ISSN 0252–9742

Bulletin

of the

European Association for

Theoretical Computer Science

EATCS

Number 116 June 2015

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

Lars Arge	Denmark	Elvira Mayordomo	Spain
Jos Baeten	The Netherlands	Luke Ong	UK
Paul Beame	USA	Catuscia Palamidessi	France
Mikolaj Bojanczyk	Poland	David Peleg	Israel
Josep Díaz	Spain	Giuseppe Persiano	Italy
Zoltán Ésik	Hungary	Alberto Policriti	Italy
Fedor Fomin	Norway	Alberto Marchetti Spaccamela	Italy
Pierre Fraigniaud	France	Vladimiro Sassone	UK
Leslie Ann Goldberg	UK	Roger Wattenhofer	Switzerland
Monika Henzinger	Austria	Thomas Wilke	Germany
Christos Kaklamanis	Greece	Peter Widmayer	Switzerland
Juhani Karhumaki	Finland	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	Italy
	Efi Chita	Greece

EATCS Council Members

email addresses

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.

The EATCS home page is `http://www.eatcs.org`

Table of Contents

EATCS MATTERS

Letter from the President 3
Letter from the Bulletin Editor 11
Memories of Jiří Matoušek in Japan 15

EATCS COLUMNS

The Algorithmics Column, *by G.J. Woeginger*
Query-Competitive Algorithms for Computing with Uncertainty, *by T. Erlebach, M. Hoffmann* 21

The Computational Complexity Column, *by V. Arvind*
Combinatorial Game Complexity: An Introduction with Poset Games, *by S.A. Fenner, J. Rogers* 41

The Distributed Computing Column, *by P. Fatourou*
Maurice Herlihy's 60th Birthday Celebration, *by P. Fatourou* 77
A Quarter-Century of Wait-Free Synchronization, *by V. Hadzilacos* 79
Hardware Trends: Challenges and Opportunities in Distributed Computing, *by T. Harris* 91
Transactional Memory Today, *by M. Scott* 99

The Distributed Computing Column, *by S. Schmid*
Fault-tolerant Distributed Systems in Hardware, *by D. Dolev, M. Fugger, C. Lenzen, U. Schmid, A. Steininger* 111

The Education Column, *by J. Hromkovič*
Informatics - New Basic Subject, *by W. Gander* 155

The Formal Language Theory Column, *by G. Pighizzini*
Average Size of Automata Constructions from Regular Expressions, *by S. Broda, A. Machiavelo, N. Moreira, R. Reis* 167

THE LOGIC IN COMPUTER SCIENCE COLUMN, *by Y. Gurevich*

SELECTED PAPERS FROM THE 1ST WORKSHOP "LOGIC, LANGUAGE, AND INFORMATION", *by G. Sciavicco, A. Burrieza* 193

DISTRIBUTED EXPLICIT KNOWLEDGE AND COLLECTIVE AWARENESS, *by A. Burrieza, C. Fernández-Fernández* 195

A LOGICAL APPROACH FOR DIRECT-OPTIMAL BASIS OF IMPLICATIONS, *by E. Rodríguez-Lorenzo, P. Cordero, M. Enciso, A. Mora* 204

ABDUCTIVE REASONING IN DYNAMIC EPISTEMIC LOGIC - GENERATION AND SELECTION OF HYPOTHESIS,, *by I.D. Arroniz* 212

LCC-PROGRAM TRANSFORMERS THROUGH BRZOZOWSKI'S EQUATIONS, *by E. Sarrión-Morillo* 220

MEREOLOGY AND TEMPORAL STRUCTURES, *by P. González Nuñez* 230

NEWS AND CONFERENCE REPORTS

REPORT FROM THE JAPANESE CHAPTER, *by R. Uehara* 241

REFLECTIONS ON INFLUENTIAL SCIENTISTS AND IDEAS

GEORGE DANTZIG: FATHER OF THE SIMPLEX METHOD, *by D. Avis* 249

CONTRIBUTIONS BY EATCS AWARD RECIPIENTS

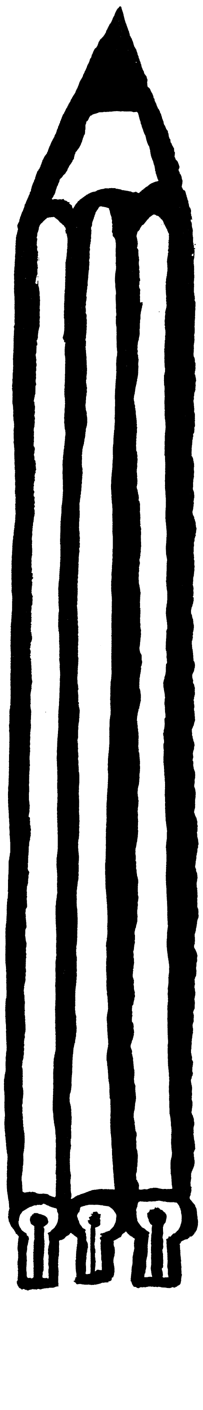
SAMPLING FROM DISCRETE DISTRIBUTIONS AND COMPUTING FRECHET DISTANCES, *by K. Bringmann* 257

BOOK INTRODUCTION BY THE AUTHORS

SEMANTICS OF PROBABILISTIC PROCESSES AN OPERATIONAL APPROACH, *by Y. Deng* 269

EATCS LEAFLET 279

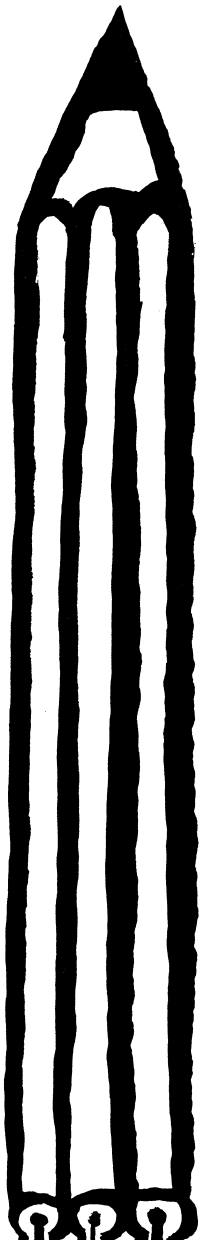
EATCS Matters



Dear colleagues,

As you know, the EATCS is about to hold its first ever ICALP conference outside Europe. The 42nd ICALP will take place in the period 4-10 July 2015 in Kyoto, Japan, and will be co-located with the 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015).

ICALP 2015 had the largest number of submissions in history (507) and the largest ever number of submissions for Track A (327). Moreover, all tracks received many submission of very high quality. The PCs for the three tracks, which were expertly led by Bettina Speckmann (Track A), Naoki Kobayashi (Track B) and Magnús M. Halldórsson (Track C), did an amazing job in selecting an exciting conference programme. So, despite the difficult economic climate, it seems that the decision to hold the conference in Japan in co-location with LICS was a good one to take at this moment in time. Moreover, I feel that it is also a long-overdue sign of recognition of the important role that the Japanese Chapter of the EATCS plays in the life of our association and for theoretical computer science in general, and of the growing status of TCS research in Asia as a whole. The EATCS is firmly rooted in Europe, but is an association with members from all over the world. In fact, I would dearly like to see more members from extra-European countries and I encourage TCS researchers from all over the world to join the EATCS. I hope that, with your help and support, our association can play an



even more important role in the development of TCS across the world.

I am very grateful to Masahito Hasegawa, Kazuo Iwama and their team for the extraordinary work they have done in organizing ICALP/LICS 2015 and for the support they have received from the Japanese community at large. In particular, on behalf of the EATCS, I gratefully acknowledge the financial support of the ERATO Kawarabayashi Large Graph Project (grant holder: Ken-ichi Kawarabayashi), of the project Exploring the Limits of Computation (grant holder: Osamu Watanabe), of the Research Institute for Mathematical Sciences (RIMS), Kyoto University, and of the Tateisi Science and Technology Foundation. Thanks to the sponsorship received by the organizers, the early registration fee for members of the EATCS attending ICALP/LICS 2015 was of 45,000 *JPY (roughly* 320 *Euros), which, by today's standards, is very low for a five-day event.*

Note that ICALP 2015 and LICS 2015 are truly co-located events. The two conferences will take place in the same week and at the same location. There is only one registration fee and conference participants will be able to attend whatever session they fancy at either conference. The seven invited talks, the three invited tutorials and the award session, as well as the coffee and lunch breaks, will be plenary events at which participants in both conferences will congregate and mingle. Moreover, an ICALP Masterclass held by Ryuhei Uehara will celebrate Japan's fascination with puzzles, following on the footsteps of the very

successful masterclass on mathematical puzzles held by Peter Winkler at ICALP 2008 in Reykjavik.

Since ICALP and LICS span the whole spectrum of theoretical computer science, in my admittedly biased opinion ICALP/LICS 2015 will be a veritable "theory festival" with a mouthwatering scientific menu. I invite you to look at www.kurims.kyoto-u.ac.jp/icalp-lics2015/ for details on the conference programme, the invited talks and invited tutorials. As usual, a report on the conference will be published in the October 2015 issue of the Bulletin. I also hope that there will be some coverage of the event on the blogs devoted to theoretical computer science. I will issue a call for guest bloggers on my professional blog myself. Apart from the invited and contributed talks, ICALP 2015 will feature the presentation of the EATCS Award 2015 to Christos Papadimitriou and of the Presburger Award 2015 to Xi Chen. Moreover, during the conference, we will honour the EATCS Fellows class of 2015, who are

- *Artur Czumaj (University of Warwick, United Kingdom) for "contributions to analysis and design of algorithms, especially to understanding the role of randomization in computer science";*

- *Mariangiola Dezani-Ciancaglini (University of Torino, Italy) for "distinguished and seminal achievements in formal methods and foundations of programming languages, introducing or developing new type systems for the*

lambda-calculus as well as for the pi-calculus and related calculi";

- *Thomas A. Henzinger (Institute of Science and Technology Austria) for "fundamental contributions to formal verification and synthesis of computer and biological systems";*
- *Dexter Kozen (Cornell University, USA) for "pioneering and seminal work in fields as diverse as complexity theory, logics of programs, algebra, computer algebra and probabilistic semantics"; and*
- *Moshe Y. Vardi (Rice University, USA) for "fundamental and lasting contributions to the development of logic in computer science and exceptional services to the community of theoretical computer science."*

Last, but not least, the EATCS Distinguished Dissertation Award Committee 2015 has selected the following three theses for the EATCS Distinguished Dissertation Award for 2015:

- *Karl Bringmann, "Sampling from Discrete Distributions and Computing Fréchet Distances",*
- *Michal Skrzypczak, "Descriptive set theoretic methods in automata theory" and*
- *Mary Wootters, "Any errors in this dissertation are probably fixable: topics in probability and error-correcting codes".*

On behalf of the EATCS, I heartily thank the members of the award, dissertation and fellow committees for their work in the selection of this stellar set of award recipients and fellows. It will be a great honour to celebrate the work of these colleagues during ICALP 2015.

If this weren't enough, the programme for ICALP/LICS 2015 will also include a session to celebrate the 40th anniversary of the journal Theoretical Computer Science. As part of that session, Christos Papadimitriou will deliver a presentation on the past, the present and the future of Theoretical Computer Science. I look forward to Christos' presentation.

Even though ICALP 2015 has not taken place yet, we are already busy preparing future editions of the conference. As you may know already, ICALP 2016 will be held in the period 11-15 July 2016 in Rome. Italy. The PC chairs for the conference will be Yuval Rabani (Hebrew University Jerusalem, Israel) for Track A, Davide Sangiorgi (University of Bologna, Italy) for Track B and Michael Mitzenmacher (Harvard University, USA) for Track C. The preliminary call for papers for the conference will be available in Kyoto. Moreover, during the general assembly at ICALP 2015, we will examine a bid to host ICALP 2017 in Warsaw, Poland, and choose the location for ICALP 2017. I am also happy to inform you that our colleagues in Prague are working on a bid for hosting ICALP 2018.

Apart from ICALP, the EATCS is becoming increasingly involved in many initiatives

and uses its financial resources to support young researchers and meritorious activities in Theoretical Computer Science as a whole. We try hard to give back to the community what our members provide via their membership fees. By way of example, I remind you that the second EATCS Young Researcher School will be devoted to Understanding COMPLEXITY and CONCURRENCY through TOPOLOGY of DATA *and is organized by Emanuela Merelli at the University of Camerino, Italy, in the period 13-22 July 2015. Moreover, we have decided to provide some modest financial support to the Conference on Computational Complexity (CCC) and to the Symposium on Computational Geometry (SoCG).*

The above-mentioned activities are just a sample of the increasingly many ones in which the EATCS is involved. In addition, we have stipulated reciprocity agreements with the European Association for Computer Science Logic and the ACM Special Interest Group on Logic and Computation. Later this year, we also hope to be in a position to issue the first call for nominations for a new major prize in TCS in cooperation with ACM Special Interest Group on Logic and Computation, the European Association for Computation, the European Association for Computer Science Logic and the Kurt Gödel Society.

As usual, let me remind you that you are always most welcome to send me your comments, criticisms and suggestions for improving the impact of the EATCS on the theoretical-computer-science community at president@eatcs.org.

I look forward to seeing many of you in Kyoto for ICALP 2015 and to discussing ways of improving the impact of the EATCS within

the theoretical-computer-science community at the general assembly.

Luca Aceto, Reykjavik, Iceland
June 2015

Dear Reader,

My recent days have been a bit hectic because, as you see, we have only a couple of weeks to go before ICAP/LICS 2015 will be starting. In the letter of the president, Luca says that the early registration fee is roughly 320 EUR, which is very low for a five day conference. Thanks for the appreciation, but it is also the fact that this benefit for people outside Japan is just due to the exchange rate. The current rate is approximately 140 JPY per EUR, but just looking back a couple of years ago, it was about 100 JPY per EUR. Some 40 % down only in a few years! If it is the same rate now, then the fee would be some 450 EUR, which may be ok, but no one would say "very cheap." ICALP 2015 in Kyoto was decided two years ago. I gave a short presentation about what it would look like in the business meeting of ICALP 2013, in which I made a sort of promise that the fee should not be more than 350 EUR. At that time, the rate was already something like 130 JPY per EUR, but I had a clear memory that it had been 100 JPY just one year or so before. Who knows if it would come back soon...

We had two sad news. One is that of Jiří Matoušek. We have an obituary by Takeshi Tokuyana in this issue. The other is that of John Nash; Mr and Mrs Nash were killed by a car crash in May. I read a NY Times article as well as many reader's comments there. Many of the readers mentioned the New Jersey Turnpike, one of the busiest and toughest highways in the US, and rear-seat seat-belts (they were not wearing

seal-belts in the rear-seat of the cab and were thrown out of the car). I do know that rear seat-belts in old US cabs are often broken and/or are buried in a gap of rear seats; it is not easy at all for senior people to wear them quickly. I also want to say something about the highway (I remember well when I first drove on the Turnpike, which seemed to me a most exciting American culture), that is its efficiency. The data show that the traffic of the Turnpike is something like 20,000 at a peak time, meaning that number of vehicles (and roughly the same number of people because most cars carry only a driver) go one direction. This may seem a lot, but just consider a single subway line in Tokyo. One train has ten cars, each car accommodates well more than 100 people and they run every 2.5 minutes. Simple calculation shows that this is already more than the monster highway with 6-7 lanes each direction.

I should come back to the Bulletin. As you see in a moment, we started another new section "Reflections on Influential Scientists and Ideas." There is an introduction by Luca about its purpose, but simply speaking, that is a section for essays about big influential scientists and/or their ideas. Our first essay is about George Dantzig and his simplex method by David Avis (the last year is the centenary of George Bernard Dantzig's birth). When David was a Stanford student, he took a class of Dantzig, thus I am sure he is a best person to start this new column. Also this year is the 100th anniversary of the birth of Richard Hamming. I studied the Hamming code long

time ago, but to me the Hamming distance is more impressive as an important notion for TCS people. I think it is a fun to approach such big guys from several different angles of our own. Please consider your contributions to this new section.

This issue's columns are very rich; many thanks to the column editors again and again. Finally I hope I can see you in Kyoto very soon.

Kazuo Iwama, Kyoto
June 2015

Jiří Matoušek
1963 – 2015

Memories of Jiří Matoušek in Japan

I first saw Jirka at SOCG1990 in Berkeley, where he gave the opening talk on his seminal work on cutting hyperplane arrangement. He wore white casual shirts, looked delicate, and spoke dispassionately with low and characteristic voice (with good contrast to Pankaj, another young star in the conference). I felt that I was seeing a picture of a young genius.

Fortunately, Prof. Hiroshi Imai of U. Tokyo invited him to Japan in 1991, and I had opportunity to guide him. After talking in my office, we attended a workshop, where he whispered to me 'Takeshi, I think I have a solution to your problem, if I understand it correctly.' We escaped from the workshop, and wrote a small paper (Complexity of Projected Images of Convex Subdivisions. CGTA (1994)).

Jirka visited us many times after that, sometimes with his family. His talks given calmly using a blackboard and a small memo in his hand enchanted us, and his books were eagerly read and translated into Japanese.

I found Jirka was delicate, scholastic, curious, faithful and unique, and it was fun to work with him. He often spoke euphemistic, and considered carefully before making action. I remember the first time I took him to the cafeteria of my office (IBM Tokyo Research) for lunch. He gazed at the samples (partially plastic imitations) displayed in the showcase, wanted me to take time to explain each of them, and then asked 'Then Takeshi, what is your recommendation'. He replied 'Well, it looks promising' to my suggestion, grinned happily, and enjoyed lunch. Ms. Mary Inaba, who evidenced it and later became an associate professor of U. Tokyo, described him as a 'charming and complicated boy', and became a good friend of Jirka's family.

In 2005, a research project named New Horizon of Computing lead by Kazuo Iwama started in Japan, and we invited him to its kick-off symposium in Kyoto. Since it was his policy not to travel for a symposium/conference, we persuaded him to come to give a talk on the last day and then work privately with Tetsuo Asano (currently the president of Japan Advanced Institute of Science of Technology) and me. At a nice coffee shop in Kyoto, Tetsuo showed us a picture of the distance trisector curve, which he had just invented. To our surprise, the first action of Jirka was to compute the Taylor expansion of the curve using Mathematica. He grinned and told us that he loved to do such experiments to investigate problems. The research was successful, and we submitted a paper to STOC. I was a little afraid that it might be out-of-scope, but Jirka did not consider any possibility of rejection, and proposed to submit its journal version to a good mathematical journal without waiting for STOC's acceptance. The suggestion was wise since the paper (The distance trisector curve. Advances in Math. 212 (2007), no. 1, 338-360) has been widely read and a group of mathematicians resolved one of our conjectures.

