b(\bar{t}) & \text{if } b(\bar{t}) \text{ is a built-in constraint} \\ ab(\bar{t}) & \text{otherwise} \end{cases}$$

The program encoding $\mathcal{T}(P)$ from CHR^{ω_p} into $static\ CHR^{\omega_p}$ is instead defined as the function that, given a program P, produces the following program:

$$\begin{gathered} \text{for every predicate name } ak \in \mathcal{INP}(Head(P)) \\ 1 :: rule_{(1,k)} \ @\ start \backslash id(V), ak(\bar{X}) \Leftrightarrow id(V+1), new_{ak}(V, \bar{X}) \\ 2 :: rule_{(2,k)} \ @\ ak(\bar{X}) \Rightarrow start, id(0) \end{gathered}$$

$$2 :: rule_3 \ @\ start \Leftrightarrow highest\_priority(inf)$$

$$\begin{gathered} \text{for every } i \in \{1, \ldots, m\} \\ 3 :: rule_{(4,i)} \ @\ end \backslash instance_i(\_) \Leftrightarrow true \end{gathered}$$

$$4 :: rule_5 \ @\ end \Leftrightarrow true$$

$$\text{for every } i \in \{1, \ldots, m\} \quad \texttt{EVALUATE\_PRIORITIES}(i)$$

$$7 :: rule_9 \ @\ highest\_priority(inf), id(V) \Leftrightarrow end$$

$$\text{for every } i \in \{1, \ldots, m\} \quad \texttt{ACTIVATE\_RULE}(i)$$

If $rule_i$ is not a propagation rule then $\texttt{EVALUATE\_PRIORITIES}(i)$ are the following rules

$$6 :: rule_{(7,i)} \ @ \ new'(\mathcal{INP}(H_i), \bar{Z}), new'(\mathcal{INP}(H'_i), \bar{U}) \backslash highest\_priority(inf) \Leftrightarrow G_i | highest\_priority(p_i)$$

$$6 :: rule_{(8,i)} \ @ \ new'(\mathcal{INP}(H_i), \bar{Z}), new'(\mathcal{INP}(H'_i), \bar{U}) \backslash highest\_priority(P) \Leftrightarrow G_i, p_i < P | highest\_priority(p_i)$$

if $rule_i$ is a propagation rule then `EVALUATE_PRIORITIES`(i) are the following rules

$$5 :: rule_{(6,i)} \ @ new'(\mathcal{INP}(H_i), \bar{Z}) \Rightarrow G_i | instance_i(\bar{Z})$$

$$6 :: rule_{(7,i)} \ @ \ instance_i(\bar{Z}), new'(\mathcal{INP}(H_i), \bar{Z}) \backslash highest\_priority(inf) \Leftrightarrow G_i | highest\_priority(p_i)$$

$$6 :: rule_{(8,i)} \ @ \ instance_i(\bar{Z}), new'(\mathcal{INP}(H_i), \bar{Z}) \backslash highest\_priority(P) \Leftrightarrow G_i, p_i < P | highest\_priority(p_i)$$

if $rule_i$ is a propagation rule then `ACTIVATE_RULE`(i) is the following rule

$$8 :: rule_{(10,i)} \ @ \ new'(\mathcal{INP}(H_i), \bar{Z}) \backslash instance_i(\bar{Z}), highest\_priority(P), id(V) \Leftrightarrow G_i, p_i = P | Update(\mathcal{INP}(B_i), V), highest\_priority(inf)$$

if $rule_i$ is not a propagation rule then `ACTIVATE_RULE`(i) is the following rule

$$8 :: rule_{(10,i)} \ @ \ new'(\mathcal{INP}(H_i), \bar{Z}), new'(\mathcal{INP}(H'_i), \bar{U}), highest\_priority(P), id(V) \Leftrightarrow G_i, p_i = P | Update(\mathcal{INP}(B_i), V), highest\_priority(inf)$$

In the above encoding we assume that the constraint theory $\mathcal{CT}$ allows to use equalities and inequalities (so we can evaluate whether $p_i = h$ and $p_i > h$ where $h \in \mathbb{Z}$ and p_i is an arithmetic expression). We also assume inf is a conventional constant which is bigger than all p_i (i.e. it represents the lowest priority). The $Update(C, V)$ function is defined instead as follows

$$Update(k(\bar{t}), V) = new_k(V, \bar{t})$$
if $k(\bar{t})$ is a CHR constraint

$$Update(c(\bar{t}), V) = c(\bar{t})$$
if $c(\bar{t})$ is a built-in constraint

$$Update([\,], V) = id(V)$$

$$Update([d(\bar{X}) \mid Ds], V) = Update(d(\bar{X}), V), Update(Ds, V + 1).$$

Example 4.1. *L*et us consider as P the shortest path program depicted in Figure 2. The correspondent $\mathcal{T}(P)$ is the following program:

$$1 :: start\backslash id(V), asource(\bar{X}) \Leftrightarrow id(V+1), new_{asource}(V, \bar{X})$$
$$1 :: start\backslash id(V), adist(\bar{X}) \Leftrightarrow id(V+1), new_{adist}(V, \bar{X})$$
$$1 :: start\backslash id(V), aedge(\bar{X}) \Leftrightarrow id(V+1), new_{aedge}(V, \bar{X})$$

$$2 :: asource(\bar{X}) \Rightarrow start, id(0)$$
$$2 :: adist(\bar{X}) \Rightarrow start, id(0)$$
$$2 :: aedge(\bar{X}) \Rightarrow start, id(0)$$

$$2 :: start \Leftrightarrow highest\_priority(inf)$$

$$3 :: end\backslash instance_1(\bar{Z}) \Leftrightarrow true$$
$$3 :: end\backslash instance_2(\bar{Z}) \Leftrightarrow true$$
$$3 :: end\backslash instance_3(\bar{Z}) \Leftrightarrow true$$

$$4 :: end \Leftrightarrow true$$

$$5 :: new_{asource}(V, X) \Rightarrow instance_1(V)$$
$$6 :: new_{asource}(V, X)\backslash highest\_priority(inf) \Leftrightarrow highest\_priority(1)$$
$$6 :: new_{asource}(V, X)\backslash highest\_priority(P) \Leftrightarrow 1 < P | highest\_priority(1)$$

$$6 :: new_{adist}(V_1, X_1, X_2), new_{adist}(V_2, Y_1, Y_2)\backslash highest\_priority(inf) \Leftrightarrow X_2 \leq Y_2 | highest\_priority(1)$$
$$6 :: new_{adist}(V_1, X_1, X_2), new_{adist}(V_2, Y_1, Y_2)\backslash highest\_priority(P) \Leftrightarrow X_2 \leq Y_2, 1 < P | highest\_priority(1)$$

$$5 :: new_{adist}(V_1, X_1, X_2), new_{aedge}(V_2, \bar{Y}) \Rightarrow instance_3(V_1, V_2)$$
$$6 :: new_{adist}(V_1, X_1, X_2), new_{aedge}(V_2, \bar{Y})\backslash highest\_priority(inf) \Leftrightarrow highest\_priority(X_2 + 2)$$
$$6 :: new_{adist}(V_1, X_1, X_2), new_{aedge}(V_2, \bar{Y})\backslash highest\_priority(P) \Leftrightarrow X_2 + 2 < P | highest\_priority(X_2 + 2)$$

$$7 :: highest\_priority(inf), id(V) \Leftrightarrow end$$

$$8 :: new_{asource}(V, X)\backslash instance_1(\bar{V}), highest\_priority(P), id(V') \Leftrightarrow 1 = P | new_{adist}(V', X, 0), id(V'+1), highest\_priority(inf)$$
$$8 :: new_{adist}(V_1, X, X_1)\backslash new_{adist}(V_2, X, X_2), highest\_priority(P), id(V') \Leftrightarrow X_1 \leq X_2, 1 = P | id(V'), highest\_priority(inf)$$
$$8 :: new_{adist}(V_1, X, X_1), new_{aedge}(V_2, X, X_2, X_3)\backslash instance_3(V_1, V_2), highest\_priority(P), id(V') \Leftrightarrow X_1 + 2 = P | new_{adist}(X_3, X_1 + X_2), id(V'+1), highest\_priority(inf)$$

□

We now provide some explanations for the above encoding. Intuitively the result of the encoding can be divided in three phases:

1. **Init.** In the init phase, for each (user defined) predicate symbol $ak \in \mathcal{INP}(Head(P))$ we introduce a rule $rule_{(1,k)}$, which replaces $ak(\bar{t})$ distinct variables) by $new_{ak}(V, \bar{t})$ where V is a variable which will be used to simulate the identifier used in identified constraints. Moreover we use the *id* predicate symbol to memorize the highest identifier used. Rules $rule_{(2,k)}$ (one for each predicate symbol $ak \in \mathcal{INP}(Head(P))$, as before) are used to fire rules $rule_{(1,k)}$ and also to start the following phase (via $rule_3$). Note that rules $rule_{(1,k)}$ have maximal priority and therefore are tried before rules $rule_{(2,k)}$.