Jirka had his own aesthetic sense in selecting the journals to publish papers. The following is his email (with some editing) replying my suggestion to submit another paper to Advances in Mathematics.

"I'm not so keen about Advances since it's an Elsevier journal (which is the worst, in terms of prices and policy, among the publishers, I think). Some other possibilities might be, say, Israel J. Math., GAFA, JOURNAL FUR DIE REINE UND ANGEWANDTE MATHEMATIK (Crelle), or Math. Annalen. For Annalen or Crelle I've never tried, so we could try just for the fun of it. but DCG of course is a natural place too. or still something else ... what do you think? best, Jirka"

I naturally liked his idea to have a paper in one of those legendary journals, and we published the paper in Math. Annalen (Zone diagrams in Euclidean spaces and in other normed spaces. Math. Ann. 354 (2012), no. 4, 1201-1221.)

Jirka was interested in not only mathematics but also culture, history, life, and even politics. The following is the reply to my invitation in 2008.

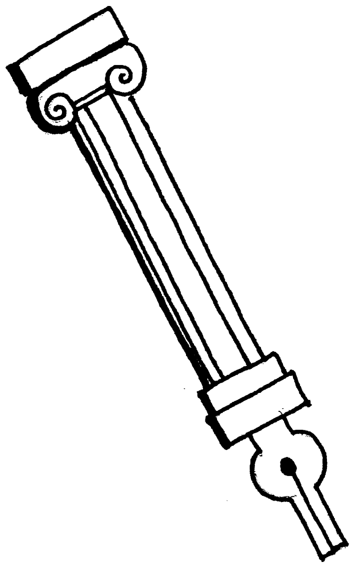
"Thanks for the invitation, I'm seriously interested in coming. I can't promise 100% I'll make it but I hope so. Best, Jirka
P.S. one thing that slightly discourages me - I've heard that Japan now takes (or will take) fingerprints of all visitors, like the US. Do you know anything about it? I find it very unpleasant (even though I probably have nothing to hide...)."

I really miss Jirka, a friend with humanity, kindness, leadership, and sense of humor. However, a privilege of great mathematicians is that they will live forever as legends by their works, and our research community will never forget him. I feel I hear his voice saying "Takeshi, do not worry. I will manage to find papers and a blackboard in the heaven".

Takeshi Tokuyama
Tohoku University, Japan

Institutional Sponsors

EATCS Columns



The Algorithmics Column

BY

Gerhard J Woeginger

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
gwoegi@win.tue.nl

Query-Competitive Algorithms for Computing with Uncertainty

Thomas Erlebach Michael Hoffmann

Department of Computer Science
University of Leicester, UK
{te17,mh55}@leicester.ac.uk

Abstract

Motivated by real-life situations in which exact input data is not available initially but can be obtained at a cost, one can consider the model of computing with uncertainty where the input to a problem is initially not known precisely. For each input element, only a set that contains the exact value of that input element is given. The algorithm can repeatedly perform queries to reveal the exact value of an input element. The goal is to minimize the number of queries needed until enough information has been obtained to produce the desired output. The performance of an algorithm is measured using competitive analysis, comparing the number of queries with the best possible number of queries for the given instance. We give a survey of known results and techniques for this model of queryable uncertainty, mention results for some related models, and point to possible directions for future work.

1 Introduction

In the study of algorithms, we often assume that exact input data is available to the algorithm, and our goal is to design algorithms that compute good solutions efficiently. However, there are also scenarios where precise input data is not immediately available. Instead, only rough, uncertain information about the input may be given, for example intervals or other sets that contain the unknown exact input values. In some such scenarios, it is possible to obtain exact information about each part of the input at an additional cost. We refer to the operation of obtaining exact information for one input element as a *query*, and we are interested in minimizing the number of queries to be made until sufficient information has been obtained for calculating a solution. The performance of an algorithm can then be measured by comparing the number of queries made by the algorithm with the best possible number of queries for the given input, following the approach of competitive analysis for online algorithms.

In this article, we give a survey of known results and techniques for the design of query-competitive algorithms in the model of computing with uncertainty sketched above. We also mention results for related questions and point to possible directions for future work.

1.1 Motivation

There are numerous examples of application scenarios that provide motivation for studying computing with uncertainty. For example, as in Kahan's pioneering paper [12], one can consider applications where the input involves objects (e.g., airplanes) that are in motion and whose current positions are not known precisely. If the exact location of an object was known some time in the past and if the object moves with limited speed, then the current location of the object must lie within a known region, and it may be possible to obtain the exact location of the object by communicating with it (e.g., a radio communication with the pilot of an airplane). The goal may be to compute a function of the locations of the objects (e.g., the closest pair of airplanes). A similar application arises in mobile ad-hoc networks, where estimates of the current positions of the nodes may be easily available but obtaining the exact position of a node requires extra communication or measurements. Kahan mentions further applications such as graphic animation (selecting from moving objects those within the field of vision), integrating the maximum of a set of functions of bounded variation, and dynamic scheduling of jobs with time-varying priorities [12].

Another set of applications originates from distributed computing. For example, in systems with distributed database caches, an estimate of an aggregate data value may be available from a local cache, but obtaining the exact value requires more expensive communication (e.g., a query to the master server). Moreover, a system could maintain intervals of values in local database caches, so that updates to data on the master server need to be replicated to a local cache only when the new value lies outside the interval stored there. When a database query cannot be answered based on the intervals stored in the local cache, exact values can be retrieved from the master server. Olston and Widom propose a concrete replication mechanism employing this principle, the TRAPP system [16].

1.2 Other Approaches to Dealing with Uncertainty

There are various other approaches to modeling or addressing problems whose input involves uncertain information. In *stochastic optimization*, uncertain input values are assumed to be drawn from (known) probability distributions, and the goal is to find a solution that optimizes the expected value of the objective function, or that is good or feasible with high probability. In *robust optimization*, a single solution has to be computed that is good in each possible scenario of how the uncertain values can be realized. In two-stage optimization problems, a partial solution has to be computed based on the given uncertain values, and the solution has to be completed in a second stage after the uncertain values have been realized as exact values. This notion can be generalized to multi-stage optimization problems. Neither of these models or approaches involve querying specific input elements in order to obtain their exact values. Therefore, we consider such work outside the scope of this article.

2 Preliminaries

In computing with uncertainty, an instance of a problem typically consists of some structural information S, a set U of elements with uncertain values, and a function A that maps each element $u \in U$ to an *uncertainty set*[1] A_u. The exact values of the uncertain input elements are represented by a function w that maps each $u \in U$ to its exact value w_u, and we require $w_u \in A_u$. Initially the algorithm might not know any of the values w_u. Querying[2] an element $u \in U$ reveals its exact value w_u. We can view a query of $u \in U$ as the operation of replacing A_u by the singleton set $\{w_u\}$. Note that depending on the concrete problem under consideration, w_u could be a real number, a vector of real numbers, or any other type of input data. In cases where the exact values are vectors representing point coordinates, we will sometimes write p_u for w_u.

For a given instance $I = (S, U, A, w)$ of a problem in the model of computing with uncertainty (we refer to such a problem also as an *uncertainty problem*), we let $\phi(S, U, w)$ denote the set of solutions. Note that the set of solutions depends only on the exact values w_u for $u \in U$ but not the uncertainty sets A_u. An algorithm only receives (S, U, A) as input. The goal of the algorithm is to compute a solution in $\phi(S, U, w)$ (or all solutions in $\phi(S, U, w)$) after making a minimum number of queries. Queries are made one by one, and the results of previous queries can be taken into account when determining the next query to make. Therefore, this model is also referred to as the *adaptive* query model.

For a given instance $I = (S, U, A, w)$, we denote by OPT_I (or simply OPT) the minimum number of queries that provide sufficient information for computing a solution in $\phi(S, U, w)$. An algorithm that makes ALG_I queries to solve an instance I is called *ρ-query-competitive* or simply *ρ-competitive* if $ALG_I \leq \rho OPT_I + c$ for

[1]In the literature, the term *uncertainty area* has also been used.

[2]In the literature, queries have also been called *updates* in this context.

all instances I of the problem, where c is a constant independent of the instance. If $c = 0$, we say that the algorithm is *strongly* ρ*-competitive*. For randomized algorithms, ALG_I is the expected number of queries that the algorithm makes on instance I. Note that the exact value w_u of an uncertain element $u \in U$ can be any value in A_u. We do not assume that w_u is drawn from a probability distribution over the set A_u.

We should briefly clarify what it means formally to say that a set of queries is sufficient to solve a problem. Let $Q \subseteq U$ be a set of queries. Let A^Q be the uncertainty sets arising from A by setting $A_u^Q = \{w_u\}$ if $u \in Q$ and $A_u^Q = A_u$ otherwise. The set of queries Q is sufficient for computing a solution x if and only if $x \in \phi(S, U, w')$ for all w' such that $w'_u \in A_u^Q$ for all $u \in U$ (i.e., if x is a solution for all choices of exact values w' that are consistent with A^Q). If Q is sufficient for computing a solution, we also say that the instance (S, U, A^Q, w) is *solved* (and that Q is a *query solution*), and otherwise *unsolved*.

Another aspect to discuss is which types of sets are allowed as uncertainty sets A_u. In cases where the uncertain elements represent numbers, we will usually assume that each set A_u is either an interval or a singleton set $\{w_u\}$. In the latter case, the exact weight of u is already known, and we say that the element u and its uncertainty set are *trivial*. Elements that are not trivial (and their uncertainty sets) are called *non-trivial*. We write open intervals as (a, b) and closed intervals as $[a, b]$.

Problems in the model of computing with uncertainty (e.g., the minimum spanning tree problem) often display the following behavior: If the task is to compute a single solution, there is a constant-competitive algorithm for the case of open intervals as uncertainty sets but not for the case of closed intervals. The difficulty with closed intervals is that a single query of the right interval can prove that the value of the corresponding element is maximum or minimum among a number of candidates, while a deterministic algorithm has no chance of identifying that interval without querying a large number of elements. On the other hand, if the task is to output all solutions, then a constant competitive ratio can often be achieved also for closed uncertainty sets [12]. Similarly, if the task is to compute the lexicographically first solution, a constant competitive ratio is sometimes possible for closed uncertainty sets [11].

2.1 Example: MST with Uncertainty

Let us consider the minimum spanning tree problem with edge uncertainty. The structural part of an instance of the problem is a connected, undirected graph $G = (V, E)$. The uncertain elements are the edges, and for each edge $e \in E$ an uncertainty set A_e containing its exact weight w_e is given. The task is to compute the edge set of a minimum spanning tree (MST) of the graph G with edge weights

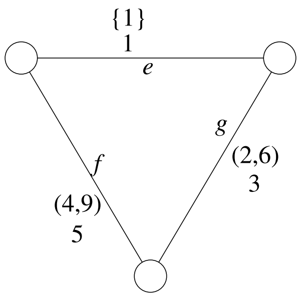


Figure 1: Instance of MST with Uncertainty

given by w, i.e., $\phi(G, E, w)$ is the set consisting of the edge sets of all minimum spanning trees of (G, w).

Consider the example instance $I = (G, E, A, w)$ depicted in Figure 1. The three edges e, f, g have uncertainty sets $A_e = \{1\}$, $A_f = (4, 9)$ and $A_g = (2, 6)$. The exact weights of the edges are $w_e = 1$, $w_f = 5$ and $w_g = 3$. Initially, the algorithm is given only G and A. This information is not sufficient for determining the edge set of an MST: The edge e is clearly contained in any MST, but without querying f or g we do not know which of these two edges is the more expensive one and therefore not contained in the MST. If we decide to query f first, the query reveals that $w_f = 5$. At this point, we still do not know whether f is cheaper or more expensive than g as $A_g = (2, 6)$, so we also need to query g. When we receive the answer $w_g = 3$, we know that g is cheaper than f, and we can determine and output the edge set $\{e, g\}$ as a correct minimum spanning tree. Note that querying only the edge g would also have been sufficient for determining that $\{e, g\}$ is a minimum spanning tree. Therefore, $OPT_I = 1$ but we have made two queries, twice the optimal number. Algorithmic results for the minimum spanning tree problem with uncertainty will be discussed in Section 6.

3 The Witness Algorithm

The main approach that has been used to obtain query-competitive algorithms for computing with uncertainty, as introduced by Bruce et al. [1] and stated in general form by Erlebach et al. [8], is based on considering witness sets: For a given instance $I = (S, U, A, w)$ of an uncertainty problem, a set $W \subseteq U$ is called a *witness set* if it is impossible to determine a solution without querying at least one element of W. In other words, W is a witness set if the instance $(S, U, A^{U \setminus W}, w)$ is unsolved. A natural idea for solving an uncertainty problem is now to repeatedly determine a witness set and query all elements of that witness set (elements with

```
while instance is unsolved do
    $W \leftarrow$ a witness set of the current instance;
    query all $u$ in $W$;
end
```

Algorithm 1: Witness algorithm

a trivial uncertainty set can be skipped, of course), until the instance is solved. An algorithm based on this template, shown in Algorithm 1, is called a *witness algorithm*. Note that the instance changes in each iteration of the while-loop in the algorithm, as the uncertainty sets of the elements in W are replaced by singleton sets.

Most of the known results regarding query-competitive algorithms for uncertainty problems are instantiations or adaptations of the witness algorithm. There is a direct link between the size of witness sets and the competitive ratio, as shown by the following lemma.

Lemma 1 ([1, 8])**.** *If the size of each witness set W that a witness algorithm queries is bounded by ρ, then the algorithm is strongly ρ-competitive*

Lemma 1 can be proved by showing that any query solution must contain at least one distinct element from each witness set queried by the algorithm.

We remark that if an algorithm during its execution queries k sets of elements that are witness sets (and possibly some additional elements), we can infer that any query solution must make at least k queries.

4 Selection and Related Problems

4.1 Identifying All Solutions

Kahan [12] considers the setting where the input consists of a set U of n elements with uncertain values. The uncertainty set of each element $u \in U$ is assumed to be either a closed interval $[u_l, u_h]$ or trivial (in which case the lower bound u_l equals the upper bound u_h).

Maximum. For the problem of identifying all elements of U whose value is equal to the maximum value $\max_{u \in U} w_u$ in U, Kahan presents an algorithm that makes at most $OPT+1$ queries [12]. The algorithm repeatedly queries a non-trivial element u with largest upper bound u_h until all elements with maximum value V have been queried and no non-trivial element u with $u_h \geq V$ remains. Kahan shows that in any query solution at most one non-trivial element u with value $u_h \geq V$ is not queried. As the algorithm queries only elements u with $u_h \geq V$, it makes at most $OPT + 1$ queries.

Considering the input $U = \{u, v\}$ with $A_u = [1, 5]$ and $A_v = [3, 10]$ and either $w_u = 2, w_v = 4$ or $w_u = 4, w_v = 7$, it is clear that making $OPT + 1$ queries in the worst case is optimal among deterministic algorithms.

Median. Assume that $n = |U|$ is odd. For the problem of identifying all elements of U whose value is equal to the value of the median (i.e., the $\lceil n/2 \rceil$-smallest value) of $\{w_u \mid u \in U\}$, Kahan also presents an algorithm that makes at most $OPT + 1$ queries [12]. Let $k = \lceil n/2 \rceil$. It is clear that the value of the median must lie in the range $[m_l, m_h]$, where m_l is the k-smallest lower bound and m_h is the k-largest upper bound. Kahan shows that there must be at least one element $u \in U$ with $[m_l, m_h] \subseteq [u_l, u_h]$. The algorithm for the median problem now works as follows: Compute $[m_l, m_h]$. If there is only one element u whose interval intersects $[m_l, m_h]$, return u as the unique median and quit. If every element whose range contains $[m_l, m_h]$ has already been queried, we must have $m_l = m_h$ and we output all those elements as the set of medians and quit. Otherwise, we query any element whose interval contains $[m_l, m_h]$ and repeat. Again, Kahan shows that in any query solution at most one non-trivial element u whose interval contains the median value is not queried. As the algorithm queries only elements u whose intervals contain the median value, it makes at most $OPT + 1$ queries, and this is again optimal among deterministic algorithms.

Minimum Gap. Now consider the Minimum Gap problem, where the goal is to identify all pairs of elements $u, v \in U$ such that their distance $|w_u - w_v|$ is minimum. Note that for any two elements of U, their uncertainty sets imply lower and upper bounds on the distance between these two elements (and we thus also have an upper bound on the minimum gap). For the Minimum Gap problem, Kahan presents a (weakly) 2-competitive greedy algorithm. The algorithm first queries all non-trivial elements whose interval contains the interval of at least one other element. These elements (except possibly one of them) must be queried by any query solution. The algorithm then repeatedly queries a non-trivial element u for which there is another element v such that the lower bound on the distance between u and v is smaller than the current upper bound on the minimum gap, and is the smallest current lower bound among all pairs of elements (with ties broken in favor of an element whose interval minimizes the maximum distance to non-trivial intervals on the left and right). Kahan analyzes the algorithm by considering connected components of elements in an auxiliary graph that contains an edge between any two elements whose distance lower bound is less than or equal to the minimum gap. He shows that any query solution must contain at least half the elements in a component (with at most one exception), while the greedy algorithm may contain all its elements. Hence, the algorithm is 2-competitive. Furthermore, one can show that no deterministic algorithm can achieve weak competitive ratio smaller than 2 [12].

We remark that an alternative 2-competitive algorithm can be obtained using a witness algorithm: Let L be the current lower bound on the minimum gap. While there are two elements u, v whose distance lower bound is at most L (and at least one of which is non-trivial), query all non-trivial elements in $\{u, v\}$. It is not difficult to see that the set $\{u, v\}$ is a witness set, except possibly in the case where u, v is the unique pair of elements whose distance equals the minimum gap. If the algorithm terminates after k iterations, the algorithm makes at most $2k$ queries and, by the remark after Lemma 1, any query solution makes at least $k-1$ queries. Hence, the algorithm makes at most $2OPT + 2$ queries, matching the performance of Kahan's algorithm.

4.2 Identifying One Solution

Khanna and Tan [13] also consider the setting where the input consists of a set U of n elements with uncertain values and the uncertainty set of each element $u \in U$ is either a closed interval $[u_l, u_h]$ or trivial. Defining ω to be the maximum clique size of the interval graph induced by the uncertainty sets of the elements of U, they present an ω-competitive algorithm for the problem of identifying a k-smallest $u \in U$, for given $k \in \{1, \ldots, n\}$. They generalize their result also to the case where queries have different costs, i.e., querying $u \in U$ has a cost of c_u instead of unit cost. Furthermore, they consider the variations of the problem where the goal is to identify an element whose rank in U differs from the given rank k at most by a factor of α (*relative precision*) or by an absolute difference of α (*absolute precision*). They present $O(\omega)$-competitive algorithms for these variations. To show that no algorithm can be better than ω-competitive for the problem of identifying an element with smallest value, Khanna and Tan consider an instance with n elements where all uncertainty sets are $[0, 2]$, one element has value 0, and $n - 1$ elements have value 1. This instance has clique size $\omega = n$. A deterministic algorithm can be forced to query all n elements (all queries except the last one return the value 1), while the optimal query set has size 1 and consists of only the minimum element.

4.3 Computing or Approximating the Solution Value

Khanna and Tan [13] additionally consider the problem of approximating the sum (or average) of the values of the n elements in U up to a certain relative precision $\alpha \geq 1$. They present an $O(\min\{\omega, \alpha\})$-competitive algorithm that can also handle different query costs. For the problem of approximating the sum of the values of the elements in U with absolute precision in the case of unit query costs, they remark that a simple greedy algorithm yields the optimal query set. Furthermore, they consider the computation of functions that are composed of aggregation functions and selection functions.

Charikar et al. [3] consider the evaluation of AND/OR trees and generalizations including MIN/MAX trees and present query algorithms with best possible competitive ratio (or within a factor of 2 of the best possible competitive ratio in the generalized case).

5 Geometric Problems

Bruce et al. [1] consider geometric problems in the Euclidean plane in the model of computing with uncertainty. Each input element $u \in U$ corresponds to a point $p_u = (x_u, y_u)$ in the Euclidean plane. The uncertainty set A_u of each $u \in U$ is assumed to be either trivial or the closure of an open, connected region. We refer to an uncertainty set also as an *uncertainty region* in the following.

Bruce et al. introduce the concepts of witness sets and witness algorithms and apply them to the maximal points problem and the convex hull problem. For a property ϕ (such as being a maximal point or a point lying on the convex hull), they classify each uncertainty region A_u as *always* ϕ, *partly* ϕ, *dependent* ϕ or *never* ϕ, depending on whether all points or some point in A_u satisfy the property ϕ and on whether this depends on the point locations in the other uncertainty regions or not.

Maximal points. The task of the maximal points problem is to output all elements $u \in U$ that are maximal, i.e., all u such that there is no element $v \in U$ with $x_v \geq x_u$ and $y_v \geq y_u$ where at least one of the two inequalities is strict. Bruce et al. show that if there is a partly maximal region, then there is a witness set of size 2, and if there is a dependent maximal region but no partly maximal region, then there is a witness set of size 3. Hence, one can always find a witness set of size at most 3 until the instance is solved (i.e., until all uncertainty regions are either always maximal or never maximal), and by Lemma 1 the algorithm is strongly 3-competitive. They also give a lower bound showing that no deterministic algorithm can be better than 3-competitive and prove that there is a linear lower bound on the competitive ratio if arbitrary connected sets are allowed as uncertainty regions.

Convex hull. The task of the convex hull problem is to identify all elements $u \in U$ that correspond to points that lie on the boundary of the convex hull of the point set $\{p_u \mid u \in U\}$. Assuming that the intersection between any two non-trivial uncertainty regions that is non-empty contains at least an ε-ball for some $\varepsilon > 0$, Bruce et al. show that there is always a witness set of size 3, implying a strongly 3-competitive algorithm by Lemma 1. Furthermore, they show that no deterministic algorithm can achieve competitive ratio less than 3, and that there is a linear lower bound on the competitive ratio if arbitrary connected uncertainty sets are allowed.

6 Minimum Spanning Tree

Erlebach et al. [8] consider two variants of the minimum spanning tree (MST) problem in the model of computing with uncertainty. In both variants, the task of the algorithm is to output the edge set of a minimum spanning tree of a given graph $G = (V, E)$. The graph G here represents the structural information S that is provided to the algorithm in addition to the uncertain information.

In the MST problem with edge uncertainty, the edge weights of the given graph $G = (V, E)$ are given in the form of uncertainty sets, i.e., we have $U = E$ and the weight of each edge $e \in E$ is a value w_e contained in the uncertainty set A_e. The uncertainty set of each edge is either trivial or an open set. Each edge has a lower bound and an upper bound (the infimum and the supremum of its uncertainty set). A strongly 2-competitive algorithm is obtained by adapting Kruskal's algorithm to the setting of uncertainty. Edges are inserted into a growing forest in non-decreasing order of lower bounds. When a cycle is formed and if it is not possible to identify an edge of maximum weight in the cycle, a witness set consisting of two edges of the cycle can be identified and queried, and the algorithm is restarted. Otherwise, an edge of maximum weight in the cycle is removed and the algorithm continues. Erlebach et al. also show that no deterministic algorithm can achieve competitive ratio smaller than 2. We illustrate the proof of this lower bound using the graph shown in Figure 1 in Section 2.1: If the algorithm queries f first, the weights of f and g are as shown in the figure. If the algorithm queries g first, the weights are set to $w_f = 8$ and $w_g = 5$. In either case, the algorithm must make two queries while there is a query solution with just one query, and the example can be scaled up arbitrarily by repeating the triangle.

In the MST problem with vertex uncertainty, each vertex $v \in V$ corresponds to a point p_v in the Euclidean plane and the weight of an edge $\{u, v\}$ is the Euclidean distance between p_u and p_v. Instead of the exact point locations, only an uncertainty set A_v is given for each vertex v. It is assumed that each uncertainty set is either trivial or an open region. Erlebach et al. address the problem by observing that it can be viewed as an edge uncertainty problem since the uncertainty sets of the two endpoints of an edge $e = \{u, v\}$ yield an uncertainty set for the weight of e that is trivial or an open set. Hence, the 2-competitive algorithm for the edge uncertainty case can be applied to the vertex uncertainty case, where the query of an edge is replaced by queries of both endpoints of the edge. This yields a strongly 4-competitive algorithm for MST with vertex uncertainty, and it is also shown that no deterministic algorithm can achieve a better competitive ratio.

In [6], it is shown that the strongly 2-competitive algorithm for MST with edge uncertainty can be generalized to the minimum weight matroid base problem.

7 Cheapest Set Problems

Many combinatorial problems can be viewed as set selection problems: The input contains a set U of elements, some subsets of U are *feasible* solutions, and the goal is to output a feasible solution of minimum cost (where the cost of a solution could be the sum of the costs of its elements). Examples of problems that can be seen as set selection problems include the minimum spanning tree problem (the feasible solutions are the edge sets of all spanning trees), the minimum-weight perfect matching problem (the perfect matchings are the feasible subsets of the edge set), or minimum cut problems (the set of edges crossing a cut is a feasible solution). Erlebach et al. [6] study the general formulation of the set selection problem where the input specifies a set U and the family $\mathcal{F}$ of all feasible subsets of U. Instead of the exact weight w_u of each element $u \in U$, an uncertainty set A_u that can be an open interval or trivial is given. The task of the cheapest set problem is to identify a set C in $\mathcal{F}$ that has minimum weight, i.e., $\sum_{u \in C} w_u$ is minimum among all sets in $\mathcal{F}$.

For the case that all sets in $\mathcal{F}$ have cardinality at most d, it is shown that there is an algorithm for the cheapest set problem that makes at most $dOPT + d$ queries, and that this is best possible among deterministic algorithms. The algorithm repeatedly queries all elements of a *robust cheapest set* until a cheapest set can be identified. Here, a robust cheapest set is a set C in $\mathcal{F}$ with the following property: For any choice of exact weights $w_u \in A_u$ for the elements of $u \in U \setminus C$, there are weights $w_u \in A_u$ for each element $u \in C$ such that C is a cheapest set. The analysis shows that a robust cheapest set always exists and that all sets queried by the algorithm, except at most one, are witness sets. By the remark after Lemma 1, this shows that the algorithm makes at most $dOPT + d$ queries.

For cheapest set problems where the sets have a special structure, better algorithms are possible. As we have seen, the minimum spanning tree problem, although corresponding to a cheapest set problem with sets of size $n - 1$ for graphs with n vertices, admits a 2-competitive algorithm. Another family of cheapest set problems with special structure considered in [6] is the family of problems that satisfy the *1-gap property*: If the instance is not solved, there exist two sets $B, C \in \mathcal{F}$ such that $|B \cup C| \leq d + 1$, C is a robust cheapest set, and B is a set that is potentially cheaper than C. An algorithm is presented that makes at most $dOPT + 1$ queries to solve a cheapest set problem that satisfies the 1-gap property. Furthermore, it is shown that the minimum-multicut problem in trees with d terminal pairs satisfies the 1-gap property and can hence be solved with $dOPT + 1$ queries. A matching lower bound showing that this is best possible among deterministic algorithms is also presented.

8 Computing *OPT*: The Verification Problem

In the competitive analysis of algorithms for computing with uncertainty, the number of queries is always compared with the best possible number OPT of queries that suffice to produce the desired output. (Kahan [12] refers to this number OPT as the *lucky number*.) The question then arises how difficult it is to calculate the value of OPT (and possibly also a query solution consisting of OPT queries) if the whole instance (S, U, A, w) of a problem is provided as input (i.e., if also the exact weights of all elements in U are provided to the algorithm). There are at least two scenarios that motivate the computation of an optimal query solution: On the one hand, if the performance of algorithms for computing with uncertainty is evaluated in computational experiments, it is useful to be able to calculate OPT in order to compare the number of queries made by different algorithms with OPT. On the other hand, there may be application scenarios where the values of uncertain elements can change over time, but changes are rare events. Assume that the exact values w_u of all uncertain elements $u \in U$ were known at a point in the past. Let w'_u denote the (unknown) current exact value of each uncertain element $u \in U$. For each $u \in U$, an uncertainty set A_u that guarantees $w'_u \in A_u$ is given. The values w'_u are unknown to the algorithm. We would now like to have an algorithm, called a *verification algorithm*, that queries elements of U and produces one of the following outputs:

(1) A valid solution x with respect to the current weights w'_u.

(2) At least one element $u \in U$ has changed its value (i.e., $w_u \neq w'_u$).

In scenarios where the typical case is that $w'_u = w_u$ for all $u \in U$, querying the elements of an optimal query set for (S, U, A, w) is the best possible strategy for a verification algorithm since it minimizes the number of queries that need to be made until a valid solution x can be identified. Following the terminology of the latter application setting, we refer to the (off-line) problem of computing an optimal query set for a given instance (S, U, A, w) (which is wholly presented to the algorithm) as a *verification problem* (or the *verification version* of an uncertainty problem).

It is easy to see that the verification versions of the maximum, median and minimum gap problems discussed in Section 4 can be solved in polynomial time. In [5], we show that the verification version of MST with edge uncertainty can be solved in polynomial time while the verification version of MST with vertex uncertainty is *NP*-hard.

For general cheapest set problems, Erlebach et al. [7] show that the verification problem is *NP*-hard for sets of size d for any $d \geq 2$. For the minimum multicut problem in trees with d terminal pairs under uncertainty, they prove that the verification version can be solved in polynomial time for fixed d and is *NP*-hard if d is part of the input.

For the problem of outputting all maximal points for a given uncertain point set, Charalambous and Hoffmann [2] show that the verification problem is *NP*-hard. In their reduction, every uncertainty set is either trivial or consists of just two points. The complexity of the verification problem for the case of connected uncertainty sets is left open.

9 Model Variations

9.1 Refinement Queries

We have assumed so far that the query of an uncertain element $u \in U$ returns its exact value w_u. One may also consider settings where the answer to a query is not the exact value, but a reduction of uncertainty. For example, the query of an uncertainty interval $[5, 10]$ could produce a smaller uncertainty interval $[7, 9]$. The same uncertain element may now have to be queried several times, each query reducing the uncertainty interval further. We can then again aim to minimize the number of queries needed until a solution can be computed.

Tseng and Kirkpatrick [17] consider this problem variation in a setting where each input number is given as a finite stream of bits (given in decreasing order of significance) that are initially unknown to the algorithm. Reading an additional bit from one of the bit streams corresponds to a query. The goal is to minimize the number of bits that the algorithm needs to read before it can compute a solution. Algorithms that perform well with respect to this measure are called *input-thrifty* algorithms. Tseng and Kirkpatrick analyze the performance of input-thrifty algorithms relative to a suitably defined *intrinsic cost* of the given instance. They present an algorithm for the extrema testing problem that is within a logarithmic factor of the intrinsic cost of the given instance. They mention that their results also hold in a more general model where a query does not yield one extra bit of a number but the query results are nested uncertainty intervals.

Gupta et al. [11] study the selection problem and the minimum spanning tree problem in a general framework where queries yield refined estimates in the form of sub-intervals. They distinguish all possible cases of combinations of closed intervals, open intervals and trivial sets for the input uncertainty and for the results of queries. Although there are 49 possible models in principle, they are able to classify them into five different main categories and present results on the competitive ratio for selection problems and minimum spanning tree problems for them. For example, they show that the approach of witness algorithms can be generalized to the model with refinement queries for several of the five categories. They also show that for models with closed intervals, query-competitive algorithms become possible if the output is required to be a lexicographically smallest solution rather than an arbitrary solution.

9.2 Non-adaptive Queries

Instead of allowing the algorithm to make queries one by one and take into account the results of previous queries when selecting the next query, one can also require that the algorithm specifies a set $Q \subseteq U$ of queries only once. The queries are then executed in parallel and the algorithm receives the exact values w_u for all queries $u \in Q$. This information must be enough for the algorithm to solve the problem, i.e., no further queries are allowed. This means that the query results, no matter which element of A_u turns out to be the exact value of $u \in Q$, must be sufficient to be able to identify a solution. A query set Q with this property is called *feasible*. The problem of computing a smallest feasible query set Q in this non-adaptive model is actually an off-line problem, as the feasibility of a query set Q does not depend on the exact values of $u \in U$. Note that the problem of determining a smallest feasible query set for the non-adaptive model is different from the verification problem discussed in Section 8.

Much of the existing work on the non-adaptive query model has considered the generalization where different queries have different costs, i.e., each $u \in U$ has a cost $c_u \geq 0$ and the goal is to compute a feasible query set Q of minimum total cost. For given elements with closed intervals as uncertainty sets, Olston and Widom consider selection and aggregation problems in the non-adaptive model, assuming that the goal is to output a range $[L, H]$ that contains the exact solution, where $H - L \leq \delta$ for a given precision requirement δ [16]. For example, it turns out that for the problem of computing the sum of uncertain values up to a given precision, determining an optimal feasible query set boils down to solving a knapsack problem. Feder et al. [10] further study the problem of determining the value of the k-th smallest element, for any $1 \leq k \leq n$, up to a precision of δ in the same model and show that a feasible query set of minimum cost can be computed in polynomial time via linear programming.

In another paper, Feder et al. [9] consider the shortest s-t path problem in the non-adaptive query model. The input consists of a directed acyclic graph $G = (V, E)$ with distinguished vertices $s, t \in V$. The edge weights are uncertain. Each uncertainty set A_e that contains the exact weight w_e of an edge $e \in E$ is assumed to be a closed interval. A set P of candidate s-t-paths is given explicitly as part of the input or implicitly via a description in a certain recursive form (that captures, e.g., the set of all s-t paths in a series-parallel graph). For a given precision parameter $\delta > 0$, the task is to determine a range $[L, H]$ with $H - L \leq \delta$ that contains the length of a shortest s-t-path in P with respect to the exact weights $w_e \geq 0$ for all $e \in E$. They show that the problem of computing a feasible query set of minimum total cost can be solved in polynomial time if $\delta = 0$, i.e., if the exact length of the shortest path in P is sought. For $\delta > 0$, they show that different restrictions of the problem are *NP*-hard or co-*NP*-hard and so the general problem cannot be in *NP* nor co-*NP* if *NP* $\neq$ co-*NP*. They also show that the general problem is in Σ_2.

9.3 Solutions of Minimum or Maximum Objective Value

Another question that has been studied for optimization problems with uncertain input is determining the minimum or maximum possible objective value of an optimal solution, where the minimum/maximum is taken over all possible exact values that the uncertain input elements could take. This is an off-line problem, and queries are not considered in this setting.

For example, Löffler and van Kreveld study the problem of minimizing or maximizing a number of basic geometric measures for input point sets whose locations are described by uncertainty sets. They consider measures such as diameter, width, closest pair, smallest enclosing circle, smallest enclosing bounding box, length of the convex hull, area of the convex hull, or length of a tour visiting the points in a given order. For all these measures and different types of uncertainty sets (e.g., disks, squares, line segments), they present hardness results or efficient algorithms [14, 15]. Dorrigiv et al. [4] consider the setting where the input consists of points in the plane whose uncertainty regions are disjoint disks. They prove that even for this restricted setting it is *NP*-hard to determine the minimum or maximum possible cost of a Euclidean spanning tree of the exact points, and that there is no FPTAS for these problems unless $P = NP$. They also give approximation algorithms with ratios depending on a certain separability parameter of the given instance.

10 Directions for Future Work

We have aimed to give an overview of the existing body of work for computing with uncertainty in models where the algorithm can query the exact values of uncertain elements. Interesting questions that could be addressed in future work include:

- For which uncertainty problems can *randomized algorithms* achieve better competitive ratios than the best possible deterministic algorithms? In particular, it would be interesting to see whether a randomized algorithm can improve over the 2-competitive algorithm for the minimum spanning tree problem under edge uncertainty [8]. The adversarial construction that yields the lower bound of 2 for deterministic algorithms can be adapted to a lower bound of 1.5 for randomized algorithms, but it is open whether there exists a randomized algorithm with competitive ratio better than 2.

- Existing work has considered the adaptive query model where queries are asked one by one, and the non-adaptive query model where all queries are asked in parallel. It may be interesting to consider a model where *queries are asked in rounds*. Each round makes a number of queries in parallel, and the results of queries made in previous rounds can be taken into account when determining the queries for the next round. For example, one could study the trade-off between the number of rounds and the total query cost or consider settings where the number of rounds is constrained to be at most a given value k. Another direction would be to restrict the number of queries that can be made in one round to a given number and aim to minimize the number of query rounds.

- As far as we are aware, in the existing published work on computing with uncertainty it has been assumed that the uncertainty of different input elements is independent, in the sense that receiving the answer of one query does not provide any information about the exact values of the remaining uncertain elements. One could imagine settings where querying one element also reduces the uncertainty of other elements. For example, if the input consists of three uncertain numbers together with the exact value of their sum, learning the exact value of one number may potentially reduce the size of the uncertainty sets of the other two numbers. It would be interesting to study uncertainty problems with such *dependencies between the uncertainty sets of different input elements*.