2. **Main.** The main phase is divided into two phases: the *evaluation* phase starts when the init phase adds the constraint $highest\_priority(inf)$. Rules $rule_{(6,i)}, \ldots, rule_{(8,i)}$ store in *highest_priority* the highest priority on all the rule instances that can be fired. After the end of the evaluation phase the *activation* starts. During this phase if a rule can be fired one of the rules $rule_{(10,i)}$ is fired. After the rule has been fired the constraint $highest\_priority(inf)$ is produced which starts a new evaluation phase.

3. **Termination.** The termination phase is triggered by rule $rule_9$. This rule fires when no instance from the original program can fire. During the termination phase all the constraints produced during the computation (namely *id*, $instance_i$, *highest_priority*, *end*) are deleted.

In the following we now provide some more details on the two crucial points in this translation: the evaluation and the activation phases.

- **Evaluation.** The rules in the set denoted by

  $$\texttt{EVALUATE\_PRIORITIES}(i)$$

  are triggered by the insertion of $highest\_priority(inf)$ in the constraint store. In the case of a propagation rule $rule_i \in P$, the rules in

  $$\texttt{EVALUATE\_PRIORITIES}(i)$$

  should consider the possibility that there is an instance of $rule_i$ that can not be fired because it has been previously fired. When an instance of a propagation rule can fire, rule $rule_{(6,i)}$ adds a constraint $instance_i(\bar{v})$, where $\bar{v}$ are the identifiers of the CHR atoms which can be used to fire $rule_i$. The absence of the constraint $instance_i(\bar{v})$ in the constraint store means that either $rule_i$ can not be fired by using the CHR atoms identified by $\bar{v}$ or has already fired for the CHR atoms identified by $\bar{v}$.

  The evaluation of the priority for a simpagation or a simplification rule is instead more simple because the propagation history does not affect the execution of these two types of rules.

Rules $rule_{(7,i)}$ and $rule_{(8,i)}$ replace the constraint $highest\_priority(p)$ with the constraint $highest\_priority(p')$ if a rule of priority p' can be fired and $p > p'$.

- **Activation.** When the evaluation phase ends if a rule can fire then one of the rules $rule_{(10,i)}$ is fired since $highest\_priority(inf)$ has been removed from the constraint store.

  The only difference between a propagation rule and a simpagation/simplification rule is that when a propagation rule is fired the corresponding constraint $instance_i(\bar{v})$ is deleted to avoid the execution of the same propagation rule in the future.

  It is worth noting that the non-determinism in the choice of the rule to be fired provided by the ω_p semantics is preserved, since all the priorities of `ACTIVATE_RULE`(i) are equal.

To conclude the definition of the acceptable encoding we need the last ingredient: the output decoding function. If we run the goal $\mathcal{INP}(G)$ in the program $\mathcal{T}(P)$ we obtain the same qualified answers obtained by running G in the program P, with the only difference that if in the qualified answer of P there is a CHR constraint $k(\bar{t})$ then in the corresponding qualified answer of the encoded program $\mathcal{T}(P)$ there will be either a constraint $new_{ak}(V, \bar{t})$ (if $k \in Head(P)$ or $k(\bar{t})$ is introduced by an Apply transition step) or a constraint $ak(\bar{t})$ (if $k \notin Head(P)$ and $k(\bar{t})$ is in the initial goal G).

Therefore the decoding function that we need is:

$$\mathcal{OUT}(b(\bar{t})) = \begin{cases} b(\bar{t}) & \text{if } b(\bar{t}) \text{ is a built-in constraint} \\ k(\bar{t}') & \text{if } b(\bar{t}) = new_{ak}(V, \bar{t}') \\ k(\bar{t}) & \text{if } b(\bar{t}) = ak(\bar{t}). \end{cases}$$

The following result proven in [19] shows that the qualified answers are preserved by our encoding.

Theorem 3. *The triple $(\mathcal{T}(), \mathcal{INP}(), \mathcal{OUT}())$ provides an acceptable encoding between CHR^{ω_p} and static CHR^{ω_p}.*

4.2 No encoding of *static* CHR^{ω_p} into CHR^{ω_t}

In this section we prove that priorities do augment the expressive power of CHR. To do so we prove that there exists no acceptable encoding from *static* CHR^{ω_p} into CHR^{ω_t}.

In order to prove this separation result we need the following lemma which states a key property of CHR computations under the ω_t semantics. Essentially

it says that, given a program P and goal G, if there exists a derivation for G in P which produces a qualified answer (d, K) where d is a built-in constraint, then when considering the goal (d, G) we can perform a derivation in P, which is essentially the same of the previous one, with the only exception of a Solve transition step (in order to evaluate the constraint d). Hence it is easy to observe that such a new computation for (d, K) in P will terminate producing the same qualified answer (d, K).

The proof of the following Lemma is then immediate.

Lemma 1. *Let P be a CHR^{ω_t} program and let G be a goal. Assume that G in P has the qualified answer (d, K). Then the goal (d, G) has the same qualified answer (d, K) in P.*

Lemma 1 is not true anymore if we consider CHR^{ω_p} programs. Indeed if we consider the program P consisting of the rules

$$1 :: h(X) \Leftrightarrow X = yes | false$$

$$2 :: h(X) \Leftrightarrow X = yes$$

then the goal $h(X)$ has the qualified answer $X = yes$ in P, while the goal $X = yes, h(X)$ has no qualified answer in P. With the help of the previous lemma we can now prove our main separation result.

Theorem 4. *There exists no acceptable encoding for data sufficient answers from CHR^{ω_p} into CHR^{ω_t}. class $\mathcal{G}$.*

Proof. The proof is by contradiction. Consider the following program P in CHR^{ω_p}

$$1 :: h(X) \Leftrightarrow X = yes | false$$

$$2 :: h(X) \Leftrightarrow X = yes$$

and assume that $(\gamma(), \mathcal{INP}(), \mathcal{OUT}())$ is an acceptable encoding for data sufficient answers from CHR^{ω_p} into CHR^{ω_t}.

Let G be the goal $h(X)$. Then $\mathcal{SA}_P(G) = \{X = yes\}$. Since the goal $h(X)$ has the data sufficient answer $X = yes$ in the program P and since the encoding preserves data sufficient answers, $\mathcal{QA}_{\gamma(P)}(\mathcal{INP}(a(X)))$ contains a qualified answer S such that $\mathcal{OUT}(S) = (X = yes)$. Moreover, since the output decoding function is such that the built-ins appearing in the answer are left unchanged, we have that S is of the form $(X = yes, K)$, where K is a (possibly empty) multiset of CHR constraints.

Then since the goal encoding function is such that the built-ins present in the goal are left unchanged $\mathcal{INP}(X = yes, h(X)) = (X = yes, \mathcal{INP}(h(X)))$ and

therefore from previous Lemma 1, it follows that the program $\gamma(P)$ with the goal $\mathcal{INP}(X = yes, h(X))$ has the qualified answer S.

However $(X = yes, h(X))$ has no data sufficient answer in the original program P. This contradicts the fact that $(\gamma(), \mathcal{INP}(), \mathcal{OUT}())$ is an acceptable encoding for data sufficient answers from CHR^{ω_p} into CHR^{ω_t}, thus concluding the proof. □

Since the existence of an acceptable encoding implies the existence of an acceptable encoding for data sufficient answers we have the following immediate corollary:

Corollary 2. *There exists no acceptable encoding from CHR^{ω_p} into CHR^{ω_t}.*

4.3 Overview

Some immediate acceptable encodings derive directly from the language definitions. Indeed, when a language $\mathcal{L}$ is a sublanguage of $\mathcal{L}'$ then a tern of identity functions provides an acceptable encoding between the two languages. We first observe that *static* $CHR_2^{\omega_p}$ is a sublanguage of *static* CHR^{ω_p} that, in its turn, is a sublanguage of CHR^{ω_p}. Therefore we have the following.

Fact 1. *There exists acceptable encodings from static $CHR_2^{\omega_p}$ to static CHR^{ω_p}, and from static CHR^{ω_p} to CHR^{ω_p}.*

As far as CHR^{ω_t} is concerned, at a first glance it could be considered as a sublanguage of *static* CHR^{ω_p} where all the rules have equal priority. However this is not completely true since in the ω_p semantics, for the application of an Apply transition, the goal multiset of the configuration must be empty while in the ω_t semantics it is possible to fire a rule even though some constraints have not being introduced into the CHR store by a Solve or an Introduce transition. However it is easy to see that from the monotonicity of ω_t it follows that for every computation reaching a non-failed final configuration there is one computation reaching the same final configuration where the Solve and Introduce transitions are performed as soon as they can be executed. Hence every final configuration reached by a CHR^{ω_t} program P can be reached by the *static* CHR^{ω_p} program having the same rules as P with a fixed and constant priority. Therefore we have the following.