- As discussed in Section 9.3, for every uncertain problem input one can ask for the minimum and maximum possible objective value of the solution, over all possible exact input values that lie in the given uncertainty sets. One can view the difference between the minimum and the maximum objective value as uncertainty in the solution value, and an interesting direction could be to study query strategies that *reduce the uncertainty in the solution value* to a given target value.

- It would be interesting to develop additional techniques or even a *general theory* that allows us to classify problems with respect to the best possible query-competitive ratio that can be achieved. For example, we know that the existence of small witness sets (that can be found by the algorithm) for a problem leads to algorithms with small competitive ratio by Lemma 1, but it is unclear whether there are further general criteria with this property. As an example of a more general type of result, we mention that Kahan [12] considers the problem of computing a function f of n uncertain real-valued elements. The n exact input values can be represented as a point in n-dimensional space. The function f maps $\mathbb{R}^n$ to $\mathbb{R}$ and partitions its domain into regions where the function value is constant. Kahan relates the achievable competitive ratio to the existence of a point on some partition boundary that is tangent to an $(n-1)$-dimensional hyperplane intersecting k co-ordinate axes.

- As a query algorithm that solves an uncertainty problem can be viewed as trying to learn the solution by asking queries about the input, it appears that there may be *connections to work in machine learning* that could be explored further.

References

[1] R. Bruce, M. Hoffmann, D. Krizanc, and R. Raman. Efficient update strategies for geometric computing with uncertainty. *Theory of Computing Systems*, 38(4):411–423, 2005.

[2] G. Charalambous and M. Hoffmann. Verification problem of maximal points under uncertainty. In *International Workshop on Combinatorial Algorithms (IWOCA 2013)*, LNCS 8288, pages 94–105. Springer, 2013.

[3] M. Charikar, R. Fagin, V. Guruswami, J. M. Kleinberg, P. Raghavan, and A. Sahai. Query strategies for priced information. *Journal of Computer and System Sciences*, 64(4):785–819, 2002.

[4] R. Dorrigiv, R. Fraser, M. He, S. Kamali, A. Kawamura, A. López-Ortiz, and D. Seco. On minimum- and maximum-weight minimum spanning trees with neighborhoods. *Theory of Computing Systems*, November 2014. Online First publication.

[5] T. Erlebach and M. Hoffmann. Minimum spanning tree verification under uncertainty. In *41st International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2014)*, LNCS 8747, pages 164–175. Springer, 2014.

[6] T. Erlebach, M. Hoffmann, and F. Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. In *39th International Symposium on Mathematical Foundations of Computer Science (MFCS 2014), Part II*, pages 263–274. Springer, 2014.

[7] T. Erlebach, M. Hoffmann, and F. Kammer. Query-competitive algorithms for cheapest set problems under uncertainty. Full version of [6]. Submitted, 2014.

[8] T. Erlebach, M. Hoffmann, D. Krizanc, M. Mihalák, and R. Raman. Computing minimum spanning trees with uncertainty. In *25th International Symposium on Theoretical Aspects of Computer Science (STACS'08)*, pages 277–288, 2008.

[9] T. Feder, R. Motwani, L. O'Callaghan, C. Olston, and R. Panigrahy. Computing shortest paths with uncertainty. *Journal of Algorithms*, 62(1):1–18, 2007.

[10] T. Feder, R. Motwani, R. Panigrahy, C. Olston, and J. Widom. Computing the median with uncertainty. *SIAM Journal on Computing*, 32(2):538–547, 2003.

[11] M. Gupta, Y. Sabharwal, and S. Sen. The update complexity of selection and related problems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*, volume 13 of *LIPIcs*, pages 325–338. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

[12] S. Kahan. A model for data in motion. In *23rd Annual ACM Symposium on Theory of Computing (STOC'91)*, pages 267–277, 1991.

[13] S. Khanna and W.-C. Tan. On computing functions with uncertainty. In *20th Symposium on Principles of Database Systems (PODS'01)*, pages 171–182, 2001.

[14] M. Löffler and M. J. van Kreveld. Largest and smallest convex hulls for imprecise points. *Algorithmica*, 56(2):235–269, 2010.

[15] M. Löffler and M. J. van Kreveld. Largest bounding box, smallest diameter, and related problems on imprecise points. *Computational Geometry*, 43(4):419–433, 2010.

[16] C. Olston and J. Widom. Offering a precision-performance tradeoff for aggregation queries over replicated data. In *26th International Conference on Very Large Data Bases (VLDB'00)*, pages 144–155, 2000.

[17] K.-C. R. Tseng and D. G. Kirkpatrick. Input-thrifty extrema testing. In *22nd International Symposium on Algorithms and Computation (ISAAC 2011)*, LNCS 7074, pages 554–563. Springer, 2011.

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

Combinatorial games are a fascinating topic, as both recreational and serious mathematics. One aspect of the mathematical theory deals with assigning values that quantify positions in games. This has lead to deep connections between numbers and games and a rich algebraic theory as discovered by Conway.

A natural algorithmic question that arises is the following: given a position in a combinatorial game as input, determine which player has a winning strategy. In their article, Steve Fenner and John Rogers give an excellent self-contained introduction to the computational complexity of combinatorial games. They first explain the basic theory of combinatorial games, with focus on poset games. With this background, they survey the complexity-theoretic results in this field and discuss several open questions.

Combinatorial Game Complexity: An Introduction with Poset Games

Stephen A. Fenner
University of South Carolina,
Computer Science and Engineering Department

John Rogers
DePaul University,
School of Computing

Abstract

Poset games have been the object of mathematical study for over a century but little has been written on the computational complexity of determining important properties of these games. In this introduction we define combinatorial games and focus for the most part on impartial poset games, of which Nim is perhaps the best-known example. We present the complexity results known to date, some discovered very recently.

An extended version of this paper, with detailed proofs, is available online at `http://www.cse.sc.edu/~fenner/papers/games.html` and on `arXiv.org`.

1 Introduction

Combinatorial games have long been studied (see [5, 1], for example) but the record of results on the complexity of questions arising from these games is rather spotty. Our goal in this introduction is to present several results—some old, some new—addressing the complexity of the fundamental problem given an instance of a combinatorial game:

> Determine which player has a winning strategy.

A secondary, related problem is

> Find a winning strategy for one or the other player, or just find a winning first move, if there is one.

The former is a decision problem and the latter a search problem. In some cases, the search problem clearly reduces to the decision problem, i.e., having a solution for the decision problem provides a solution to the search problem. In other cases this is not at all clear, and it may depend on the class of games you are allowed to query.

We give formal definitions below, but to give an idea of the subject matter, we will discuss here the large class of games known as the *poset games*. One of the best known of these is Nim, an ancient game, but given its name by Charles Bouton in 1901 [2]. There are many others, among them, Hackendot, Divisors, and Chomp [5]. Poset games not only provide good examples to illustrate general combinatorial game concepts, but they also are the subject of a flurry of recent results in game complexity, which is the primary focus of this article.

The rest of this section gives some basic techniques for analyzing poset games. Section 2 lays out the foundations of the general theory of combinatorial games, including numeric and impartial games. The rest of the paper is devoted to computational complexity. Section 3 gives an upper bound on the complexity of so-called "N-free" games, showing that they are solvable in polynomial time. Section 4 gives lower bounds on the complexity of some games, showing they are hard for various complexity classes. The section culminates in two recent **PSPACE**-completeness results—one for impartial poset games, and the other for "black-white" poset games. Section 5 discusses some open problems.

An extended version of this paper, with detailed proofs, is available online at `http://www.cse.sc.edu/~fenner/papers/games.html` and on `arXiv.org`.

1.1 Poset games

Definition 1.1. A *partial order* on a set P (hereafter called a *poset*) is a binary relation $\leq$ on P that is reflexive, transitive, and antisymmetric (i.e., $x \leq y$ and $y \leq x$ imply $x = y$). For any $x \in P$, define $P_x := \{y \in P \mid x \not\leq y\}$.

We identify a finite poset P with the corresponding *poset game*: Starting with P, two players (Alice and Bob, say) alternate moves, Alice moving first, where a move consists of choosing any point x in the remaining poset and removing all y such that $x \leq y$, leaving P_x remaining. Such a move we call *playing* x. The first player unable to move (because the poset is empty) loses.[1]

Poset games are *impartial*, which means that, at any point in the play, the set of legal moves is the same for either player. There is a rich theory of impartial games, and we cover it in Section 2.5.

[1]Games can be played on some infinite posets as well, provided every possible sequence of moves is finite. This is true if and only if the poset is a well-quasi-order (see, e.g., Kruskal [18]).

In an impartial game, the only meaningful distinction between players is who plays first (and we have named her Alice). Since every play of a poset game has only finitely many moves, one of the two players (but clearly not both!) must have a winning strategy. We say that a poset P is an $\exists$-*game* (or *winning position*) if the first player has a winning strategy, and P is a $\forall$-*game* (or *losing position*) if the second player has a winning strategy. In the combinatorial game theory literature, these are often called $\mathcal{N}$-games ("Next player win") and $\mathcal{P}$-games ("Previous player win"), respectively. We get the following concise inductive definition for any poset P:

P is an $\exists$-game iff there exists $x \in P$ such that P_x is a $\forall$-game.
P is a $\forall$-game iff P is not an $\exists$-game (iff, for all $x \in P$, P_x is an $\exists$-game).

We call the distinction of a game being a $\forall$-game versus an $\exists$-game the *outcome* of the game.

There are at least two natural ways of combining two posets to produce a third.

Definition 1.2. For posets $P = \langle P, \leq_P \rangle$ and $Q = \langle Q, \leq_Q \rangle$,

- define $P + Q$ (the *parallel union of P and Q*) to be the disjoint union of P and Q, where all points in P are incomparable with all points in Q:

$$P + Q := \langle P \dot{\cup} Q, \leq \rangle ,$$

where $\leq := \leq_P \dot{\cup} \leq_Q$.

- Define P/Q (or $\frac{P}{Q}$—the *series union of P over Q*) to be the disjoint union of P and Q where all points in P lie above (i.e., are $\geq$ to) all points in Q:

$$\frac{P}{Q} := \langle P \dot{\cup} Q, \leq \rangle ,$$

where $\leq \; := \; \leq_P \dot{\cup} \leq_Q \dot{\cup} (Q \times P)$.

Note that $+$ is commutative and associative, and that $/$ is associative but not commutative. Using these two operations, let's build some simple posets. Let C_1 be the one-element poset. For any $n \in \mathbb{N}$, let

1. $C_n := \underbrace{C_1/C_1/\ldots/C_1}_{n}$ is the chain of n points (totally ordered). This is also called a *NIM stack*.

2. $A_n := \underbrace{C_1 + C_1 + \cdots + C_1}_{n}$ is the antichain of n pairwise incomparable points.

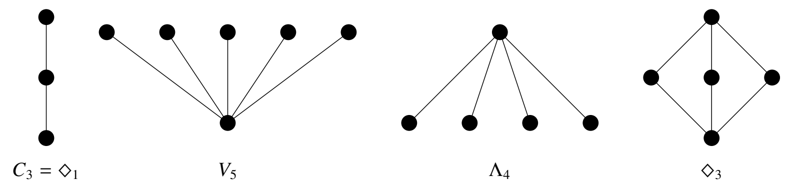


Figure 1: Some simple posets constructed from individual points via parallel and series union.

3. $V_n := A_n/C_1$ is the n-antichain with a common lower bound.
4. $\Lambda_n := C_1/A_n$ is the n-antichain with a common upper bound.
5. $\Diamond_n := C_1/A_n/C_1$ is the n-antichain with common upper and lower bounds.

Some examples are shown in Figure 1.

Exercise 1.3. Find a simple way, given m and n, to determine whether A_m/A_n is an $\exists$-game or a $\forall$-game.

Exercise 1.4. Show that P/Q is an $\exists$-game if and only if either P is an $\exists$-game or Q is an $\exists$-game.

1.1.1 More examples

The best-known poset game is Nim, an ancient game first formally described and "solved" by C. L. Bouton in 1902 [2]. Here, the poset is a union of disjoint chains, i.e., of the form $C_{n_1} + C_{n_2} + \cdots + C_{n_k}$ for some positive integers $n_1, \ldots, n_k$. A move then consists of choosing a point in one of the chains and remove that point and everything above it.

Other families of poset games include

Chomp, introduced in 1974 by D. Gale [11], which, in its finite form, is represented by a rectangular arrangement of squares with the leftmost square in the bottom row removed. This is a poset with two minimal elements (first square on the second row, second square on bottom row). Every element in a row is greater than all of the elements to the left and below so playing an element removes it and all elements to the right and above.

Hackendot, attributed to von Newmann, where the poset is a forest of upside-down trees (roots at the top). Hackendot was solved in 1980 by Úlehla [29].

DIVISORS, introduced by F. Schuh [22], the poset is the set of all positive divisors (except 1) of a fixed integer n, partially ordered by divisibility. DIVISORS is a multidimensional generalization of CHOMP. CHOMP occurs as the special case where $n = p^m q^n$ for distinct primes p, q.

1.2 Dual symmetry

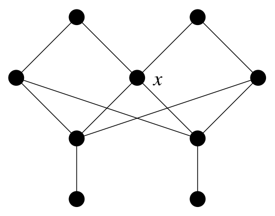
Some poset games can be determined (as $\exists$-games or $\forall$-games just by inspection). For example, suppose a poset P has some kind of dual symmetry, that is, there is an order-preserving map $\varphi : P \to P$ such that $\varphi \circ \varphi = \text{id}$.

Fact 1.5. *Let P be a poset and let $\varphi : P \to P$ be such that $\varphi \circ \varphi = \text{id}_P$ and $x \leq y \iff \varphi(x) \leq \varphi(y)$ for all $x, y \in P$.*

- *If φ has no fixed points, then P is a $\forall$-game.*
- *If φ has a minimum fixed point (minimum among the set of fixed points), then P is an $\exists$-game.*

Proof. If φ has no fixed points, then Bob can answer any x played by Alice by playing $\varphi(x)$. If φ has a least fixed point z, then Alice plays z on her first move, leaving P_z, which is symmetric with no fixed points, and thus a $\forall$-game. □

For example, the poset below is symmetric with a unique fixed point x, which Alice can win by playing on her first move:



1.3 Strategy stealing

Another class of posets that are easy to determine by inspection are those with an *articulation point*, i.e., a point that is comparable with every other point in the poset. For example, minimum and maximum points of P are articulation points.

Fact 1.6. *If a poset P contains an articulation point, then P is an $\exists$-game.*

Proof. Let x be some articulation point of P. If x is a winning first move for Alice, then we are done. If x is a losing first move for Alice, then there must be some winning response y for Bob if Alice first plays x. But if Alice plays x, then all points $\geq x$ are now gone, and so we have $y < x$. This means that the game after Bob moves is P_y, which is a $\forall$-game by assumption. But then, Alice could have played y instead on her first move, leaving the $\forall$-game P_y for Bob, and thus winning. □

We call this "strategy stealing" because Alice steals Bob's winning strategy. The interesting thing about this proof is how nonconstructive it is. It shows that Alice has a winning first move, but gives virtually no information about what that first move could be. All we know is that the winning first play must be $\leq x$. If x is a maximum point of P, then the proof gives no information whatsoever about Alice's winning first move. Several poset games, including Chomp, have initial posets with maximum points, so we know that they are $\exists$-games. But determining a winning first move for Alice in Chomp appears quite difficult, and no fast algorithm is known. This suggests that, in the case of Chomp at least, the search problem (finding a winning first move) is apparently difficult, whereas the decision problem ($\exists$-game or $\forall$-game?) is trivial. The search versus decision issue is discussed further in Section 4.1, below.

Exercise 1.7. Show that the winning first moves in any poset form an antichain.

1.4 Black-white poset games

Many interesting games are not impartial because the legal moves differ for the players. In chess, for example, one player can only move white pieces and the other only black pieces. We will informally call a game "black-white" when each player is assigned a color (black or white) and can only make moves corresponding to their color.[2] Many impartial games have natural black-white versions. Here, then, is a black-white version of a poset game:

Definition 1.8. A *black-white poset game* consists of finite poset P, each of whose points are colored either black or white. The same rules apply to black-white poset games as to (impartial) poset games, except that one player (Black) can only play black points and the other player (White) can only play white points. (All points above a played point are still removed, regardless of color.)

One could generalize this definition by allowing a third color, grey, say, where grey points can be played by either player. We will not pursue this idea further.

[2]A different, popular color combination is red-blue. We use black-white so that illustrations are faithfully rendered on a black-and-white printer.

Other "colored" games include red-blue Hackenbush and red-green-blue Hackenbush [1].

Combinatorial games that are not impartial are known as *partisan*. In partisan games, we must make a distinction between the two players beyond who moves first. Generically, these players are called Left and Right. There is a surprisingly robust general theory of combinatorial games, both impartial and partisan, developed in [1, 5], and we give the basics of this theory in the next section.

2 Combinatorial game theory basics

In this section we give some relevant definitions and a few facts from the general theory of combinatorial games. We give enough of the theory to understand later results. Thorough treatments of this material, with lots of examples, can be found in [1, 5] as well as other sources, e.g., the recent book by Siegel [23]. Our terminology and notation vary a little bit from [1, 5], but the concepts are the same. When we say, "game," we always mean what is commonly referred to as a *combinatorial game*, i.e., a game between two players, say, Left and Right, alternating moves with perfect information, where the first player unable to move loses (and the other wins). In their fullest generality, these games can be defined abstractly by what options each player has to move, given any position in the game.

To save space, we will omit proofs of the results of this section, leaving them as exercises to the reader. These proofs are usually straightforward applications of previous results in this section, or induction, or both. The extended paper contains the full proofs.

2.1 Notation

We let $\mathbb{N}$ denote the set $\{0, 1, 2, \ldots, \}$ of natural numbers. We let $|X|$ denote the cardinality of a finite set X. We use the relation ":=" to mean "equals by definition." We extend the definition of an operator on games to an operator on sets of games in the customary way; for example, if $*$ is a binary operation on games, and G and H are sets of games, then $G * H := \{g * h \mid g \in G \wedge h \in H\}$, and if g is a game, then $g * H := \{g\} * H$, and so on.

2.2 Basic definitions

Definition 2.1. A *game* is an ordered pair $G = (G^L, G^R)$, where G^L and G^R are sets of games. The elements of G^L (respectively, G^R) are the *left options* (respectively, *right options*) of G. An *option* of G is either a left option or a right option of G.

It is customary to write $\{G^L|G^R\}$ or $\{\ell_1, \ell_2, \ldots | r_1, r_2, \ldots\}$ rather than (G^L, G^R), where $G^L = \{\ell_1, \ell_2, \ldots\}$ and $G^R = \{r_1, r_2, \ldots\}$. We will do the same.

For this and the following inductive definitions to make sense, we tacitly assume that the "option of" relation is well-founded, i.e., there is no infinite sequence of games $g_1, g_2, \ldots$ where g_{i+1} is an option of g_i for all i.[3] A *position* of a game G is any game reachable by making a finite series of moves starting with G (the moves need not alternate left-right). Formally,

Definition 2.2. A *position* of a game G is either G itself or a position of some option of G. We say that G is *finite* iff G has a finite number of positions.[4]

Starting with a game G, we imagine two players, Left and Right, alternating moves as follows: the initial position is G; given the current position P of G (also a game), the player whose turn it is chooses one of her or his options of P (left options for Left; right options for Right), and this option becomes the new game position. The first player faced with an empty set of options loses. The sequence of positions obtained this way is a *play* of the game G. Our well-foundedness assumption implies that every play is finite, and so there must be a winning strategy for one or the other player. We classify games by who wins (which may depend on who moves first) when the players play optimally. This is our broadest and most basic classification. Before giving it, we first introduce the "mirror image" of a game G: define $-G$ to be the game where all left options and right options are swapped at every position, as if the players switched places. Formally,

Definition 2.3. For any game G, define $-G := \{-G^R|-G^L\}$.

It is a good warm-up exercise to prove—inductively, of course—that $-(-G) = G$ for every game G. For impartial games, e.g., poset games, the "$-$" operator has no effect; for black-white poset games, this is tantamount to swapping the color of each point in the poset.

We can consider the following definition to be the most fundamental property of a game:

Definition 2.4. Let G be a game. We say that $G \geq 0$ (or $0 \leq G$) iff there is no right option g^R of G such that $-g^R \geq 0$. We will say $G \leq 0$ to mean that $-G \geq 0$.

So $G \geq 0$ if and only if no right option g^R of G satisfies $g^R \leq 0$. Symmetrically, $G \leq 0$ if and only if no left option g^L of G satisfies $g^L \geq 0$. In terms of strategies, $G \geq 0$ means that G is a *first-move loss for Right* or a *second-move win for Left*.

[3]This follows from the Foundation Axiom of set theory, provided ordered pairs are implemented in some standard way, e.g., $(x, y) := \{\{x\}, \{x, y\}\}$ for all sets x and y.

[4]Finite games are sometimes called *short games*; see [23].

If Right has to move first in G, then Left can win. Symmetrically, $G \leq 0$ means that G is a *first-move loss for Left* or a *second-move win for Right*.

The $\leq$ notation suggests that a partial order (or at least, a preorder) on games is lurking somewhere. This is true, and we develop it below.

Definition 2.4 allows us to partition all games into four broad categories.

Definition 2.5. Let G be a game.

- G is a *zero game* (or a *first-move loss*, or $\mathcal{P}$-game) iff $G \leq 0$ and $G \geq 0$.
- G is *positive* (or a *win for Left*, or $\mathcal{L}$-game) iff $G \geq 0$ and $G \not\leq 0$.
- G is *negative* (or a *win for Right*, or $\mathcal{R}$-game) iff $G \leq 0$ and $G \not\geq 0$.
- G is *fuzzy* (or a *first-move win*, or $\mathcal{N}$-game) iff $G \not\leq 0$ and $G \not\geq 0$.

These four categories, $\mathcal{P}$ (for previous player win), $\mathcal{L}$ (for Left win), $\mathcal{R}$ (for Right win), and $\mathcal{N}$ (for next player win), partition the class of all games. The unique category to which G belongs is called the *outcome* of G, written $o(G)$.

For example, the simplest game is the *endgame* $0 := \{|\}$ with no options, which is a zero game ($o(0) = \mathcal{P}$). The game $1 := \{0|\}$ is positive ($o(1) = \mathcal{L}$), and the game $-1 := \{|0\}$ is negative $o(-1) = \mathcal{R}$, while the game $* := \{0|0\}$ is fuzzy ($o(*) = \mathcal{N}$).

2.3 Game arithmetic, equivalence, and ordering

Games can be added, and this is a fundamental construction on games. The sum $G + H$ of two games G and H is the game where, on each move, a player may decide in which of the two games to play. Formally:

Definition 2.6. Let G and H be games. We define

$$G + H := \{(G^L + H) \cup (G + H^L) \mid (G^R + H) \cup (G + H^R)\} .$$

In Section 1 we used the + operator for the parallel union of posets. Observe that this corresponds exactly to the + operator on the corresponding games, i.e., the game corresponding to the parallel union of posets P and Q is the game-theoretic + applied to the corresponding poset *games* P and Q.

We write $G - H$ as shorthand for $G + (-H)$. One can easily show by induction that + is commutative and associative when applied to games, and the endgame 0 is the identity under +. This makes the class of all games into a commutative monoid (albeit a proper class). One can also show for all games G and H that $-(G + H) = -G - H$. Furthermore, if $G \geq 0$ and $H \geq 0$, then $G + H \geq 0$. It is *not* the case, however, that $G - G = 0$ for all G, although $G - G$ is always a zero game. These easy results are important enough that we state and prove them formally.

Lemma 2.7. *For any games G and H,*

1. *$G - G$ is a zero game.*
2. *Suppose $G \geq 0$. Then $H \geq 0$ implies $G + H \geq 0$, and $H \not\leq 0$ implies $G + H \not\leq 0$.*
3. *Suppose $G \leq 0$. Then $H \leq 0$ implies $G + H \leq 0$, and $H \not\geq 0$ implies $G + H \not\geq 0$.*
4. *$-(G + H) = -G - H$.*

The outcome $o(G)$ of a game G is certainly the first question to be asked about G, but it leaves out a lot of other important information about G. It does not determine, for example, the outcome when G is added to a fixed game X. That is, it may be that two games G and H have the same outcome, but $o(G+X) \neq o(H+X)$ for some game X. Indeed, defining $2 := \{1|\}$, one can check that $o(1) = o(2) = \mathcal{L}$, but we have $o(2-1) = \mathcal{L}$ (left wins by choosing $1 \in 2^L$ when she gets the chance), whereas we know already from Lemma 2.7 that $o(1 - 1) = \mathcal{P}$.

Behavior under addition leads us to a finer classification of games.

Definition 2.8. Let G and H be games. We say that G and H are *equivalent*, written $G \approx H$, iff $o(G + X) = o(H + X)$ for all games X.[5]

It follows immediately from the definition that $\approx$ is an equivalence relation on games, and we call the equivalence classes *game values*. We let **PG** denote the Class[6] of all game values.[7] Letting X be the endgame 0 in the definition shows that equivalent games have the same outcome. Using the associativity of $+$, we also get that $G \approx H$ implies $G + X \approx H + X$ for any game X. Thus $+$ respects equivalence and naturally lifts to a commutative and associative Operation (also denoted $+$) on **PG**.

The remaining goal of this subsection is finish showing that $\langle \mathbf{PG}, +, \leq \rangle$ is a partially ordered abelian Group. We have built up enough basic machinery that we can accomplish our goal in a direct, arithmetic way, without referring to players' strategies.

[5] In much of the literature, the overloaded equality symbol = is used for game equivalence. We avoid that practice here, preferring to reserve = for set theoretic equality. There are some important game properties that are not $\approx$-invariant.

[6] We will start to capitalize words that describe proper classes.

[7] Since each game value itself is a proper Class, we really cannot consider it as a member of anything. A standard fix for this in set theory is to represent each game value v by the *set* of elements of v with minimum rank, so **PG** becomes the Class of all such sets.

Lemma 2.9. *A game G is a zero game if and only if $G + H \approx H$ for all games H.*

Corollary 2.10. *A game G is a zero game if and only if $G \approx 0$ (where 0 is the endgame).*

Here is our promised Preorder on games.

Definition 2.11. Let G and H be games. We write $G \leq H$ (or $H \geq G$) to mean $H - G \geq 0$ (equivalently, $G - H \leq 0$). As usual, we write $G < H$ to mean $G \leq H$ and $H \not\leq G$.[8]

You can interpret $G < H$ informally as meaning that H is more preferable a position for Left than G, or that G is more preferable for Right than H. For example, if Left is ever faced with moving in position G, and (let us pretend) she had the option of replacing G with H beforehand, she always wants to do so.

Proposition 2.12. *The $\leq$ Relation on games is reflexive and transitive.*

Proposition 2.13. *For any two games G and H, $G \approx H$ if and only if $G - H$ is a zero game, if and only if $G \leq H$ and $G \geq H$.*

The last two propositions show that the binary Relation $\leq$ on games is a Preorder that induces a partial Order on **PG**. Proposition 2.13 also gives a good working criterion for proving or disproving game equivalence—just check whether $G - H$ is a second player win—without having to quantify over all games.

Proposition 2.14. *$\langle \textbf{\textit{PG}}, + \rangle$ is an abelian Group, where the identity element is the $\approx$-equivalence class of zero games, and inverses are obtained by the negation Operator on games.*

Finally, $\leq$ is translation-invariant on **PG**, making it a partially ordered abelian Group:

Corollary 2.15. *For any games G, H, and X, if $G \leq H$ then $G + X \leq H + X$.*

We next look at two important subclasses of games—the numeric games and the impartial games.

[8]We now have two ways of interpreting the expression "$G \geq 0$": one using Definition 2.4 directly and the other using Definition 2.11 with 0 being the endgame. One readily checks that the two interpretations coincide.

2.4 Numeric games

A numeric game is one where at each position all the left options are < all the right options. Formally,

Definition 2.16. A game G is *numeric* iff $\ell < r$ for every $\ell \in G^L$ and $r \in G^R$, and further, every option of G is numeric.

One can show that G is numeric if and only if $\ell < G$ for every $\ell \in G^L$ and $G < r$ for every $r \in G^R$. If H is also numeric, then either $G \leq H$ or $H \leq G$. The + and − operations also yield numeric games when applied to numeric games.[9] Numeric games have a peculiar property: making a move only worsens your position (for Left this means having to choose a smaller game; for Right, having to choose a larger game). Thus neither player wants to make a move—if they were given the option to skip a turn, they would always take it. For these games, an optimal play is easy to describe: Left always chooses a maximum left option (i.e., one that does the least damage), and Right always chooses a minimum right option, assuming these options exist.[10] This intuitive idea is formalized in the following theorem, which is referred to in the literature as the "dominating rule." It applies to all games, not just numeric games.

Theorem 2.17. *Let G be a game. If $y \leq \ell$ for some $\ell \in G^L$, then $G \approx \{y, G^L | G^R\}$. Similarly, if $y \geq r$ for some $r \in G^R$, then $G \approx \{G^L | G^R, y\}$.*

If $y \leq \ell \in G^R$, then we say that y is *dominated* by ℓ in G. Similarly, if $y \geq r \in G^R$, then y is *dominated* by r in G. We obtain equivalent games by removing dominated options. A player never needs to play a dominated option; it is just as well (or better) to choose an option that dominates it.

Numeric games are called such because their values act like real numbers; for one thing, their values are totally ordered by $\leq$. These games are constructed in a way somewhat akin to how the real numbers are constructed from the rationals via Dedekind cuts. The left options of a game form the left cut, the right options the right cut, and the game itself represents a number strictly between the two. The differences are that the two cuts might be bounded away from each other (one or the other may even be empty), and the left cut might contain a maximum element.

2.4.1 Finite numeric games

The values of *finite* numeric games form a subgroup of **PG** naturally isomorphic (in an order-preserving way) to the dyadic rational numbers under addition, ac-

[9]The property of being numeric is *not* invariant under $\approx$. One can easily concoct two equivalent games, one of which is numeric and the other not.

[10]In general, Left can win by choosing any option $\ell \geq 0$, and Right can win by choosing any option $r \leq 0$.

cording to the following “simplicity rule”:

Definition 2.18. Let G be a finite numeric game. The *(numerical) value* of G, denoted $v(G)$, is the unique rational number $a/2^k$ such that

1. k is the least nonnegative integer such that there exists an integer a such that $v(\ell) < a/2^k$ for all $\ell \in G^L$ and $a/2^k < v(r)$ for all $r \in G^R$, and

2. a is the integer with the least absolute value satisfying (1.) above.

So for example, the endgame 0 has value $v(0) = 0$, the game 1 has value $v(1) = 1$, and the game -1 has value $v(-1) = -1$, as the notation suggests. Intuitively, $|v(G)|$ indicates the number of “free moves” one of the players has before losing (Left if $v(G) > 0$, and Right if $v(G) < 0$). In fact, for any two finite numeric games P and Q, one can show that $v(P+Q) = v(P)+v(Q)$ and that $v(-P) = -v(P)$. Also, $P \leq Q$ if and only if $v(P) \leq v(Q)$.[11] The valuation map v is not one-to-one on games, but induces a one-to-one map on *values* of numeric games.

To illustrate the simplicity rule, consider the game $h := \{0|1\}$. The rule says that $v(h)$ is the simplest dyadic rational number strictly between 0 and 1, namely, $1/2$. First note that Left can always win h whether or not she plays first, so $h > 0$. If v respects $+$, then we should also have $h + h \approx 1$. Let us check this. First consider $1 - h$:

$$\begin{aligned} 1 - h &= 1 + (-h) = \{0|\} + \{-1|0\} = \{0 - h, 1 - 1|1 + 0\} \\ &= \{-h, 0|1\} \approx \{0|1\} = h \end{aligned}$$

(the equivalence is by the dominating rule and $-h < 0$). Thus

$$h + h \approx h + (1 - h) \approx 1 \ .$$

Black-white poset games are numeric [10]. Here we identify Black with Left and White with Right. So for example, an antichain of k black points has numeric value k, and an antichain of k white nodes has numeric value $-k$. Figure 2 shows the numeric value of two simple, two-level black-white poset games.

Exercise 2.19. Use the simplicity rule to prove the values in Figure 2.

The numerical values of arbitrary numeric games (not necessarily finite) form an ordered, real-closed field **No** into which the real numbers embed, but which also contains all the ordinals as well as infinitesimals [5]. Donald Knuth dubbed **No** the *surreal numbers* [17], and they are formed via a transfinite construction. The dyadic rationals are those constructed at finite stages, but numbers constructed through stage ω already form a proper superset of $\mathbb{R}$.

[11]One can define a purely game-theoretic multiplication operation on numeric games in such a way that $v(PQ) = v(P)v(Q)$ for all P and Q. See [5] for details.

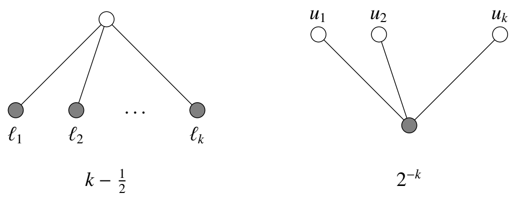


Figure 2: The numerical values of two simple black-white poset games. The left has value $k - \frac{1}{2}$ and the right has value 2^{-k}, for $k \geq 1$.

2.5 Impartial games and Sprague-Grundy theory

A game is *impartial* if at every position, the two players have the same options. Formally,

Definition 2.20. A game G is *impartial* iff $G^L = G^R$ and every $g \in G^L$ is impartial.

Equivalently, G is impartial if and only if $G = -G$. This means that values of impartial games are those that have order two in the group $\langle \mathbf{PG}, + \rangle$.

Examples of impartial games include 0 and $*$. Families of impartial games include Nim, Geography, Node Kayles, and poset games.[12] There is a beautiful theory of impartial games, developed by R. P. Sprague and P. M. Grundy [25, 14] that predates the more general theory of combinatorial games described in [1, 5]. We develop the basics of this older theory here. First note that, since there are no Left/Right biases, all impartial games are either zero ($\mathcal{P}$) or fuzzy ($\mathcal{N}$), and we can assume that Left always moves first. We will call impartial zero games $\forall$-*games* ("for all first moves ...") and impartial fuzzy games $\exists$-*games* ("there exists a first move such that ..."). In this section only, we restrict our attention to impartial games, so when we say "game," we mean impartial game.

Two (impartial) games G and H are equivalent ($G \approx H$) if and only if $G + H$ is a $\forall$-game, because $H = -H$ (Sprague and Grundy defined this notion for impartial games). One can associate an ordinal number with each game, which we call the *g-number*[13] of the game, such that two games are equivalent if and only if they have the same g-number. The g-number of a finite game is a natural number. We will restrict ourselves to finite games.

Definition 2.21. Let A be any coinfinite subset of $\mathbb{N}$. Define mex A (the *$\underline{m}$inimum*

[12]Impartiality is not $\approx$-invariant.

[13]also called the *Grundy number* or the *NIM number*—not to be confused with the value of a numerical game

excluded element from A) to be the least natural number not in A, i.e.,

$$\operatorname{mex} A := \min(\mathbb{N} - A) \ .$$

More generally, for $i = 0, 1, 2, \ldots$, inductively define

$$\operatorname{mex}_i A := \min\left(\mathbb{N} - (A \cup \{\operatorname{mex}_0(A), \ldots, \operatorname{mex}_{i-1} A\})\right) \ ,$$

the i'th least natural number not in A. (So in particular, $\operatorname{mex}_0 A = \operatorname{mex} A$.)

Definition 2.22. Let G be any (finite) game. Define the *g-number* of G as

$$g(G) := \operatorname{mex} \textit{g-set}(G) \ ,$$

where $\textit{g-set}(G) := \{g(x) \mid x \in G^L\}$ is called the *g-set* of G.

That is, $g(G)$ is the least natural number that is not the g-number of any option of G, and the set of g-numbers of options of G is $\textit{g-set}(G)$. For example, $\textit{g-set}(0) = \emptyset$, and so $g(0) = 0$. Also, $\textit{g-set}(*) = \{g(0)\} = \{0\}$, and so $g(*) = 1$.

Exercise 2.23. Prove the following for any finite poset P and any $n \in \mathbb{N}$.

1. $g(P) \le |P|$. (Generally, $g(G) \le \left|G^L\right|$ for any impartial G.)
2. $g(C_n) = n$ for all $n \in \mathbb{N}$.
3. $g(A_n) = n \bmod 2$.
4. $g(V_n) = (n \bmod 2) + 1$.

What is $g(\Lambda_n)$? What is $g(\Diamond_n)$?

Exercise 2.24. Describe $g(A_m/A_n)$ simply in terms of m and n.

Here is the connection between the g-number and the outcome of a game.

Proposition 2.25. *A game G is a $\forall$-game if and only if $g(G) = 0$.*

Proof idea. If $g(G) \neq 0$, then there is some option x of G that Left can play such that $g(x) = 0$, but if $g(G) = 0$, then no move Left makes can keep the g-number at 0. □

The central theorem of Sprague-Grundy theory—an amazing theorem with a completely nonintuitive proof—concerns the g-number of the sum of two games.