Fact 2. *There exists an acceptable encoding from CHR^{ω_t} to static CHR^{ω_p}.*

In [19] it is shown that, differently from the case of standard CHR, allowing more than two atoms in the head of rules does not augment the expressive power of the language. Moreover, as proven in Section 4.1, dynamic priorities do not increase the expressive power w.r.t. static ones.

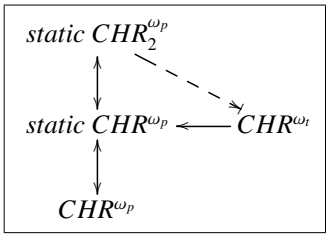


Figure 3: Graphical summary:
⇢: absence of an acceptable encoding
→: presence of an acceptable encoding

On the other hand we have proven that, when considering the theoretical semantics, there exists no acceptable encoding of CHR with (static) priorities into standard CHR. This means that, even though both languages are Turing powerful, priorities augment the expressive power of the language in a quite reasonable sense. For a graphical overview of the expressive power of CHR with and without priority we refer to Figure 3.

Our notion of acceptable encoding has been recently used in [3] to justify a source-to-source transformation. When instead we move to the more general field of concurrent languages one can find several works related to the present one. In particular, concerning priorities, [35] shows that the presence of priorities in process algebras does augment the expressive power. More precisely the authors show, among other things, that a finite fragment of asynchronous CCS with (global) priority can not be encoded into π-calculus nor in the broadcast based b-π calculus. This result is related to our separation result for CHR^{ω_p} and CHR, even though the formal setting is completely different.

More generally, often in process calculi and in distributed systems separation results are obtained by showing that a problem can be solved in a language and not in another one (under some additional hypothesis, similar to those used here). For example, in [25] the author proves that there exists no *reasonable* encoding from the π-calculus to the asynchronous π-calculus by showing that the symmetric leader election problem has no solution in the asynchronous version of the π-calculus. A survey on separation results based on this problem can be found in [36].

5 Conclusions

We considered Constraint Handling Rules (CHR), a well known concurrent language that supports constraints as primitive constructs. We studied its expressive power focusing first on some of its fragments and then considering what happens when priorities are added.

There are still plenty of open question to address. For instance one may won-

der if the expressive power of CHR with priorities is equal to the CHR with the refined semantics [12]. Another question to answer could be if range-restricted $CHR^{\omega_a}(C)$ is more expressive that $CHR_1^{\omega_a}(C)$, since the decidability result for the second language is stronger. Moreover there are hundreds of concurrent languages that can be enriched with constraint primitives to improve their expressive power. We are experiencing an increasingly attention towards this kind of tasks, e.g. [5–7], and we expect a continuation of this trend in the feature.

References

[1] Parosh Aziz Abdulla, Karlis Cerans, Bengt Jonsson, and Yih-Kuen Tsay. General decidability theorems for infinite-state systems. In *in Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, LICS'96*, pages 313–321, 1996.

[2] Hariolf Betz. Relating coloured Petri nets to Constraint Handling Rules. In K. Djelloul, G. J. Duck, and M. Sulzmann, editors, *4th Workshop on Constraint Handling Rules*, pages 33–47, Porto, Portugal, 2007.

[3] Hariolf Betz, Frank Raiser, and Thom W. Frühwirth. A complete and terminating execution model for Constraint Handling Rules. *TPLP*, 10(4-6):597–610, 2010.

[4] Stefano Bistarelli, Thom W. Frühwirth, and Michael Marte. Soft constraint propagation and solving in CHRs. In *SAC*, pages 1–5, 2002.

[5] Stefano Bistarelli and Francesco Santini. A Nonmonotonic Soft Concurrent Constraint Language for SLA Negotiation. *Electr. Notes Theor. Comput. Sci.*, 236:147–162, 2009.

[6] Maria Grazia Buscemi and Ugo Montanari. Open Bisimulation for the Concurrent Constraint Pi-Calculus. In Sophia Drossopoulou, editor, *ESOP*, volume 4960 of *Lecture Notes in Computer Science*, pages 254–268. Springer, 2008.

[7] Maria Grazia Buscemi and Ugo Montanari. A survey of constraint-based programming paradigms. *Computer Science Review*, 2(3):137–141, 2008.

[8] Nadia Busi, Maurizio Gabbrielli, and Gianluigi Zavattaro. Comparing Recursion, Replication, and Iteration in Process Calculi. In *ICALP*, pages 307–319, 2004.

[9] Frank S. de Boer, Maurizio Gabbrielli, and Maria Chiara Meo. A Temporal Logic for reasoning about Timed Concurrent Constraint Programs. In *TIME*, pages 227–233, 2001.

[10] Frank S. de Boer and Catuscia Palamidessi. Embedding as a tool for language comparison. *Inf. Comput.*, 108(1):128–157, 1994.

[11] Cinzia Di Giusto, Maurizio Gabbrielli, and Maria Chiara Meo. Expressiveness of multiple heads in CHR. In *SOFSEM*, pages 205–216, 2009.

[12] Gregory J. Duck, Peter J. Stuckey, Maria J. García de la Banda, and Christian Holzbaur. The refined operational semantics of Constraint Handling Rules. In *ICLP*, pages 90–104, 2004.

[13] Alain Finkel and Ph. Schnoebelen. Well-structured transition systems everywhere! *Theor. Comput. Sci.*, 256(1-2):63–92, 2001.

[14] Thom Frühwirth. Temporal reasoning with Constraint Handling Rules. Technical Report ECRC-94-5, European Computer-Industry Research Centre, Munchen, Germany, 1994.

[15] Thom Frühwirth. *Constraint Handling Rules*. August 2009.

[16] Thom W. Frühwirth. Theory and practice of Constraint Handling Rules. *J. Log. Program.*, 37(1-3):95–138, 1998.

[17] Thom W. Frühwirth. As Time Goes by: Automatic Complexity Analysis of Simplification Rules. In *KR*, pages 547–557, 2002.

[18] Thom W. Frühwirth and Slim Abdennadher. The Munich Rent Advisor: A Success for Logic Programming on the Internet. *TPLP*, 1(3):303–319, 2001.

[19] Maurizio Gabbrielli, Jacopo Mauro, and Maria Chiara Meo. The expressive power of chr with priorities. *Inf. Comput.*, 228:62–82, 2013.

[20] Maurizio Gabbrielli, Jacopo Mauro, Maria Chiara Meo, and Jon Sneyers. Decidability properties for fragments of CHR. *TPLP*, 10(4-6):611–626, 2010.

[21] Maurizio Gabbrielli, Catuscia Palamidessi, and Frank D. Valencia. Concurrent and Reactive Constraint Programming. In *25 Years GULP*, pages 231–253, 2010.

[22] Leslie De Koninck, Tom Schrijvers, and Bart Demoen. User-definable rule priorities for CHR. In *PPDP*, pages 25–36, 2007.

[23] Michael J. Maher. Logic semantics for a class of committed-choice programs. In *ICLP*, pages 858–876, 1987.

[24] Jacopo Mauro. *Constraints meet concurrency*. PhD thesis, University of Bologna, 2012.

[25] Catuscia Palamidessi. Comparing the expressive power of the synchronous and asynchronous pi-calculi. *Mathematical Structures in Computer Science*, 13(5):685–719, 2003.

[26] Catuscia Palamidessi and Frank D. Valencia. A Temporal Concurrent Constraint Programming Calculus. In *CP*, pages 302–316, 2001.

[27] Vijay A. Saraswat, Radha Jagadeesan, and Vineet Gupta. Default Timed Concurrent Constraint Programming. In *POPL*, pages 272–285, 1995.

[28] Vijay A. Saraswat and Martin C. Rinard. Concurrent Constraint Programming. In *POPL*, pages 232–245, 1990.

[29] Ehud Y. Shapiro. The family of concurrent logic programming languages. *ACM Comput. Surv.*, 21(3):413–510, 1989.

[30] Gert Smolka. The Oz Programming Model. In *Computer Science Today*, pages 324–343. 1995.

[31] Jon Sneyers. Turing-complete subclasses of CHR. In *ICLP*, pages 759–763, 2008.

[32] Jon Sneyers, Tom Schrijvers, and Bart Demoen. The computational power and complexity of Constraint Handling Rules. *ACM Trans. Program. Lang. Syst.*, 31(2), 2009.

[33] Kazunori Ueda. Guarded Horn Clauses. In *LP*, pages 168–179, 1985.

[34] Frits W. Vaandrager. Expressive Results for Process Algebras. In *REX Workshop*, pages 609–638, 1992.

[35] Cristian Versari, Nadia Busi, and Roberto Gorrieri. On the Expressive Power of Global and Local Priority in Process Calculi. In *CONCUR*, pages 241–255, 2007.