Definition 2.26. For any $m, n \in \mathbb{N}$, define $m \oplus n$ to be the natural number k whose binary representation is the bitwise exclusive OR of the binary representations of m and n. We may also call k the *bitwise XOR* of m and n.

For example, $23 \oplus 13 = 10111 \oplus 01101 = 11010 = 26$.

Theorem 2.27 (Sprague, Grundy [25, 14]). *For any finite games G and H,*

$$g(G + H) = g(G) \oplus g(H) .$$

Corollary 2.28. *Two impartial games G and H are equivalent if and only if $g(G) = g(H)$.*

Proof. G and H are equivalent iff $G + H$ is a $\forall$-game, iff $g(G + H) = 0$ (Proposition 2.25), iff $g(G) \oplus g(H) = 0$ (Theorem 2.27), iff $g(G) = g(H)$. □

Since every natural number n is the g-number of the poset game C_n, this means that every game is equivalent to a single NIM stack.

We can use Theorem 2.27 to solve Nim. Given a Nim game $P = C_{n_1} + \cdots + C_{n_k}$, we get $g(P) = n_1 \oplus \cdots \oplus n_k$. If this number is nonzero, then let i be largest such that $(g(P))_i = 1$. Alice can win by choosing a j such that $(n_j)_i = 1$ and playing in C_{n_j} to reduce its length (and hence its g-number) from n_j to $n_j \oplus (g(P))_i$. This makes the g-number of the whole Nim game zero.

Theorem 2.27 shows how the g-number behaves under parallel unions of posets (Definition 1.2). How does the g-number behave under series unions? Unfortunately, $g(P/Q)$ might not depend solely on $g(P)$ and $g(Q)$. For example, $g(V_2) = g(C_1) = 1$, but $g(C_1/V_2) = g(\Diamond_2) = 3$ whereas $g(C_1/C_1) = g(C_2) = 2$. However, *g-set*(P/Q) *does* depend solely on *g-set*(P) and *g-set*(Q) for any posets P and Q, and this fact forms the basis of the Deuber & Thomassé algorithm of the next section.

There is one important case where $g(P/Q)$ does only depend on $g(P)$ and $g(Q)$:

Fact 2.29. *For any finite poset P and any $k \geq 0$,*

$$g\left(\frac{P}{C_k}\right) = g(P) + k .$$

This can shown by first showing that $g(P/C_1) = g(P) + 1$, then using induction on k. By Fact 2.29, we get that $g(\Diamond_n) = 1 + g(\Lambda_n)$ for example.

3 Upper bounds

When asking about the computational difficulty of determining the outcome of a game, we really mean a family of similar games, represented in some way as finite inputs. In discussing game complexity, we will abuse terminology and refer to a family of games simply as a game. (The same abuse occurs in other areas of complexity, notably circuit complexity.) We will also use the same small-caps notation to refer both to a family of games and to the corresponding decision problem about the outcomes.

Perhaps the most common upper bound in the literature on the complexity of a game is membership in **PSPACE**. Without pursuing it further, we will just mention that, if a game G of size n satisfies: (i) every position of G has size polynomial in n; (ii) the length of any play of G is polynomial in n; and (iii) there are polynomial-time (or even just polynomial-space) algorithms computing the "left option of" and "right option of" relations on the positions of G, then $o(G)$ can be computed in polynomial space. These properties are shared by many, many games.

In this section we will give some better upper bounds on some classes of finite poset games, the best one being that N-free poset games are in **P** [6]. We will assume that a poset is represented by its Hasse diagram, a directed acyclic graph (DAG) in which each element is represented as a node and an arc is placed from a node for element x to the node for y when $x < y$ and there is no element z such that $x < z < y$. The poset is the reflexive, transitive closure of the edge relation of the DAG.

3.1 N-free games

With the Hasse diagram representation, we can apply results from graph theory to devise efficient ways to calculate g-numbers for certain classes of games. A good example is the class of N-free poset games. An "N" in a poset is a set of four elements $\{a, b, c, d\}$ such that $a < b$, $c < d$, $c < b$, and the three other pairs are incomparable. When drawn as a Hasse diagram the arcs indicating comparability form the letter "N". A poset is *N-free* if it contains no N as an induced subposet. We let N-Free denote the class of N-free poset games.

Valdes, Tarjan, and Lawler [30] show that an N-free DAG can be constructed in linear time from a set of single nodes. New components are created either by applying parallel union ($G+H$) or by applying series union (G/H). As with posets, the parallel union is the disjoint union of G and H. The series union is a single DAG formed by giving to every element in H with out-degree 0 (the sinks in H) an arc to every element in G with in-degree 0 (the sources in G). This gives the Hasse diagram of the series union of the corresponding posets. Their algorithm provides a sequence of + and / operations that will construct a given N-free DAG from single points.

Deuber & Thomassé [6] show that N-Free $\in$ **P** by applying this construction to demonstrate how to calculate the g-number of an N-free poset game based on the sequence of construction steps obtained by the VTL algorithm above. Their algorithm, which we now describe, works by keeping track of the g-sets of the posets obtained in the intermediate steps of the construction, rather than the g-numbers. There is no need to store the g-numbers, because the g-number of any poset can always be easily computed from its g-set by taking the mex.

The g-number of a single node is 1. This is the base case.

Fact 3.1. *Given posets P and Q, the g-set of the parallel union $P + Q$ is*

$$\begin{aligned} \textit{g-set}(P+Q) &= \{g(P+Q_q) : q \in Q\} \cup \{g(P_p+Q) : p \in P\} \\ &= \{g(P) \oplus g(Q_q) : q \in Q\} \cup \{g(P_p) \oplus g(Q) : p \in P\} . \end{aligned}$$

The second equality follows from the Sprague-Grundy theorem. This is easy to see if you consider the root of the game tree for $P + Q$. Each of its children results from playing either an element in P or one in Q. The left-hand set in the union contains the g-numbers of the games resulting from playing an element in Q; the right-hand set from playing an element in P. Their union is the g-set of $P + Q$, so its g-number is the mex of that set.

To calculate the g-set of a series union, we will need the definition of the *Grundy product* of two finite sets of natural numbers:

$$A \odot B := B \cup \{\operatorname{mex}_a B \mid a \in A\} .$$

$A \odot B$ is again a finite set of natural numbers that is easy to compute given A and B. Basically, $A \odot B$ unions B with the version of A we get after re-indexing the natural numbers to go "around" B. Notice that $\operatorname{mex}(A \odot B) = \operatorname{mex}_{\operatorname{mex} A} B$. We will use this fact below.

Lemma 3.2 (Deuber & Thomassé [6]). *For any finite posets P and Q, $\textit{g-set}(P/Q) = \textit{g-set}(P) \odot \textit{g-set}(Q) = \textit{g-set}(Q) \cup \{\operatorname{mex}_i(\textit{g-set}(Q)) : i \in \textit{g-set}(P)\}$.*

The left-hand set of the union results from playing an element in Q, which removes all of the elements in P. Using induction, we can see what happens when an element in P is played.

Proof of Lemma 3.2. The fourth equality uses the inductive hypothesis.

$$\begin{aligned} \textit{g-set}(P/Q) &= \{g((P/Q)_r) : r \in P/Q\} \\ &= \{g((P/Q)_p) : p \in P\} \cup \{g((P/Q)_q) : q \in Q\} \\ &= \{g((P_p/Q)) : p \in P\} \cup \{g(Q_q) : q \in Q\} \\ &= \{\operatorname{mex}(\textit{g-set}(P_p) \odot \textit{g-set}(Q)) : p \in P\} \cup \textit{g-set}(Q) \\ &= \{\operatorname{mex}_{\operatorname{mex} \textit{g-set}(P_p)} \textit{g-set}(Q) : p \in P\} \cup \textit{g-set}(Q) \\ &= \{\operatorname{mex}_{g(P_p)}(\textit{g-set}(Q)) : p \in P\} \cup \textit{g-set}(Q) \\ &= \{\operatorname{mex}_i(\textit{g-set}(Q)) : i \in \textit{g-set}(P)\} \cup \textit{g-set}(Q) \\ &= \textit{g-set}(P) \odot \textit{g-set}(Q) \end{aligned}$$

□

In particular, the g-number of P/Q is greater than or equal to the sum of the g-numbers of P and Q. Notably, it's an equality if Q is C_n for some n (Fact 2.29) and the reason is that the g-set of C_n has no gaps, that is, it contains all of the values from 0 to $n - 1$. It's easy to see that it's true when P and Q are both singletons. Their g-numbers are both 1 and forming their series-union creates a NIM stack of size 2 and that has g-number 2.

Another way to understand Lemma 3.2 is to consider the game tree of P/Q, and we'll look at the simple case where P is an arbitrary game with g-number k and Q is a singleton. Consider the root node r of the game tree of P/Q. One of its children represents playing the single element in Q and that child has g-number 0. The rest of r's children represent game configurations reached by playing an element in P. By the induction hypothesis the g-number of each of these nodes will be one more than in P's game tree where they had g-numbers 0 to $k - 1$, and perhaps g-numbers $k + 1$ and larger. So in P/Q's tree they have g-numbers 1 to k, with perhaps g-numbers $k + 2$ or larger. Because the child reached by playing Q's single element has g-number 0, the first missing value in the g-set formed from these g-numbers is $k + 1$.

Now using Fact 3.1 and Lemma 3.2, the decomposition described in [30] can generate a binary tree where each internal node is labeled with a poset P and an operation (parallel union or series union), and its children are the two posets combined to form P. Starting with each leaf, where the poset is a singleton and the g-set is $\{0\}$, and moving up the tree, one can apply Fact 3.1 and Lemma 3.2 to compute the g-set of the root (and none of the g-numbers involved exceed the size of the final poset). This can all be done in time $O(n^4)$.

3.2 Results on some classes of games with N's

General results for classes of games containing an "N" have been few. In 2003, Steven Byrnes [3] proved a poset game periodicity theorem, which applies to, among others, Chomp-like games, which contain many "N"-configurations.

Here's the theorem, essentially as stated in the paper:

Theorem 3.3. *In an infinite poset game X, suppose we have two infinite chains C ($c_1 < c_2 < \cdots$) and D ($d_1 < d_2 < \cdots$), and a finite subset A, all pairwise disjoint, and assume that no element of C is less than an element of D. Let $A_{m,n} = A \cup C \cup D - \{x \in X | x \geq c_{m+1}\} - \{x \in X | x \geq d_{n+1}\}$ (that is, $A_{m,n}$ is the position that results from starting with the poset $A \cup C \cup D$, then making the two moves c_{m+1} and d_{n+1}). Let k be a nonnegative integer. Then either:*

1. *there are only finitely many different $A_{m,n}$ with g-number k; or*

2. *we can find a positive integer p such that, for large enough n, $g(A_{m,n}) = k$ if and only if $g(A_{m+p,n+p}) = k$.*

Thus, as the poset A expands along the chains C and D, positions with any fixed g-number have a regular structure.

A simple example of a class of games covered by the theorem is the family of two-stack Nim games, where A is empty and $A_{m,n}$ consists of an m-chain and an n-chain. The g-number 0 occurs for every $A_{n,n}$ so the periodicity is 1. The g-number 1 occurs for every $A_{2n,2n+1}$ and so has periodicity 2. In fact, one can find a periodic repetition for every g-number. The surprising thing is that this is still true when you allow elements in one chain to be less than elements in the other.

Another family contains Chomp, described in Section 1.1.1. We can generalize Chomp to games where the rows do not have to contain the same number of elements. Byrnes showed that for such games there is a periodicity in the g-numbers when we fix the size of all but the top two rows.

As Byrnes claims, this yields a polynomial-time decision algorithm for each family generated from a fixed A but not a uniformly polynomial-time algorithm across the families, as the time is parameterized by A.

3.2.1 Bounded-width poset games

If a poset P has width k, that is, if k is the maximum size of any antichain in P, then there are only $|P|^k$ many positions at most in the game: if $x_0, x_1, \ldots, x_{n-1} \in P$ are the elements chosen by the players in the first n moves of the game, then the resulting position is completely determined by the minimal elements of the set $\{x_0, \ldots, x_{n-1}\}$, i.e., an antichain of size $\leq k$.

This means that, for constant k, one can compute the g-number of P in polynomial time using dynamic programming. The exponent on the running time depends on k, however. For certain families of bounded-width posets, one can beat the time of the dynamic programming algorithm; for example, one can compute the g-number of width-2 games in linear time.

4 Lower bounds

In this section we give some lower bounds on game complexity. There is a vast literature on combinatorial game complexity, and we make no attempt to be thorough, but rather concentrate on poset game complexity.

4.1 A note about representations of games

The complexity of a game depends quite a bit on its representation. The choice of representation is usually straightforward, but not always. For example, how should we represent an N-free poset? Just by its Hasse diagram, or by an expression for the poset in terms of single points and parallel union and series union operators? The results of Valdes, et al. [30] show that one representation can be converted into the other in polynomial time, so the choice of representation is not an issue unless we want to consider complexity classes within **P** or more succinct representations of posets, as we will do below. There, fortunately, our hardness results apply to either representation.

Even if the representation of a game is clear, the results may be counterintuitive. For example, how should we represent members of the class of *all* finite games? In Section 2, we defined a game as an ordered pair of its left and right options. We must then represent the options, and the options of options, and so on. In effect, to represent an arbitrary finite game explicitly, we must give its entire game tree (actually, game DAG, since different sequences of moves may end up in the same position). Under this representation, there is a straightforward algorithm to compute the outcome of any game: use dynamic programming to find the outcome of every position in the game. Since every position is encoded in the string representing the game, this algorithm runs in polynomial time.

What makes a game hard, then, is that we have a succinct representation for it that does not apply to all games. For example, the obvious representation of a poset game is the poset itself, and the number of positions is typically exponential in the size of the poset. Subfamilies of poset games may have even more succinct representations. For example, a Nim game can be represented as a finite list of natural numbers in binary, giving the sizes of the stacks, and a game of Chomp can be represented with just two natural numbers m and n in binary, giving the dimensions of the grid. Notice that this Chomp representation is significantly shorter than what is needed to represent an arbitrary *position* in a Chomp game; the latter is polynomial in $m + n$.

In what sense does finding a winning strategy in Chomp reduce to determining the outcome of Chomp games? We already know that every Chomp game is an $\exists$-game because it has a maximal point. We could find a winning strategy if we were able to determine the outcome of every Chomp position, but even writing down a query to an "outcome oracle" takes time linear in $m + n$, which is exponential in the input size. The more modest goal of finding a winning first move may be more feasible, because the position after one move is simple enough to describe by a polynomial-length query string. To our knowledge, no efficient algorithm is known to determine the outcome of an arbitrary Chomp position after a single move, even allowing time $(m + n)^{O(1)}$.

We will more to say about representations below when we discuss lower bounds for poset games within the complexity class **P**.

4.2 Some PSPACE-hard games

Many games have been shown **PSPACE**-hard over the years. Early on, Even and Tarjan showed that Hex generalized to arbitrary graphs is **PSPACE**-complete [7]. A typical proof of **PSPACE**-hardness reduces the **PSPACE**-complete True Quantified Boolean Formulas (TQBF [26]) problem to the outcome of a game. We can consider a quantified Boolean formula $\varphi = (\exists x_1)(\forall x_2)\cdots\psi$ (where ψ is a Boolean formula in conjunctive normal form (cnf)) itself as a game, where players alternate choosing truth values for $x_1, x_2, \ldots$, the first player (Right, say) winning if the resulting instantiation of ψ is true, and Left winning otherwise.[14]

TQBF seems ideal for encoding into other games. Thomas Schaefer showed a number of interesting games to be **PSPACE**-hard this way [21]. One interesting variant of TQBF that Schaefer proved **PSPACE**-complete is the game where a positive Boolean formula ψ is in cnf with no negations, and players alternate choosing truth values for the Boolean variables. Schaefer called this game G_{pos}(POS CNF). Unlike TQBF, however, the variables need not be chosen in order; players may choose to assign a truth value to any unassigned variable on any move. Left (who moves first) wins if ψ is true after all variables have been chosen, and Right wins otherwise. Since ψ is positive, Left always wants to set variables to 1 and Right to 0.

As another example, consider Geography. The input is a directed graph G and a designated vertex s of G on which a token initially rests. The two players alternate moving the token on G from one node to a neighboring node, trying to force the opponent to move to a node that has already been visited. Geography is a well-known **PSPACE**-complete game [21, 24]. In [19], Lichtenstein & Sipser show that Geography is **PSPACE**-complete even for bipartite graphs.

An obvious way to turn Geography into a black-white game is to color the nodes of graph G black and white. Each player is then only allowed to move the token to a node of their own color. Since moves are allowed only to neighboring nodes, the black-white version is equivalent to the uncolored version on bipartite graphs. The standard method of showing that Geography is **PSPACE**-complete is via a reduction from True Quantified Boolean Formulas (TQBF) to Geography (see for example [24]). Observe that the graph constructed in this reduction is not bipartite. That is, there are nodes that potentially may be played by both players. Hence, we cannot directly conclude that the black-white version is **PSPACE**-

[14]This is technically not a combinatorial game by our definition, because the end condition is different. One can modify the game slightly to make it fit our definition, however.

complete. However, in [19] Lichtenstein & Sipser show that Geography is indeed **PSPACE**-complete for bipartite graphs.

We now consider the game Node Kayles. This game is defined on an undirected graph G. The players alternately play an arbitrary node from G. In one move, playing node v removes v and all the direct neighbors of v from G. In the black-white version of the game, we color the nodes black and white. Schaefer [21] showed that determining the winner of an arbitrary Node Kayles instance is **PSPACE**-complete. He also extended the reduction to bipartite graphs, which automatically yields a reduction to the black-white version of the game (see [12]). Therefore, black-white Node Kayles is also **PSPACE**-complete.

The game of Col [1] is a two-player combinatorial strategy game played on a simple planar graph, some of whose vertices may be colored black or white. During the game, the players alternate coloring the uncolored vertices of the graph. One player colors vertices white and the other player colors vertices black. A player is not allowed to color a vertex neighboring another vertex of the same color. The first player unable to color a vertex loses. A well-known theorem about Col is that the value of any game is either x or $x + *$ where x is a number. Removing the restriction that Col games be played on planar graphs and considering only those games in which no vertex is already colored, we get a new game, GenCol (generalized Col). It is shown in [10] that GenCol is **PSPACE**-complete; furthermore, GenCol games only assume the two very simply game values 0 and $*$.