[36] Maria Grazia Vigliotti, Iain Phillips, and Catuscia Palamidessi. Tutorial on separation results in process calculi via leader election problems. *Theor. Comput. Sci.*, 388(1-3):267–289, 2007.

News and Conference Reports

REPORT ON ICALP 2013

The 40th International Colloquium on Automata, Languages and Programming

Luca Aceto
School of Computer Science, Reykjavik University

The 40th International Colloquium on Automata, Languages and Programming (ICALP), the main conference and annual meeting of the European Association for Theoretical Computer Science (EATCS), took place from the 8th to the 12th of July 2013 in Riga, Latvia. The main conference was preceded by a series of workshops, which took place on Sunday, 7 July. This year, the following five workshops were co-located with ICALP:

- Workshop on Automata, Logic, Formal languages, and Algebra (ALFA'13);
- International Workshop on Approximation, Parameterized and EXact algorithms (APEX 2013);
- Foundations of Network Science (FONES);
- Learning Theory and Complexity; and
- Workshop on Quantum and Classical Complexity.

ICALP 2013 marked two milestones in the history of the ICALP conference series: it was the first ICALP conference that has been organized in a country of the former Soviet Union and it was the 40th ICALP conference.

First of all, let me express my heartfelt thanks to the local organizers, who did their very best to make the conference a festive occasion and a very pleasant experience for all the attendees. The city of Riga is very pretty and all ICALP participants had a very warm welcome. The University of Latvia was also enrolling a new batch of students during ICALP and this meant that there was a continuous flow of young and happy-looking people on the university premises. This made the main hall of the university a very lively place to be. We were also blessed with sunny and warm weather, which also helped to lift the spirits of the conference participants.

According to the data presented by Agnis Skuskovniks, on behalf of the organizing committee, to the EATCS council and to the EATCS general assembly, there were 193 registered participants in ICALP 2013, of which 67 were students. Including the local participants, the six invited speakers, and the two recipients

of the honorary doctorates awarded on Tuesday, 9 July 2013, a total of 217 people attended ICALP 2013. The pre-conference workshops were attended by 102 participants, which is a very healthy number.

ICALP 2013 was an action packed conference and I think that it was a great success, both scientifically and socially. The scientific programme for ICALP 2013 consisted of six invited lectures (two of which were delivered by women), the presentation of 124 contributed papers (which were selected by the program committees out of 423 submissions) and an award session, where, in addition to the presentation of the EATCS and the Presburger awards, honorary doctorates were bestowed on Jozef Gruska and Juris Hartmanis.

The 124 contributed papers for ICALP 2013 were divided into the three tracks of ICALP 2013 as follows:

- 71 papers for "Track A: Algorithms, Complexity and Games", which were selected from 249 submissions;
- 33 papers for "Track B: Logic, Semantics, Automata and Theory of Programming", which were selected from 114 submissions; and
- 20 papers for "Track C: Foundations of Networked Computation", which were selected from 60 submissions.

The PC chairs for the three tracks of ICALP 2013 were Fedor V. Fomin (Track A), Marta Kwiatkowska (Track B) and David Peleg (Track C). I take this opportunity of thanking them, their PCs and the sub-reviewers for doing an exceptional job. To give you an idea of the amount of work that is involved in the selection of papers for ICALP, the PCs for the three tracks of ICALP 2013 had 71 members in total, 1299 reviews were produced during the PC work, of which 794 reviews were written by 684 external reviewers. It is thus fair to say that a large fraction of the TCS community was actively involved in the PC work for ICALP 2013.

Statistical information about the number of papers submitted and accepted for the last five editions of the ICALP conference, as well as acceptance rates, are available in Tables 1–3. The breakdown by country for each track of ICALP 2013, limited to the top ten countries by number of submissions, is in Tables 4–6. I thank Fedor Fomin, Marta Kwiatkowska and David Peleg for sharing this information with me.

I let you draw your own conclusions on how well each country did at ICALP 2013.

ICALP 2013 in Riga featured five invited presentations and a special EATCS lecture to celebrate the 40th edition of the ICALP conference. The scientific program for the conference was preceded by short presentations by the rector of the University of Latvia and by Rusins Freivalds. The rector welcomed the ICALP

	2013	2012	2011	2010	2009
Total	**124**	123	114	106	108
Track A	**71**	71	68	60	62
Track B	**33**	30	29	30	24
Track C	**20**	22	17	16	22

Table 1: Number of accepted papers at ICALP 2009–2013

	2013	2012	2011	2010	2009
Total	**423**	432	398	389	370
Track A	**249**	248	243	222	223
Track B	**114**	105	103	114	84
Track C	**60**	79	52	53	63

Table 2: Number of submitted papers at ICALP 2009–2013

	2013	2012	2011	2010	2009
Total	29.3	28.5	28.6	27.2	29.2
Track A	28.5	28.6	28	27	27.8
Track B	28.9	28.6	28.2	26.3	28.6
Track C	33.3	27.9	32.7	30.2	34.9

Table 3: Acceptance rates for ICALP 2009–2013

country	authors	submitted	accepted	accept rate	PC members
United States	179	73.61	20.83	0.28	6
Germany	68	30.99	13.99	0.45	1
Israel	41	20.8	6.49	0.31	2
France	28	11.56	3.71	0.32	1
Japan	26	10.75	1.25	0.12	1
China	24	10.58	0	0	0
Sweden	17	8.72	3.3	0.38	0
Poland	13	7.75	2	0.26	0
Canada	25	7.72	2.63	0.34	1
India	21	7.27	2.27	0.31	1

Table 4: ICALP 2013: Breakdown by country (Track A)

country	authors	submitted	accepted	accept rate	PC members
France	52	21.33	7.08	0.33	2
UK	36	17.02	7.07	0.42	5
Germany	27	12.58	5.5	0.44	3
United States	22	11.38	3.83	0.34	3
Italy	21	7.83	1.5	0.19	1
China	12	5.17	1	0.19	1
Czech Republic	10	3.83	0.83	0.22	1
Israel	6	3.75	0.5	0.13	0
Netherlands	9	3.17	0.33	0.11	1
Poland	7	2.35	1.35	0.57	2

Table 5: ICALP 2013: Breakdown by country (Track B)

country	authors	submitted	accepted	accept rate	PC members
United States	40	13.28	2.07	0.16	5
Germany	19	7.05	3.55	0.5	1
Israel	15	6.5	0.83	0.13	2
UK	11	5.02	2.58	0.51	1
Canada	9	3.67	2.33	0.64	1
Italy	15	3.67	1.07	0.29	2
Greece	8	3.03	1.58	0.52	1
France	8	2.6	0.6	0.23	2
Hong Kong	4	2	0	0	0
Switzerland	5	2	0	0	2

Table 6: ICALP 2013: Breakdown by country (Track C)

participants, gave us some interesting information about the University of Latvia and wished us long coffee breaks. Rusins discussed the unity of science, and reminded us that, even at the time of the iron curtain, there was *one* Computer Science.

The conference proper was kicked off on Monday, 8 July, by an invited talk delivered by the Liverpool-bound Paul Spirakis. Paul's talk was entitled *A Guided Tour in Random Intersection Graphs*. Random intersection graphs are random graphs in which there is a universe M of labels and each one of the vertices selects a random subset of M. Two vertices are connected if, and only if, their corresponding subsets of labels intersect. Random intersection graphs were introduced by Karonski, Sheinerman and Singer-Cohen and have several applications, as well as a rich theory. Paul's talk provided a survey of the main results on the topic obtained by his research group on combinatorial problems over random intersection graphs, such as independent sets, Hamiltonian cycles, colouring, maximum cliques, expansion and random walks. Paul closed the talk by saying that this model is an excellent area of study for PhD students in TCS.

The invited presentation for Tuesday, 9 July, was delivered by Dániel Marx. Dániel's talk had three chapters (his words) and was entitled *The Square Root Phenomenon in Planar Graphs*. The starting point of the talk was the observation that most of the classic NP-hard problems remain NP-hard even when restricted to planar graphs, and only exponential-time algorithms are known for the exact solution of these planar problems. However, in many cases, the exponential-time algorithms on planar graphs are significantly faster than the algorithms for general graphs. Indeed, for various problems on planar graphs, one often sees a square root appearing in the exponent of the running time of the best algorithms for their solution. Dániel told his audience that, by now, we have a good understanding of why this square root appears: most of these algorithms rely on the notion of treewidth and its relation to grid minors in planar graphs. Dániel also argued that, under the Exponential Time Hypothesis, one can show that these algorithms are essentially the best possible, and therefore the square root must appear in the running time. (In passing, let me remark that Dániel contributed also one paper to ICALP Track A and one to ICALP Track B!)

Susanne Albers delivered the invited talk on Wednesday, 10 July, on *Recent Advances for a Classical Scheduling Problem*. In her talk, Susanne revisited the classic on-line makespan minimization problem, which has been studied since the 1960s. After presenting the classic results on this problem, starting from Graham's 1966 List algorithm and its competitive analysis, she surveyed recent research on settings in which an online algorithm is given extra information or power while processing a job sequence.

The scientific programme on Thursday, 11 July, started with an invited talk by Orna Kupferman, who gave the only invited address that could be readily classi-

fied as belonging to ICALP Track B. Orna's presentation dealt with *Formalizing and Reasoning about Quality*. Traditionally, formal approaches to the verification of reactive systems are boolean in nature: either a system satisfies its specification or it doesn't. In case a system does not meet its specification, one expects a good verification framework to provide a counter-example, that is, a reason why the system is not correct. Orna and her co-authors have generalized formal specification and verification methods to address the quality of systems. In her talk, Orna introduced the linear temporal logic LTL[F], where F is a set arbitrary functions over the interval $[0, 1]$. Formulae in LTL[F] are interpreted over computations consisting of sequences of atomic propositions. The satisfaction value of an LTL[F] formula is a number between 0 and 1 that describes how well a computation satisfies a formula. The logic generalizes traditional LTL with the functions in F; examples of functions that might be in F are the maximum or minimum between the satisfaction values of subformulae (these are the quantitative counterparts of boolean OR and AND, respectively), their product and their average. In her talk, Orna showed us how to generalize classic decision problems in formal methods, such as satisfiability, model checking and synthesis, to search and optimization problems in the quantitative setting. This is achieved by means of an extension of the automata-theoretic approach to LTL to the setting of LTL[F].

Before the conference dinner, Jon Kleinberg delivered a special EATCS lecture to celebrate the 40th ICALP. Jon gave an inspiring and very articulate talk entitled *Algorithms, Networks, and Social Phenomena*. (The slides for the talk are available at `http://www.icalp2013.lu.lv/`.) Jon's presentation discussed the development of computational models for systems involving social networks and large human audiences. In particular, Jon focused on how information spreads through such systems, and the ways in which this spread is affected by the underlying network structure. Jon said a few times that, despite having so much data at our disposal, we still do not understand human behaviour. However, in my humble opinion, the work by Jon and his coworkers is shedding some light on some aspects of our behaviour.

During his presentation, Jon gave us a very interesting view of the hot-spots in the history of ICALP, as seen from title word frequencies, which you can find in Table 7. Some bibliographic notes on the method for constructing the time-line, courtesy of Jon, may be of interest to the readers of this report. The time-line was built from an algorithm in a paper presented at KDD 2002 (see `http://www.cs.cornell.edu/home/kleinber/bhs.pdf`) and, in particular, using the methodology described in Section 4 of that article, where time-lines for other conferences are presented. (For the analysis of title word frequencies for ICALP, Jon removed the phrases "extended abstract" and "preliminary version", which had a lot of temporal variation, and also a small number of very common function words.) In order to interpret the results presented in Table 7, it is also important to

note that, since the algorithm is looking for words with large variation over time, a number of terms from ICALP titles that had high frequency throughout the history of the conference didn't produce enough temporal variation to register as a hot-spot, and therefore are not present in the table. These terms include, for example, "complexity", "algorithms", "graphs", "automata", and a number of other words such as "model" and "logic".

Peter Widmayer delivered the last invited talk on Friday, 12 July. His presentation was entitled *To Be Uncertain Is Uncomfortable, But to Be Certain Is Ridiculous*, and was accessible and well paced. The starting point of Peter's talk was a "Socratic dialogue" between a statistical physicist and himself. Traditionally, in combinatorial optimization one assumes that an input instance is given with absolute certainty. The goal is then to find an optimum solution for the given instance. In contrast, as the statistical physicist would argue, in reality input data are uncertain, noisy and inaccurate. As a consequence, an optimum solution to a combinatorial optimization problem might not be meaningful in practice. (For example, the shortest path to our work place we computed yesterday evening might not be usable this morning because of changed traffic conditions.) Peter advocated the development of algorithms that find "meaningful" solutions in the presence of uncertain inputs, proposed an approach towards reaching this goal and argued that it leads to good solutions in the real world.

Videos of the invited talks (with the exception of Kleinberg's talk, which, as far as I know, was not recorded) will be available in due course on the EATCS YouTube channel at

`https://www.youtube.com/channel/UChc3QOHDEbDdPRErx1uSl6A`.

The best paper award session at ICALP 2013 was held on Monday, 8 July, before the welcome reception, which included excellent, and plentiful, finger food and wine. Mark Bun, John Fearnley and Dominik Pajak delivered very good presentations of the award-winning papers, which were:

- Track A: Mark Bun and Justin Thaler. *Dual Lower Bounds for Approximate Degree and Markov-Bernstein Inequalities*.
- Track B: John Fearnley and Marcin Jurdzinski. *Reachability in Two-Clock Timed Automata is PSPACE-complete*.
- Track C: Dariusz Dereniowski, Yann Disser, Adrian Kosowski, Dominik Pajak and Przemyslaw Uznanski. *Fast Collaborative Graph Exploration*.

I hope that you will check out their papers.

The EATCS Award and the Presburger Award were delivered on Tuesday, 9 July, in a joint ceremony that also included the award of honorary doctorates to

Word	Hot-Spot Interval
grammars	1977 — 1982
languages	1977 — 1982
data	1977 — 1983
some	1977 — 1986
equivalence	1977 — 1988
types	1977 — 1990
sets	1978 — 1986
nondeterminism	1979 — 1982
semantics	1979 — 1992
language	1979 — 1994
processes	1983 — 1998
parallel	1986 — 1996
systems	1988 — 1994
equivalences	1990 — 1999
nets	1992 — 2000
scheduling	2001 — 2010
succinct	2003 — 2011
quantum	2004 —
security	2004 — 2008
cover	2005 —
stochastic	2005 —
encryption	2005 — 2008
games	2006 —
network	2008 —
via	2008 —
online	2010 —

Table 7: ICALP, as seen from title word frequencies (courtesy of Jon Kleinberg)

Josef Gruska and Juris Hartmanis. The master of ceremony for the award session was Paul Spirakis. First, Martin Dyer received the EATCS Award from Friedhelm Meyer auf der Heide and delivered a presentation entitled *Counting ain't easy*. Martin started by citing the following fragment of a rhyme from the Winnie the Pooh books:

> Cheers for Pooh!
> (For who?)
> For Pooh -
> (Why, what did he do?)
> I thought you knew

He then proceeded to give an accessible historical overview of what he did do, covering essentially all the work mentioned in the laudatio for the award.

Next, Antonin Kucera presented the Presburger Award 2013 to Erik Demaine. Unfortunately, Erik could not be with us in Riga, but we had a virtual presentation of the award to him via Skype. Moreover, Erik produced an excellent video of a presentation that we could play and enjoy at the conference. Rather than attempting to summarize Erik's inspirational talk, I will simply limit myself to point you to `https://www.youtube.com/watch?v=ROYIVVZ5gvE`.

The honorary doctorates were an excellent addition to the standard session devoted to the EATCS Awards. Gruska and Hartmanis delivered lucid and inspirational presentations. It was truly awesome to see Juris Hartmanis deliver an off-the-cuff speech on how he left Latvia and ended up at Cornell as chair of the newly-founded Computer Science department. At 85, he is still very articulate and an inspirational figure.

The general assembly of the EATCS was held on Wednesday, 10 July, immediately after the excursion and was well attended. The general assembly decided that ICALP 2015 will be held in Kyoto, Japan, and will be co-located with LICS 2015. Kazuo Iwama is the ICALP 2015 general chair. This will be the first ever ICALP outside Europe. During the general assembly, Orna Kupferman gave a thought-provoking talk on *The Gender Challenge in TCS*. It really got the audience thinking about this important matter.

During the general assembly, the best student paper awards were presented to the authors of the following papers:

- Track A: Radu Curticapean. *Counting matchings of size k is #W[1]-hard*.
- Track B: Nicolas Basset. *A maximal entropy stochastic process for a timed automaton*.

None of the accepted papers for Track C was a student paper.

In case you are interested in having a look, the slides I used for the EATCS general assembly are available at

http://www.ru.is/~luca/EATCS/GA2013.pdf .

I hope that this conference report gives you a glimpse of the rich scientific and social programme that made the 40th ICALP in Riga an excellent conference. Everyone involved in the organization of ICALP 2013 deserves the warmest thanks from the TCS community.

ICALP 2014 will be held from Tuesday, 8 July 2014, to Friday, 11 July 2014, on the premises of the IT University in Copenhagen, Denmark, after SEA 2014 (29 June–1 July 2014) and SWAT 2014 (2–4 July 2013). It will be a four-day ICALP and the general chair for the conference is Thore Husfeldt. For jazz lovers, let me also remark that ICALP 2014 will overlap with the Copenhagen Jazz Festival, which will be held in the period 4–13 July 2014.