In [20], Stockmeyer & Chandra give examples of games that are complete for exponential time and thus provably infeasible.

4.3 Lower bounds for poset games

Until recently, virtually no hardness results were known relating to poset games, and the question of the complexity of determining the outcome of a game was wide open, save the easy observation that it is in **PSPACE**.

For the moment, let PG informally denote the decision problem of determining the outcome of a arbitrary given (impartial) poset game, that is, whether or not the first player (Alice) can win the game with perfect play. The first lower bound on the complexity of PG we are aware of, and it is a modest one, was proved by Fabian Wagner [31] in 2009. He showed that PG is **L**-hard[15] under **FO**-reductions (First-Order reductions). This is enough to show, for example, that PG $\notin$ $\mathbf{AC}^0$. Soon after, Thomas Thierauf [28] showed that PG is hard for **NL** under $\mathbf{AC}^0$ reductions.[16] A breakthrough came in 2010, when Adam Kalinich, then a high

[15]**L** is short for LOGSPACE.

[16]**NL** is nondeterministic LOGSPACE.

school student near Chicago, Illinois, showed that PG is hard for $\mathbf{NC}^1$ under $\mathbf{AC}^0$ reductions [16]. For the proof, he invents a clever way to obliviously "flip" the outcome of a game, i.e., to toggle the outcome between $\exists$ and $\forall$. This allows for the simulation of a NOT-gate in an $\mathbf{NC}^1$ circuit. (An OR-gate can be simulated by the series union construction of Definition 1.2. See below.)

The astute reader will notice that Kalinich's result appears to be weaker than the other two earlier results. In fact, the three results are actually incomparable with each other, because they make different assumptions about how poset games are represented as inputs. We say more about this below, but first we mention that Wagner's and Thierauf's results both hold even when restricted to Nim games with two stacks, and Kalinich's result holds restricted to N-free games. Modest as they are, these are currently the best lower bound we know of for N-free poset games.

Very recently, the complexity of PG was settled completely by Daniel Grier, an undergraduate at the University of South Carolina [13]. He showed that PG is **PSPACE**-complete via a polynomial reduction (henceforth, p-reduction) from Node Kayles. Here, it is not important how a game is represented as an input, so long as the encoding is reasonable. His proof shows that **PSPACE**-completeness is still true when restricted to three-level games, i.e., posets where every chain has size at most three (equivalently, posets that are partitionable into at most three antichains). The games used in the reduction are of course not N-free.

4.4 Representing posets as input

As we discussed above, for any of the various well-studied families of poset games (Chomp, Divisors, Nim, etc.), there is usually an obvious and natural way to represent a game as input. For example, an instance of Chomp can be given with just two positive integers, one positive integer for Divisors, and a finite list of positive integers for Nim, giving the heights of the stacks. When considering arbitrary finite posets, however, there is no single natural way to represent a poset as input, but rather a handful of possibilities, and these may affect the complexity of various types of poset games. We consider two broad genres of poset representation:

Explicit The poset is represented by an explicit data structure, including the set of points and the relations between them. In this representation, the size of the poset is always comparable to the size of the input.

Succinct (Implicit) The poset is represented by a Boolean circuit with two n-bit inputs. The inputs to the circuit uniquely represent the points of the poset, and the (1-bit) output gives the binary relation between these two inputs. In this representation, the size of the poset can be exponential in the size of the circuit.

Within each representational genre, we will consider three general approaches to encoding a poset P, in order from “easiest to work with” to “hardest to work with”:

Partial Order (PO) P is given as a reflexive, transitive, directed acyclic graph, where there is an edge from x to y iff $x \leq y$.

Hasse Diagram (HD) P is given as a directed acyclic graph whose reflexive, transitive closure (i.e., reachability relation) is the ordering $\leq$. The graph then gives the Hasse diagram of P.

Arbitrary (binary) Relation (AR) An arbitrary directed graph (or arbitrary binary relation) is given, whose reflexive, transitive closure is then a pre-order whose induced partial order is P. (Equivalently, P is the set of strongly connected components, and $\leq$ is the reachability relation between these components.)

The first two (PO and HD) must involve promises that the input satisfies the corresponding constraint, so problems in these categories are posed as promise problems. Notice that the PO promise is stronger than the HD promise, which is stronger than the AR (vacuous) promise. So in either the Explicit or Succinct cases, the complexity of the corresponding problems increases monotonically as PO $\rightarrow$ HD $\rightarrow$ AR.

We will ignore some additional subtleties: In the explicit case, is the graph (or relation) given by an adjacency matrix or an array of edge lists? In the succinct case, should we be able to represent a poset whose size is not a power of 2? For example, should we insist on including a second circuit that tells us whether a given binary string represents a point in the poset? These questions can generally be finessed, and they do not affect any of the results.

4.5 The decision problems

The two genres and three approaches above can be combined to give six versions of the basic decision problem for arbitrary posets: the three explicit problems PG(Explicit, PO), PG(Explicit, HD), and PG(Explicit, AR); and the three succinct problems PG(Succinct, PO), PG(Succinct, HD), and PG(Succinct, AR). We will define just a couple of these, the others being defined analogously.

Definition 4.1. PG(Succinct, HD) is the following promise problem:

Input: A Boolean circuit C with one output and two inputs of n bits each, for some n.

Promise: G is acyclic, where G is the digraph on $\{0, 1\}^n$ whose edge relation is computed by C.

Question: Letting P be the poset given by the reachability relation on G, is P an $\exists$-game?

Definition 4.2. PG(Explicit, AR) is the following promise problem:

Input: A digraph G on n nodes.
Promise: None.
Question: Letting P be the poset given by the reachability relation on the strongly connected components of G, is P an $\exists$-game?

We also can denote subcategories of poset games the same way. For example, Nim(Explicit, HD) is the same as PG(Explicit, HD), but with the additional promise that the poset is a parallel union of chains; for any $k > 0$, Nim_k(Explicit, HD) is the same as Nim(Explicit, HD) but with the additional promise that there are at most k chains; N-Free(Succinct, PO) is the same as PG(Succinct, PO) with the additional promise that the poset is N-free.

4.6 The first results

Here are the first lower bounds known for poset games, given roughly in chronological order. The first four involve Nim; the first two of these consider explicit games, and the next two consider succinct games. None of these results is currently published; proof sketches can be found in the extended paper.

Theorem 4.3 (Wagner, 2009)**.** Nim_4(Explicit, HD) *is* **L**-*hard under* $\mathbf{AC}^0$ *reductions.*

The proof reduces from the promise problem ORD (order between vertices), which is known to be complete for **L** via quantifier-free projections [8, 15].

Theorem 4.4 (Thierauf, 2009)**.** Nim_2(Explicit, AR) *is* **NL**-*hard under* AC^0 *reductions.*

The proof reduces from the reachability problem for directed graphs, which is **NL**-complete under $\mathbf{AC}^0$-reductions.

The next result about succinct poset games is straightforward.

Theorem 4.5 (F, 2009)**.** Nim_2(Succinct, PO) *is* $\mathbf{co}C_{=}\mathbf{P}$-*hard under p-reductions.*

The idea here is that, for any $L \in \mathbf{coC_{=}P}$ and any input x, we produce two NIM stacks, and $x \in L$ if and only if they are of unequal length.

Theorem 4.6 (F, 2009)**.** Nim_6(Succinct, HD) *is* **PSPACE**-*hard under p-reductions.*

The proof uses a result of Cai & Furst [4] based on techniques of David Barrington on bounded-width branching programs. Recall that S_5 is the group of permutations of the set $\{1, 2, 3, 4, 5\}$. Their result is essentially as follows:

Theorem 4.7 (Cai & Furst)**.** *For any* **PSPACE** *language L, there exists a polynomial p and a polynomial-time computable (actually, log-space computable) function σ such that, for all strings x of length n and positive integers c (given in binary), $\sigma(x, c)$ is an element of S_5, and $x \in L$ if and only if the composition $\sigma(x, 1)\sigma(x, 2)\sigma(x, 2) \cdots \sigma(x, 2^{p(n)})$, applied left to right, fixes the element* 1.

The idea is that we connect the first five NIM stacks level-by-level via permutations in S_5, as well as adding a couple of widgets. If the product of all the permutions fixes 1, then we get five NIM stacks of equal length $N + 1$ and one NIM stack of length $N + 3$, which is an $\exists$-game by the Sprague-Grundy theorem. If 1 is not fixed, then we get four stacks of length $N+1$ and two of length $N+2$—a $\forall$-game by the same theorem.

Although the above results all mention Nim, the representations we use of a Nim game as a poset are not the natural one. Therefore, it is better to consider these as lower bounds on N-free poset games, which *are* naturally represented as posets.

The next results regard N-free games. They depend on Adam Kalinich's game outcome-flipping trick. The trick turns a poset game A into another poset game $\neg A$ with opposite outcome, starting with A and applying series and parallel union operations in a straightforward way. Here we describe a simplification of the trick due to Daniel Grier:
Given a poset A,

1. Let k be any (convenient) natural number such that $2^k \geq |A|$ (that is, A has at most 2^k elements).

2. Let $B := A/C_{2^k-1}$.

3. Let $C := B + C_{2^k}$.

4. Let $D := C/C_1$.

5. Finally, define $\neg A := D + A$.

Exercise 4.8. Check that: (1) if $g(A) \neq 0$, then $g(\neg A) = 0$; (2) if $g(A) = 0$, then $g(\neg A) = 2^{k+1}$. See the extended paper for a proof.

Observe that the size of $\neg A$ is linearly bounded in $|A|$. In fact, $|\neg A| \leq 6|A|$ if $A \neq \emptyset$.

Theorem 4.9 (Kalinich [16])**.** N-Free(Explicit, PO) *is* $\mathbf{NC}^1$*-hard under* $\mathbf{AC}^0$ *reductions.*

Proof sketch. We reduce from the Circuit Value problem for $\mathbf{NC}^1$ circuits with a single output. Given an **NC** circuit C with a single output and whose inputs are constant Boolean values, we produce a poset game P so that P is an $\exists$-game if and only if $C = 1$. We can assume WLOG that all gates in C are either (binary) OR-gates or NOT-gates. Starting with the input nodes, we associate a poset P_n with every node n in C from bottom up so that the outcome of P_n matches the Boolean value at node n. P is then the poset associated with the output node of C. The association is as follows:

- If n is an input node, we set $P_n := \emptyset$ if $n = 0$; otherwise, if $n = 1$, we set $P_n := C_1$.
- If n is an OR-gate taking nodes ℓ and r as inputs, then we set $P_n := P_\ell / P_r$. (Recall Exercise 1.4.)
- If n is a NOT-gate taking node c as input, we set $P_n := \neg P_c$.

This transformation from C to P can be done in (uniform) $\mathbf{AC}^0$, producing a poset of polynomial size, provided C has $O(\log n)$ depth. □

The next theorem is not published elsewhere.

Theorem 4.10 (F, 2011)**.** N-Free(Succinct, PO) *is* **PP***-hard under p-reductions.*

To prove this, we generalize the Kalinich/Grier construction a bit.

Definition 4.11. For any poset A and any integer $t > 0$, define

$$\text{Threshold}(A, t) := \frac{(A/C_{2^k - t}) + C_{2^k}}{C_t} + A\ ,$$

where k is any convenient natural number (the least, say) such that $2^k > \max(|A| - t, t - 1)$.

Note that $\neg A = \text{Threshold}(A, 1)$. It can be checked that

$$g(\text{Threshold}(A, t)) = \begin{cases} 2^{k+1} & \text{if } g(A) < t, \\ 0 & \text{if } g(A) \geq t. \end{cases} \tag{1}$$

We then use the Threshold$(\cdot, \cdot)$ operator to polynomially reduce any **PP** language to N-Free(Succinct, PO).

4.7 A note on the complexity of the g-number

Of course, computing the g-number of an impartial game is at least as hard as computing its outcome, the latter just being a test of whether the g-number is zero. Is the reverse true, i.e., can we polynomial-time reduce computing the g-number to computing the outcome? For explicitly represented poset games, this is certainly true. Given an oracle S returning the outcome of any poset game, we get the g-number of a given poset game G as follows: query S with the games $G, G + C_1, G + C_2, \ldots, G + C_n$, where n is the number of options of G (recall that that C_i is a NIM stack of size i). By the Sprague-Grundy theorem (Theorem 2.27), all of these are $\exists$-games except $G + C_{g(G)}$, which is a $\forall$-game.

What about succinctly represented games? The approach above can't work, at least for poset games, because the poset has exponential size. Surprisingly, we can still reduce the g-number to the outcome for succinct poset games in polynomial time, using the threshold construction of Definition 4.11 combined with binary search. Given a succinctly represented poset P of size $\leq 2^n$, first query S with Threshold$(P, 2^{n-1})$. If S says that this is an $\exists$-game, then we have $g(P) < 2^{n-1}$; otherwise, $g(P) \geq 2^{n-1}$. Next, query S with Threshold$(P, 2^{n-2})$ in the former case and Threshold$(P, 3 \cdot 2^{n-2})$ in the latter case, and so on. Note that in this reduction, the queries are adaptive, whereas they are nonadaptive for explicitly represented games.

4.8 PSPACE-completeness

In this section we sketch the proofs of two recent **PSPACE**-completeness results for poset game. The first, by Daniel Grier, is that the outcome problem for general explicit (impartial) poset games is **PSPACE**-complete [13]. The second is a similar result about the complexity of black-white poset games [10].

Theorem 4.12 (Grier [13])**.** *Deciding the outcome of an arbitrary finite poset game is* **PSPACE***-complete.*

Here we describe the reduction but do not prove correctness. See the extended paper or [13] for a full proof.

Proof sketch. Membership in **PSPACE** is clear. For **PSPACE**-hardness, we reduce from Node Kayles. Let $G = (V, E)$ (a simple undirected graph) be an arbitrary instance of Node Kayles. By altering the graph slightly if necessary without changing the outcome of the game, we can assume that $|E|$ is odd and that for every $v \in V$ there exists $e \in E$ not incident with v. We can do this by adding two disjoint cliques to G—either two K_2's or a K_2 and a K_4, whichever of these options results in an odd number of edges. We then construct the following three-level poset P from G:

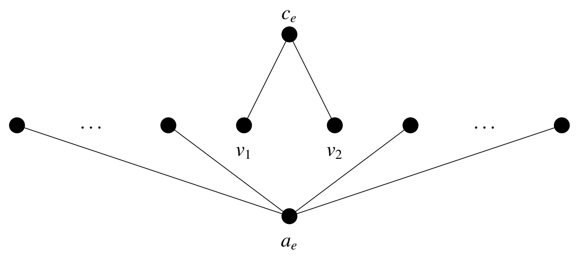


Figure 3: The $<$ relations in P obtained from the edge $e = \{v_1, v_2\}$ in G.

- The points of P are grouped into three disjoint antichains, A, B, and C, with A being the set of minimal points, C the maximal points, and B the points intermediate between A and C.
- For each edge $e \in E$ there correspond unique points $c_e \in C$ and $a_e \in A$, and vice versa.
- We let $B := V$.
- For each edge $e = \{v_1, v_2\}$ and $b \in B$, we have $b < c_e$ iff $b = v_1$ or $b = v_2$, and $a_e < b$ iff this is not the case, i.e., iff $b \neq v_1$ and $b \neq v_2$. This is illustrated in Figure 3.

This construction can clearly be done in polynomial time, given G. □

Finally, we turn to the complexity of black-white poset games. The next theorem is the first **PSPACE**-hardness result for a numeric game.

Theorem 4.13. *Determining the outcome of a black-white poset game is* **PSPACE**-*complete.*

Proof sketch. Membership in **PSPACE** is straightforward. For hardness, we reduce from TQBF. We present the reduction in detail and briefly describe optimal strategies for the winning players, but we do not show correctness. See the extended version for a more detailed sketch and [10] for a full proof.

Suppose we are given a fully-quantified boolean formula φ of the form $\exists x_1 \forall x_2 \exists x_3 \cdots \exists x_{2n-1} \forall x_{2n} \exists x_{2n+1} f(x_1, x_2, \ldots, x_{2n+1})$, where $f = c_1 \wedge c_2 \wedge \cdots \wedge c_m$ is in cnf with clauses $c_1, \ldots, c_m$. We define a two-level black-white poset (game) X based on φ as follows:

- X is divided into sections. There is a section (called a *stack*) for each variable, a section for the clauses (the *clause section*), and a section for fine-tuning the balance of the game (*balance section*).

- The ith stack consists of a set of incomparable *waiting nodes* W_i above (i.e., greater than) a set of incomparable *choice nodes* C_i. We also have a pair of *anti-cheat nodes*, α_i and β_i, on all stacks except the last stack. For odd i, the choice nodes are white, the waiting nodes are black, and the anti-cheat nodes are black. The colors are reversed for even i.

- The set of choice nodes C_i, consists of eight nodes corresponding to all configurations of three bits (i.e., $000, 001, \ldots, 111$), which we call the *left bit*, *assignment bit* and *right bit* respectively.

- The number of waiting nodes is $|W_i| = (2n + 2 - i)M$, where M is the number of non-waiting nodes in the entire game. It is important that $|W_i| \geq |W_{i+1}| + M$.

- The anti-cheat node α_i is above nodes in C_i with right bit 0 and nodes in C_{i+1} with left bit 0. Similarly, β_i is above nodes in C_i with right bit 1 and nodes in C_{i+1} with left bit 1.

- The *clause section* contains a black *clause node* b_j for each clause c_j, in addition to a black *dummy node*. The clause nodes and dummy node are all above a single white *interrupt node*. The clause node b_j is above a choice node z in C_i if the assignment bit of z is 1 and x_i appears positively in c_j, or if the assignment bit of z is 0 and x_i appears negatively in c_j.

- The balance section or *balance game* is incomparable with the rest of the nodes. The game consists of eight black nodes below a white node, which is designed to have numerical value $-7\frac{1}{2}$. All nodes in this section are called *balance nodes*.

The number of nodes is polynomial in m and n, so the poset can be efficiently constructed from φ.