I hope that you will make plans to submit your best work to ICALP 2014 and to go to Copenhagen for the conference. I heartily recommend it and look forward to seeing you there.

Report from the Italian Chapter

R. De Nicola (Universita di Firenze)

ICTCS 2013

The 14th Italian Conference on Theoretical Computer Science (ICTCS 2013) was held at the University of Palermo, Italy, from the 9th to the 11th of September 2013.

ICTCS is the conference of the Italian Chapter of the European Association for Theoretical Computer Science and, besides being a forum of exchange of ideas, it provides the ideal environment where junior researchers and Ph.D. students meet senior researchers. ICTCS 2013 is also open to researchers from outside Italy, who are welcome to submit papers and attend the Conference.

During the General Assembly, Stefano Bistarelli, the Organizing Committee Chair of ICTCS 2014 gave an entertaining report on the location Perugia and the venue at the University of Perugia. The whole team in Perugia is looking forward to being a good host for ICTCS 2014.

IC-EATCS awards

As usual, during the conference, the President of the Italian Chapter gave up some awards.

Indeed, every year the Italian Chapter assigns an award for the two best Italian PhD theses in Theoretical Computer Science. For 2013, the Selection Committee, composed by professors Andrea Clementi (Univ. of Rome tor Vergata), Mimmo Parente (Univ. of Salerno) and Elena Zucca (Univ. of Genova) selected as recipients of the award

Jacopo Mauro (whose thesis is entitled *Constraints Meet Concurrency*)

and

Alessandra Scafuro (whose thesis is entitled *Secure Computation Under Network and Physical Attacks*).

Furthermore, for the first time, this year an award for the best young Italian researcher in Theoretical Computer Science has been established. The Selection Committee, composed by professors Mariangiola Dezani (Univ. of Turin), Luisa Gargano (Univ. of Salerno) and Geppino Pucci (Univ. of Padua) selected as recipient of this new award

Luca Moscardelli (Univ. of Chieti-Pescara).

These three young researchers gave a talk to the ICTCS attendees describing their research area and summarizing their own results.

Appendix

Talks have been given in the following order:

- G.F. Italiano (Invited Talk): Strong bridges and strong articulation points of directed graphs.
- V. Lonati, D. Mandrioli, F. Panella, M. Pradella: Free grammars and languages.
- M. B. Abu Ayyash, E. Rodaro: A language theoretical approach to some problems on inverse semigroups.
- N. Beerenwinkel, S. Beretta, P. Bonizzoni, R. Dondi, Y. Pirola: Covering pairs in directed acyclic graphs.
- S. Bistarelli, S. Foley, F. Santini, F. Vicino: An introduction to consistently merging trust-networks with bipolar preferences.
- S. Brunetti, G. Cordasco, L. Gargano, E. Lodi, W. Quattrociocchi: Minimum weight multicolor dynamos.
- F. Mignosi, A. Langiu, M. Crochemore, C. Iliopoulos: On the multiple common substring problem.
- M. H. Ter Beek, F. Gadducci, F. Santini: Validating reconfigurations of Reo circuits.
- A. Frosini, S. Rinaldi, D. Battaglino, L. Vuillon, S. Socci, M. Bouvel: Pattern-avoiding polyominoes.
- E. Rodaro, R. Reis: Reset regular decomposition complexity of regular ideal languages.
- A. Bernini, S. Bilotta, R. Pinzani, V. Vajnovszki: Two Gray codes for q-ary k-generalized Fibonacci strings.
- V. V. Gusev, M. Maslennikova, E. V. Pribavkina, E. Rodaro: Language theoretic approach to synchronizing automata.
- E. Omodeo, A. Policriti, A. I. Tomescu: Bridging syllogistics with combinatorics.
- L. Aceto, D. Della Monica, A. Ingólfsdóttir, A. Montanari, G. Sciavicco: A complete classification of the expressiveness of interval logics of Allen's relations over dense linear orders.
- U. de'Liguoro: The approximation theorem for the Lambda-mu calculus.
- D. Ancona, P. Giannini, E. Zucca: Incremental rebinding.
- L. Moscardelli (Young Researcher in Theoretical Computer Science Award 2013): Convergence issues in congestion games.
- C. Barton, C. S. Iliopoulos, S. Pissis: Circular string matching revisited.

– A. Langiu, A. Alatabbi, M. S. Rahman, C. S. Iliopoulos: Computing the longest common abelian factor.
– M. M. Bersani, M. Rossi, P. San Pietro: Deciding in practice the satisfiability of continuous-time metric temporal logic.
– R. Gentilini: A SAT encoding for solving games with energy objectives.
– I. Bonacina, N. Galesi: Space complexity in algebraic proof systems.
– M. Bartoletti, P. Di Giamberardino, R. Zunino: Towards a linear contract logic.
– L. Bernardinello, C. Ferigato: On the composition of regional structures and their logics.
– M. Lenisa, F. Honsell, R. Redamalla: Polarized multigames.
– J. Mauro (Best Ph.D. Thesis in Theoretical Computer Science Award 2013): Constraints meet concurrency.
– A. Scafuro (Best Ph.D. Thesis in Theoretical Computer Science Award 2013): Secure computation under network and physical attacks.
– P. Degano (Invited Talk): A formal model of context-oriented programming.
– P. Cenciarelli, D. Gorla, I. Salvo: On two different forms of inefficiency in network design.
– L. Bernardinello, C. Ferigato, S. Haar, L. Pomello: Dynamically closed sets in occurrence nets.
– M. H. Ter Beek, A. Lluch Lafuente, M. Petrocchi: Combining declarative and procedural views in the feature-oriented specification and analysis of product families.
– E. Rodaro, R. Reis: The language of initially connected deterministic finite automata.
– M. Anselmo, D. Giammarresi, M. Madonia: Strong prefix codes of pictures.
– S. Crespi Reghizzi, P. San Pietro: Commutative consensual counter languages.
– J. Clément, L. Giambruno: On the number of prefix and border tables.

The Italian Chapter

Chair: Eugenio Moggi
V.Chair: Clelia De Felice
Secretary: Rocco De Nicola
email: denicola@dsi.unifi.it
URL: http://www.unifi.it/eatcs

Report on Calco 2013

5th Conference on Algebra and Coalgebra in Computer Science

3–6 September 2013, Warsaw, Poland

Alexandra Silva Pawel Sobocinski

Calco 2013, the 5^{th} International conference on Algebra and Coalgebra in Computer Science, took place in Warsaw, Poland, 2^{nd} – 6^{th} September 2013, in the beautiful setting of the Old Library building on the historical campus of the University of Warsaw.

The first day was dedicated to the Calco Early Ideas workshop, devoted to presentation of work in progress, with a particular emphasis on talks by PhD students and young researchers. This workshop is the follower of Calco-jnr.

The other workshop affiliated with Calco, Calco-Tools, is dedicated to work on tools based on algebraic and/or coalgebraic principles. This year it took place inlined with the main conference which led to discussion during the business meeting about the possibility of making Calco-Tools a track of the main conference instead of a separate workshop.

There were four invited talks: **Andrej Bauer** (University of Lubljana, Slovenia), **Damien Pous** (CNRS, ENS-Lyon, France), **Mikołaj Bojańczyk** (Warsaw University, Poland) and **Neil Ghani** (University of Strathclyde, United Kingdom).

Andrej Bauer opened the conference proper with an invited talk about how effects can be treated in a strikingly elegant way within a functional setting. The ideas of Plotkin and Power, who used Lawvere theories instead of monads, have led Andrej and his collaborators to design a programming language where effects can sometimes be combined in an easier and more transparent way, as was illustrated with a number of compelling examples.

The second morning session concerned coalgebrac logics and consisted of four stimulating talks. The afternoon featured the first session of Calco Tools, including a couple of talks with a strong Haskell flavour, and closed with a session on Categorical Structures. In the evening we were treated to a reception at the Tyszkiewicz-Potocki Palace on campus, featuring an official welcome by Andrzej Tarlecki.

The second day started with Damien Pous' invited talk on *Coalgebraic up-to techniques*. Damien talked about several enhancements to the bisimulation proof method that enable shorter proofs. He illustrated the application of the new techniques with basic examples from automata theory which made the talk very accessible (but not less impressive!).

The rest of the morning was filled with one Calco-Tools session, consisting of four talks. Peter Ölveczky gave two talks on Maude-related tools and presented a very compelling (and somewhat morbid) case study on the Finnish Sauna Championships (and the death of the world champion). The afternoon included a session on the semantics of processes, featuring an excellent presentation on dialgebraic modelling by Vincenzo Ciancia, which was awarded the inaugural Best Presentation award—more on this later! The final session concerned logic programming.

The third day was kicked off by a fantastic talk by Mikołaj Bojańczyk on *Automata and Algebras for Infinite Words and Trees*. He reviewed classical work on automata theory and then briefly explained two open problems which he thinks might have a coalgebraic solution. The morning continued with a session on *Behaviour Modelling*, which included 4 talks. Let us highlight Pierre Lescanne's talk on how coalgebra can be applied in economics!

In the afternoon, there was the social event: a walking tour through the historic centre of Warsaw. We visited several buildings where famous mathematicians lived and had the chance to see the beautiful library of the university with an extensive garden where students hang out (whenever the weather allows!).

The social dinner in the evening was at *Kuźnia Smaku*. Before dessert Jan Rutten and Stefan Milius gave a speech which led to one of the highlights of the evening. Filippo Bonchi and Fabio Zanasi were awarded the Best Paper Award for their paper *Saturated Semantics for Coalgebraic Logic Programming*. Congratulations!

In addition to the Best Paper Award, included also a Best Presentation Award. The audience had access to an online voting system where, in addition to a score, they could leave comments to the participants with suggestions for improvements for the future. We think this is a neat idea and might help increase the quality of presentations in the future, therefore making the conference even more enjoyable to attend! The slides of this year's talks are online.

The local organizers did stupendous work and deserve our warmest thanks for making a successful event!

The next will take place in 2015, in the picturesque city of Nijmegen, in The Netherlands. It will be co-located with MFPS, the Mathematical Foundations of Programming Semantics conference.

Report on Highlights 2013

Highlights of Logic, Games and Automata

Szymon Toruńczyk
Institute of Informatics, University of Warsaw

The Highlights conference kicked off with its first, very successful meeting in Paris. The inaugural event took place in Université Paris Diderot – near the Bibliothéque François Mitterand – and its local organisers were Dietmar Berwanger and Thomas Colcombet. It lasted only for three days, from the 19th to the 21st of September, from Thursday to Saturday.

According to Mikołaj Bojańczyk – one of the main initiators of the new conference – the idea was to continue the tradition of the GAMES workshop, and to create a proceedings-free venue, at which the most important results obtained in the last few years in the areas of logic, automata and games are presented.

Statistics prove that this aim was accomplished. A total of 95 submissions for contributed talks were received – all of them accepted – representing a total of 192 coauthors. Overall, there were 218 registered participants, filling the auditorium to its limits. By nationalities: 82 from France, 51 from Germany, 22 from Poland, 16 from the UK, 12 from Belgium, 9 from Italy, 8 from the Czech Republic, 6 from the USA and 12 from other countries.

The meeting was preceded by a tutorial day, consisting of two three-hour tutorials. Jérôme Leroux talked about the reachability problem for Petri nets, and presented a large part of his new, elegant geometric proof of the classical result that the reachability problem for Petri nets is decidable. Martin Grohe gave a tutorial on Logic for Algorithms, and gave a broad overview of algorithmic metatheorems, and announced a new, fantastic result culminating this line of research, concerning the evaluation of first-order logic on nowhere-dense classes of graphs.

The invited talks took place in the mornings, and also after lunch on Friday; they lasted 50 minutes each. In the first invited talk, Patrice Godefroid explained how SMT solvers are used in modern software verification, and how these methods lead to efficient elimination of bugs in the software produced by Microsoft. On Friday morning, Erich Grädel gave an excellent survey talk on logics for polynomial time, encouraging the audience to take part in the quest. Friday afternoon saw Marco Scarsini present a new notion of equilibria in countable games. On Saturday morning Moshe Vardi gave a very good overview talk on the rise and fall of Linear Temporal Logic.

For the first two days of the conference, the talks were delivered in a series of five sessions which lasted from 9am to 6.30pm. The third day's contributed talks were split into two parallel threads. With such a tight program, the contributed

talks were scheduled to last only about 10 minutes each on the first two days, and 15 minutes on the third day.

My personal opinion is that such short talks were beneficial for two reasons. Firstly, they allowed allowed the speakers to maintain the attention of a big part of the audience. Secondly, they forced the speakers to focus on only one or two key aspects of their entire work, resulting in a higher-than-usual average quality of the talks. One drawback of the tight schedule was the short length of some of the breaks.

To the surprise of many, despite of the tight schedule and full auditorium, the organization was extremely efficient. Several innovations helped improve the quality of the sessions. The speakers were asked to submit their presentations by Tuesday, before the start of the conference. This allowed the organizers to smoothly handle the slides during the sessions. It also had the benefit that members of the audience were not distracted by the creation of their own slides during the talks. A small innovation which helped the session chairs were specially prepared cards used to display the remaining time to the speaker. Lastly, an electronic feedback system was available to those willing to participate, allowing the speakers to gather anonymous comments concerning their own talk.

The participation in the conference was free of fees, thanks to the generosity of the invited speakers, who agreed to cover their travel costs from other funds. The organizers provided coffee and snacks for two breaks per day and a lunch on Saturday. Some participants complained on undercaffeination.

There was no official social event, in order to cut down the costs. However, Nathanaël Fijalkow has volunteered to organize an unofficial event – a contributory picnic at the banks of the Seine river, typical to the Parisians. According to unconfirmed information, up to seventy people indulged themselves in the pleasures of French wine, cheese and baguette. All survived.

After the great success of the first meeting of the new Highlights conference, the next event is planned to take place also in Paris, in 2014.

Inside the Simons Institute

Dominik Scheder

October 9, 2013

Let f be a boolean function. Define the *circuit complexity* of ... No, wait! Let's back up a little bit. Let me tell you how we ended up here. We are in Berkeley, California, in a circular-shaped building on the university campus. It is week five in the life of the *Simons Institute for the Theory of Computation*. This institute is named after James Harris Simons, an American mathematician who started a second career as a hedge fund manager, and, once this had borne sufficient fruit, a third career as a philanthropist. His Simons Foundation endowed the institute with a grant of $60 million.

The goal of the institute is to promote research on the foundations of computer science and on other scientific areas where the paradigm of computation promises to be fruitful, for example biology or economics. Right now the institute houses its first two projects: *Theoretical Foundations of Big Data Analysis* and *Real Analysis in Computer Science*.

When I arrived a couple of weeks ago, the construction workers were still there, giving the final touch to the building. This did not prevent the program from hitting the ground running, with a one-week workshop on *Real Analysis in Testing, Learning and Inapproximability*. Week three saw a *Real Analysis Boot camp*, in which Johan Håstad, Adam Klivans, and Krzysztof Oleszkiewicz made all participants familiar with the methods and problems of the field. In contrast to usual boot camps, Johan Håstad did not shout at the recruits and order them to do push-ups, but rewarded correct answers and good questions with excellent Swiss chocolate (he changed to healthier apples towards the end).

Everything at the institute is designed to encourage people to collaborate and not toil in isolation: There is lots of common space, with whiteboards and sofas. Pleasant weather and blackboards encourage working outside. During workshops, coffee breaks are frequent and long enough to go beyond chit-chat and talk research—with a cup and a cookie in your hands. And if you wonder whether the research problem that just came up has already been solved, no need to search the web or the library: Just ask the expert in the office next door.

At this point, the reader might wonder what the scientific content of *Real Analysis in Computer Science* is. Let me briefly summarize three of the many talks the

institute has seen in its first five weeks. Let us start with week one, when the institute held its first workshop, and let f be a boolean function.

Li-Yang Tan at the Week One Workshop

We are given a boolean function $f : \{0,1\}^n \to \{0,1\}$. Can we write f as a DNF formula? Surely we can. How many terms might we need? Never more than 2^n terms, and a minute or two of thought shows that 2^{n-1} are always enough. Neither is it too difficult to see that the Parity function $f(x_1, \ldots, x_n) = x_1 \oplus \cdots \oplus x_n$ requires a DNF formula of size 2^{n-1}. So this bound is exactly tight. Very nice! For once in circuit complexity, we know the answer exactly.

What about approximating f? Suppose we want to find a DNF formula D such that f and D agree on all but $\epsilon 2^n$ inputs. We say D is an ϵ-approximation of f. Can we make D smaller than 2^{n-1}? Sure! By converting f into a DNF formula D and deleting some terms, we can ϵ-approximate f with a DNF formula of size $(1-\epsilon)2^{n-1}$. The surprising result of Li-Yang Tan and Eric Blais (reference) is that one can do *much* better:

Theorem 1 (Blais and Tan [2])**.** *Every boolean function $f : \{0,1\}^n \to \{0,1\}$ can be ϵ-approximated by a DNF of size $O(2^n/\log n)$. The $O(\cdot)$ hides a constant factor which depends on ϵ.*

Besides its number of terms, a very natural complexity measure of a DNF formula is its *width*, that is, the maximum size of its terms. Suppose we ϵ-approximate f by a DNF formula D with s terms, where $s \in O(2^n/\log n)$ terms. Again, it is not too difficult to see that one can delete all terms of width larger than $\log_2(s/\epsilon)$ and still get a 2ϵ-approximation. In this case, the width of the resulting DNF formula is at most $n - \log\log n - \log\epsilon$. This is smaller than n, but can we do better?

Theorem 2 (Blais and Tan [2])**.** *Every boolean function can be ϵ-approximated by a DNF formula of width at most $(1-c_\epsilon)n$, where $c_\epsilon > 0$ and depends only on ϵ.*

To see how surprising this theorem is, think of choosing f randomly. That is, for every $x \in \{0,1\}^n$, toss a coin to decide whether $f(x)$ should be 0 or 1. What is a term of a DNF formula? It is a conjunction of literals, and its satisfying assignments correspond to a subcube of $\{0,1\}^n$. If the term has width $(1-c)n$, the subcube contains 2^{cn} points. Since $c > 0$ is a constant, it contains an exponential number of points. Thus, the random function f will be very balanced on every subcube. The theorem says that we can still select subcubes that intersect in a very special way, such that f is very unbalanced on the union of these subcubes: Almost always 1 on that union, and almost always 0 outside.

Aviad Rubinstein in the Thursday Seminar

Every Thursday morning, we meet for a seminar of one long and four short talks. Short really means short: The speaker has eight minutes to get the audience excited about his or her result. I'll try to get the reader excited about the result by Aviad Rubinstein I heard this morning.

Let $X_1, \ldots, X_n$ be unbiased coins, i.e., independent variables taking on values in $\{-1, 1\}$ with equal probability. Let $Z = a_0 + a_1X_1 + \cdots + a_nX_n$ for some real numbers $a_0, a_1, \ldots, a_n$. When does Z look like a Boolean function $f : \{-1, 1\}^n \to \{-1, 1\}$? For instance, let $Z = \frac{X_1+\cdots+X_n}{\sqrt{n}}$. Note that Z falls into the interval $\left[-\frac{1}{2}, \frac{1}{2}\right]$ with constant probability, and thus is not very close to any boolean function f. In a second example, let $Z = \frac{X_1+\cdots+X_n}{n} + X_{n+1}$. Most of the time the first summand is really small, and thus Z is usually close to X_{n+1}, which is itself a boolean function. The FKN Theorem (Friedgut, Kalai, and Naor [3]) states that if Z is close to a boolean function, it is close to some X_i:

Theorem 3 ([3]). *Let $Z = a_0 + a_1X_1 + \cdots + a_nX_n$. If $f : \{-1, 1\}^n \to \{-1, 1\}$ satisfies*

$$\mathbb{E}\left[(Z - f(X_1, \ldots, X_n)^2\right] \leq \epsilon\,,$$

then there exists some $1 \leq i \leq n$ such that $\mathbb{E}\left[(Z - X_i)^2\right] \leq \epsilon$.

Aviad Rubinstein and Muli Safra [7] proved a generalization of this theorem: The X_i need not be unbiased coins. In fact, they need not be coins. All that is required is that they be "well-behaved" independent random variables. For example, each X_i could be itself a boolean function f_i, over a set of boolean variables that is disjoint from that of the other functions f_j. A similar result was discovered independently by Jendrej, Oleszkiewicz, and Wojtaszczyk [5].

Johan Håstad at the Boot Camp on Real Analysis in Computer Science

In the first four lectures of his mini-course Johan Håstad explained his classical result that Max-3-SAT is NP-hard to approximate within a factor of $\frac{7}{8} + \epsilon$ [4]. On top of giving this beautiful proof in full detail, he treated the audience with insider stories on how he came up with this proof, which obstacles he faced and how he overcame them.

In the fifth lecture he presented a very recent result about a different notion of approximation. Suppose F is a CNF formula, and we are promised that there exists a "super-assignment" that in every clause satisfies at least half of all literals. The goal is to find any satisfying assignment. A simple random walk algorithm by Papadimitriou [6] solves this in expected polynomial time. What if we relax the

promise marginally and are only promised that the super-assignment satisfies 0.49 of the literals in every clause. On such instances, Papadimitriou's algorithm might have exponential running time. Is this problem "Super-SAT" hard in general?

The answer is yes, it is hard, as shown recently by Per Austrin, Venkatesan Guruswami, and Johan Håstad [1]. This result was the subject of Johan Håstad's fifth lecture: Given a CNF formula in which every clause consists of exactly $2k+1$ literals, and given the promise that some assignment satisfies at least k literals per clause, it is still NP-hard to find a satisfying assignment.

Their proof uses some established machinery in an unusual way. They start with an instance of the Label Cover problem—just as most inapproximability proofs, including Håstad's hardness proof for approximating Max-3-SAT. Label Cover is a certain coloring problem on bipartite graphs, which is known to be extremely hard to approximate. Austrin, Guruswami, and Håstad use Long Codes to encode the colors $\{1,\ldots,r\}$ of the Label Cover problem into binary. So far this follows well-established paths. However, their construction of a CNF gadget from the long codes is very different from the Max-3-SAT case. After all, the goal of the Super-SAT problem is not to maximize a sum (as in the case of Max-3-SAT, where we want to maximize the sum of the weight of satisfied clauses), but to minimize a maximum: We want to minimize the maximum number of unsatisfied literals a clause contains. In fact, that maximum should be 0. This instance of "bottleneck optimization" requires quite different an analysis. In fact, their reduction gadget can be analyzed combinatorially without Fourier analytic tools.

References

[1] Per Austrin, Venkatesan Guruswami, and Johan Håstad. work in progress.

[2] Eric Blais and Li-Yang Tan. Approximating boolean functions with depth-2 circuits. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:51, 2013.

[3] Ehud Friedgut, Gil Kalai, and Assaf Naor. Boolean functions whose fourier transform is concentrated on the first two levels. *Adv. in Appl. Math*, 29, 2002.

[4] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[5] Jacek Jendrej, Krzysztof Oleszkiewicz, and Jakub O. Wojtaszczyk. submitted.

[6] Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *Proceedings of the 32nd annual symposium on Foundations of computer science*, SFCS '91, pages 163–169, Washington, DC, USA, 1991. IEEE Computer Society.

[7] Aviad Rubinstein. Boolean functions whose fourier transform is concentrated on pair-wise disjoint subsets of the inputs. Master's thesis, Tel-Aviv University, 2012.

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 Nerode Award (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."

(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

(2) THE BULLETIN OF THE EATCS

Three issues of the Bulletin are published annually, in February, June and October respectively. The Bulletin is a medium for *rapid* publication and wide distribution of material such as:

- EATCS matters;
- Technical contributions;
- Columns;
- Surveys and tutorials;
- Reports on conferences;
- Information about the current ICALP;
- Reports on computer science departments and institutes;
- Open problems and solutions;
- Abstracts of Ph.D. theses;
- Entertainments and pictures related to computer science.

Contributions to any of the above areas are solicited, in electronic form only according to formats, deadlines and submissions procedures illustrated at `http://www.eatcs.org/bulletin`. Questions and proposals can be addressed to the Editor by email at `bulletin@eatcs.org`.

(3) OTHER PUBLICATIONS

EATCS has played a major role in establishing what today are some of the most prestigious publication within theoretical computer science.

These include the *EATCS Texts* and the *EATCS Monographs* published by Springer-Verlag and launched during ICALP in 1984. The Springer series include *monographs* covering all areas of theoretical computer science, and aimed at the research community and graduate students, as well as *texts* intended mostly for the graduate level, where an undergraduate background in computer science is typically assumed.

Updated information about the series can be obtained from the publisher.

The editors of the EATCS Monographs and Texts are now M. Henzinger (Wien), J. Hromkovic (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 G. Ausiello (Rome) and D. Sannella (Edinburgh).

ADDITIONAL EATCS INFORMATION

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

Prof. Dr. Luca Aceto,
School of Computer Science
Reykjavik University
Menntavegur 1 IS-101 Reykjavik, Iceland
Email: `president@eatcs.org`

EATCS MEMBERSHIP

DUES

The dues are € 30 for a period of one year (two years for students). 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 € 25 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 € 25 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, a subscription form can be downloaded from `www.eatcs.org` to be filled and sent together with the annual dues (or a multiple thereof, if membership for multiple years is required) to the **Treasurer** of EATCS:

Prof. Dr. Dirk Janssens,
University of Antwerp, Dept. of Math. and Computer Science
Middelheimlaan 1, B-2020 Antwerpen, Belgium
Email: `treasurer@eatcs.org`, *Tel: +32 3 2653904, Fax: +32 3 2653777*

The dues can be paid (in order of preference) by VISA or EUROCARD/MASTERCARD credit card, by cheques, or convertible currency cash. Transfers of larger amounts may be made via the following bank account. Please, add € 5 per transfer to cover bank charges, and send the necessary information (reason for the payment, name and address) to the treasurer.

Fortis Bank, Jules Moretuslei 229, B-2610 Wilrijk, Belgium
Account number: 220–0596350–30–01130
IBAN code: BE 15 2200 5963 5030, SWIFT code: GEBABE BB 18A

EATCS MATTERS

EATCS COLUMNS

Contributions by EATCS Award Recipients

News and Conference Reports