A sample construction is shown in Figure 4. The idea is that players take turns playing choice nodes, starting with White, and the assignment bits of the nodes they play constitute an assignment of the variables, $x_1, \ldots, x_{2n+1}$. The assignment destroys satisfied clause nodes, and it turns out that Black can win if there remains at least one clause node. The waiting nodes and anti-cheat nodes exist to ensure players take nodes in the correct order. The interrupt node and dummy node control how much of an advantage a clause node is worth (after the initial assignment), and the balance node ensures the clause node advantage can decide whether White or Black wins the game. One can show that White (i.e., Right) can force a win when playing first if and only if the formula is true. $\square$

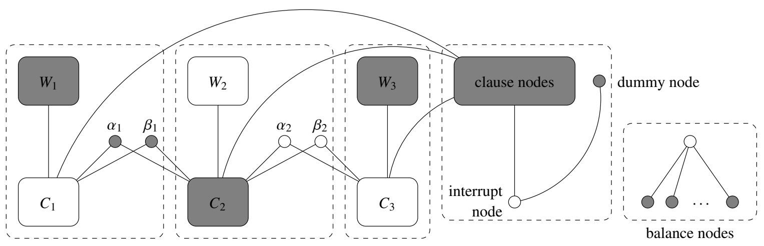


Figure 4: An example game with three variables ($n = 1$). Circles represent individual nodes, blobs represent sets of nodes, and χ is the set of clause nodes. An edge indicates that some node in the lower level is less than some node in the upper level. The dotted lines divide the nodes into sections (stacks, clause section and balance section).

5 Open questions

Are there interesting games whose complexity is complete for a subclass of **PSPACE**? The natural black-white version of GenCol is complete for the class $\mathbf{P}^{\mathbf{NP}[\log]}$, but the game itself and the reasons for its complexity are not so interesting. In this version, each uncolored node is reserved ("tinted") for being colored one or the other color, e.g., some node u can only be colored black, while some other node v can only be colored white. Then the outcome of this game depends only on which subgraph (the black-tinted nodes or the white-tinted nodes) contains a bigger independent set. Given two graphs G_1 and G_2, the problem of determining whether G_1 has a bigger independent set than G_2 is known to be complete for $\mathbf{P}^{\mathbf{NP}[\log]}$ [27].

Fix a natural number $k > 2$. For poset games of bounded width k, defined in Section 3.2.1, is there an algorithm running in time $o(n^k)$?

Grier's proof that the poset game decision problem is **PSPACE**-complete (Theorem 4.12) constructs posets having three levels, that is, whose maximum chain length is three. What about two-level games? Those having a single maximum or a single minimum element are easily solved. What is the complexity of those with more than one minimum and more than one maximum? Certain subfamilies of two-level posets have g-numbers that show regular patterns and are easily computed, or example, games where each element is above or below at most two elements, as well as "parity uniform" games [9]. Despite this, we conjecture that the class of all two-level poset games is **PSPACE**-complete, but are nowhere near a proof. Are there larger subfamilies of the two-level poset games that are in **P**?

A more open-ended goal is to apply the many results and techniques of combinatorial game theory, as we did in Theorem 4.13, to more families of games.

Finally, we mention a long-standing open problem about a specific infinite poset game: What is the outcome of the game $\mathbb{N}^3 - \{(0,0,0)\}$, where $(x_1, x_2, x_3) \le (y_1, y_2, y_3)$ iff $x_i \le y_i$ for all $i \in \{1, 2, 3\}$?

References

[1] E. R. Berlekamp, J. H. Conway, and R. Guy. *Winning Ways for your Mathematical Plays*. Academic Press, 1982.

[2] C. L. Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, 3:35–39, 1901-1902.

[3] S. Byrnes. Poset game periodicity. *INTEGERS: The Electronic Journal of Combinatorial Number Theory*, 3, 2003.

[4] Jin-Yi Cai and Merrick Furst. PSPACE survives constant-width bottlenecks. *Int. J. Found. Comput. Sci.*, 02(01):67, March 1991.

[5] J. H. Conway. *On Numbers and Games*. Academic Press, 1976.

[6] W. Deuber and S. Thomassé. Grundy sets of partial orders. www.mathematik.uni-bielefeld.de/sfb343/preprints/pr96123.ps.gz.

[7] S. Even and R. E. Tarjan. A combinatorial problem which is complete in polynomial space. *Journal of the ACM*, 23:710–719, 1976.

[8] Kousha Etessami. Counting quantifiers, successor relations, and logarithmic space. *Journal of Computer and System Sciences*, 54(3):400–411, 1997.

[9] S. A. Fenner, R. Gurjar, A. Korwar, and T. Thierauf. On two-level poset games. Technical Report TR13-019, Electronic Colloquium on Computational Complexity, 2013.

[10] S. A. Fenner, D. Grier, J. Meßner, L. Schaeffer, and T. Thierauf. Game values and computational complexity: An analysis via black-white combinatorial games. Technical Report TR15-021, Electronic Colloquium on Computational Complexity, February 2015.

[11] D. Gale. A curious nim-type game. *Amer. Math. Monthly*, 81:876–879, 1974.

[12] M. Garey and D. Johnson. *Computers and Intractability*. W. H. Freeman and Company, 1979.

[13] Daniel Grier. Deciding the winner of an arbitrary finite poset game is PSPACE-complete. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming*, volume 7965-7966 of *Lecture Notes in Computer Science*, pages 497–503. Springer-Verlag, 2013.

[14] P. M. Grundy. Mathematics and games. *Eureka*, 2:6–8, 1939.

[15] Birgit Jenner, Johannes Köbler, Pierre McKenzie, and Jacobo Torán. Completeness results for graph isomorphism. *Journal of Computer and System Sciences*, 66(3):549–566, 2003.

[16] A. O. Kalinich. Flipping the winner of a poset game. *Information Processing Letters*, 112(3):86–89, January 2012.

[17] Donald E. Knuth. *Surreal Numbers*. Addison-Wesley, 1974.

[18] J. B. Kruskal. The theory of well-quasi-ordering: A frequently discovered concept. *Journal of Combinatorial Theory*, 13(3):297–305, 1972.

[19] David Lichtenstein and Michael Sipser. GO is polynomial-space hard. *Journal of the ACM*, 27(2):393–401, 1980.

[20] L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM Journal on Computing*, 8(2):151–174, 1979.

[21] T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2):185–225, 1978.

[22] F. Schuh. Spel van delers (game of divisors). *Nieuw Tijdschrift voor Wiskunde*, 39:299, 2003.

[23] A. N. Siegel. *Combinatorial Game Theory*, volume 146 of *Graduate Studies in Mathematics*. American Mathematical Society, 2013.

[24] M. Sipser. *Introduction to the Theory of Computation (2nd Ed.)*. Course Technology, Inc., 2005.

[25] R. P. Sprague. Über mathematische Kampfspiele. *Tohoku Mathematical Journal*, 41:438–444, 1935-1936.

[26] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[27] H. Spakowski and J. Vogel. θ_p^2-completeness: A classical approach for new results. In *Proceedings of the 20th Conference on Foundations of Software Technology and Theoretical Computer Science (FST TCS)*, number 1974 in Lecture Notes in Computer Science, pages 348–360, 2000.

[28] T. Thierauf, 2009. Private communication.

[29] J. Úlehla. A complete analysis of Von Neumann's Hackendot. *International Journal of Game Theory*, 9:107–113, 1980.

[30] J. Valdes, R. E. Tarjan, and E. L. Lawler. The recognition of series parallel digraphs. *SIAM Journal on Computing*, 11:298–313, 1982.

[31] F. Wagner, 2009. Private communication.

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

Distributed Computing Column
Maurice Herlihy's 60th Birthday Celebration

Panagiota Fatourou
FORTH ICS & University of Crete
`faturu@csd.uoc.gr`

Maurice Herlihy is one of the most renowned members of the Distributed Computing community. He is currently a professor in the Computer Science Department at Brown University. He has an A.B. in Mathematics from Harvard University, and a Ph.D. in Computer Science from M.I.T. He has served on the faculty of Carnegie Mellon University and on the staff of DEC Cambridge Research Lab. He is the recipient of the 2003 Dijkstra Prize in Distributed Computing, the 2004 Gödel Prize in theoretical computer science, the 2008 ISCA influential paper award, the 2012 Edsger W. Dijkstra Prize, and the 2013 Wallace McDowell award. He received a 2012 Fullbright Distinguished Chair in the Natural Sciences and Engineering Lecturing Fellowship, and he is a fellow of the ACM, a fellow of the National Academy of Inventors, and a member of the National Academy of Engineering and the American Academy of Arts and Sciences.

On the occasion of his 60th birthday, the SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC), which was held in Paris, France in July 2014, hosted a celebration which included several technical presentations about Maurice's work by colleagues and friends. This column includes a summary of some of these presentations, written by the speakers themselves. In the first article, Vassos Hadzilacos overviews and highlights the impact of Maurice's seminal paper on wait-free synchronization. Then, Tim Harris provides a perspective on hardware trends and their impact on distributed computing, mentioning several interesting open problems and making connections to Maurice's work. Finally, Michael Scott gives a concise retrospective on transactional memory, another area where Maurice has been a leader. This is a joint column with the Distributed Computing Column of ACM SIGACT News (June 2015 issue), edited by Jennifer Welch. Many thanks to Vassos, Tim, and Michael for their contributions!

A Quarter-Century of Wait-Free Synchronization[1]

Vassos Hadzilacos
Department of Computer Science
University of Toronto, Canada
vassos@cs.toronto.edu

It is an honour and a pleasure to have the opportunity to speak about what in my opinion is Maurice Herlihy's most influential paper, and indeed one of the most significant papers in the theory of distributed computing. I am referring to his work on wait-free synchronization, which appeared in preliminary form in PODC 1988 [8] and in its final form in *TOPLAS* three years later [10]. I will first review the key contributions of this paper and then I will discuss its impact.

1 Review of the key contributions

The context for this work is a distributed system in which processes take steps ***asynchronously*** and communicate by accessing ***shared objects***. Here asynchrony means that between successive steps of a process other processes may take an arbitrary number of steps. Processes are subject to crash failures, meaning that they may stop taking steps altogether, even though they have not reached the end of their computation. For convenience, we assume that a process is designed to take steps (perhaps no-ops) forever, and so we can define a process to have crashed if it takes only finitely many steps in an infinite execution. Minimally, the shared

[1]Remarks on the occasion of Maurice Herlihy's 60th birthday in PODC 2014. Based on the transparencies used in an oral presentation on July 15, 2014, Paris, France. I have tried to preserve the informal tone of that oral presentation here. Supported in part by a grant from the Natural Sciences and Engineering Council of Canada.

objects that the processes use to communicate are registers accessible via separate but atomic write and read operations. The shared objects can also include registers with additional operations such as fetch-and-add, whereby a process atomically increments the value of the register by a specified amount and reads the value of the register before it was incremented; or even other types of shared objects, such as queues or stacks.

The key question that animates the paper is the following:

> "For given object types A and B, in a system with n processes, can we implement an object of type A using objects of type B and registers?"

In what follows, we will take registers (with atomic write and read operations) for granted. So, the above question will be simplified to "in a system of n processes, can we implement an object of type A using objects of type B?"

Here are some specific instances of this question:

- Can we implement a queue shared by two processes using only registers? Herlihy showed that the answer to this question is negative.

- Can we implement a register with a fetch-and-add operation, shared by five processes, using registers with a compare-and-swap operation?[1] Herlihy showed that the answer to this question is affirmative.

What is significant about this paper is not so much the answer to these specific questions, but the tools that it gave us to answer such questions in general.

Having set the general context for the paper, I will now describe its main contributions.

Contribution 1: Model of computation

The type of an object specifies what operations can be applied to an object (of that type) and how the object is supposed to behave ***when operations are applied to it sequentially***. For example, the type queue tells us that we can access the object via enqueue and dequeue operations only, and that in a ***sequence*** of such operations items are dequeued in the order in which they were enqueued. But how should a shared queue behave if operations are applied to it by processes concurrently?

[1]A compare-and-swap operation applied to register X takes two parameters, values *old* and *new*, and has the following effect (atomically): If the present value of X is equal to *old* then X is assigned the value *new* and the value "success" is returned; otherwise, the value of X is not changed and the value "failure" is returned.

More generally, what exactly are the properties that an implementation of an object of type A (shared by n processes) should have? Herlihy requires two properties of such an implementation: linearisability and wait freedom.[2]

Linearisability: The implemented object should behave as if each operation took effect instantaneously, at some point between its invocation and its response.

Wait freedom: An operation on the implemented object invoked by a nonfaulty process eventually terminates, regardless of whether other processes are fast, slow, or even crash.

Note that the requirement of wait freedom implies that implementations

(a) may not use locks: otherwise, a process that crashes while holding a lock could prevent all others from terminating, and

(b) must be starvation free: not only must the system as a whole make progress but each individual nonfaulty process must complete all its operations.

The first important contribution of the paper was the articulation of ***a compelling, elegant, and pragmatic model of computation***.

Contribution 2: Comparing the power of object types

Recall the basic question the paper addresses: In a system of n processes, can we implement an object of type A using objects of type B? An affirmative answer to such a question presents no methodological difficulties: One presents an implementation and a proof that it satisfies the two requisite properties. But what if the answer is negative? How can we prove that A cannot be implemented using B? One way to do so is to show that there is some task C that A can do, and that B cannot do. So, task C is a "yardstick" that can be used to compare A and B. Another key contribution of the paper is ***the identification of the right kind of yardstick*** to compare types, namely, solving the ***consensus problem***. This problem, which under various forms and guises had been studied extensively in fault-tolerant distributed computing before Herlihy's paper, can be described as follows:

- Each process starts with an input.

[2]In my oral presentation, I referred to linearisability as a safety property, and to wait freedom as a liveness property. Rachid Guerraoui, who was in the audience, brought to my attention a paper of his with Eric Ruppert in which they show that this is not quite right [6]: There are types with infinite non-determinism for which linearisability is not a safety property; for types with bounded non-determinism, however, linearisability is indeed a safety property.

- Each nonfaulty process produces an output.
- The output of any process is the input of some process (validity), and is no different than the output of any other process (agreement).

Note that we are interested in ***wait-free*** solutions for this problem. Let us examine some examples of the use of this "yardstick" to prove non-implementability results.

Example 1: To show that in a system of two processes we cannot implement a queue using registers we prove that

(1) using queues we can solve consensus for two processes; and

(2) using registers we cannot solve consensus for two processes.

From (1) and (2) we conclude that we cannot implement queues using only registers: For, if we could, we would combine such an implementation with (1) to obtain an algorithm that uses registers and solves consensus for two processes, contradicting (2).

Example 2: To show that in a system of three processes we cannot implement a register with the compare-and-swap operation using registers with a fetch-and-add operation we prove that

(1) using registers with compare-and-swap we can solve consensus for three processes; and

(2) using registers with fetch-and-add we cannot solve consensus for three processes.

Using similar reasoning as in Example 1, from (1) and (2) we conclude that we cannot implement compare-and-swap using only fetch-and-add.

So, to capture the "power" of an object type A, Herlihy attaches to A a ***consensus number***, namely the unique integer n such that:

- using objects of type A we can solve consensus for n processes, and
- using objects of type A we cannot solve consensus for $n + 1$ processes.

If no such integer n exists, the consensus number of A is ∞. The following, methodologically very useful, theorem follows immediately from this definition.

Theorem 1.1 ([8, 10])**.** *If type A has consensus number n and type B has consensus number $m < n$, then A cannot be implemented from B in a system with more than m processes.*

This leads us to Herlihy's ***consensus hierarchy*** of object types: A type A is at level n of the consensus hierarchy if and only if its consensus number is n — i.e., if and only if A solves consensus for n, but not for $n + 1$, processes. Thus, by Theorem 1.1, "stronger" types are at higher levels of this hierarchy.

Figure 1 illustrates the consensus hierarchy. I now briefly explain the types mentioned in that figure that I have not already defined.

- The test-and-set type (at level 2) refers to a register initialised to 0, with an operation that atomically sets the register to 1 and returns the value of the register before the operation. (So, the operation that is linearised first returns 0, and all others return 1.)
- The n-consensus type (at level n) refers to an object with a Propose(v) operation, where v is an arbitrary value (say a natural number); the object returns the value proposed by the first operation to the first n Propose operations applied to it, and returns an arbitrary value to subsequent operations. (Thus, it is an object designed to solve consensus for n processes.)
- The n-peekable queue type (also at level n) refers to a kind of queue to which a maximum of n values can be enqueued (any values subsequently enqueued are lost) and which allows a process to "peek" at the first value enqueued without dequeuing it.
- The n-assignment type (at level $2n-2$) allows a process to atomically assign n specified values to n specified registers.
- The consensus type (at level ∞) is similar to n consensus, except that it returns to all Propose operations (not only to the first n) the value proposed by the first one.
- Finally, the memory-to-memory swap type (also at level ∞) allows a process to atomically swap the values of two specified registers.

Contribution 3: Universality of consensus

We have seen how Herlihy used consensus as a "yardstick" to compare the relative power of object types. But why is consensus the ***right*** yardstick? In principle, we could have taken any task and used it as a yardstick. For example, consider the ***leader election*** problem:

- Each nonfaulty process outputs "winner" or "loser".
- At most one process outputs "winner".

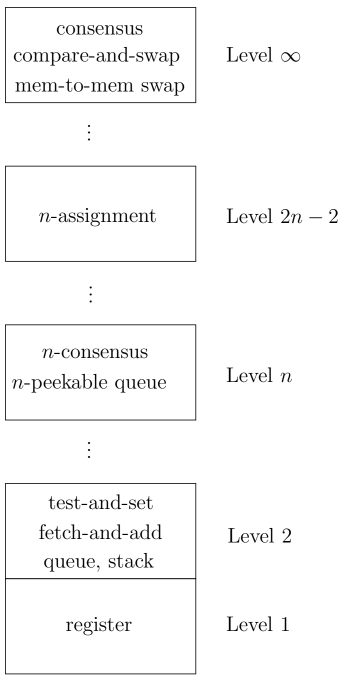


Figure 1: The consensus hierarchy

- Some process outputs "winner" or crashes after taking at least one step.

We could define the "leader election number" of type *A* to be the maximum number of processes for which A can solve the leader election problem — by analogy to the definition of the consensus number, but using a different problem as the yardstick. There is nothing in principle wrong with this, except that the resulting "leader election hierarchy" would not be very interesting: it would consist of just two levels: all types in levels two to infinity of the consensus hierarchy would coalesce into a single level! In other words, unlike consensus, the leader election yardstick is not a very discriminating one. So, what is special about consensus that makes it the right yardstick? The answer lies in the following important fact:

Theorem 1.2 ([8, 10])**.** *Any object type B with consensus number n is* ***universal*** *for n processes: it can implement an object of* ***any*** *type A, shared by n processes.*

The proof of this theorem is through an intricate algorithm that has come to be known as ***Herlihy's universal construction***. Given a function that defines the sequential behaviour of an arbitrary type A, this construction shows how to implement an object of type A shared by n processes using only registers and n-consensus objects. So, given any object of type B with consensus number n, we can solve the consensus problem for n processes (by definition of consensus number), and therefore we can implement n-consensus objects. Then, using Herlihy's universal construction, we can implement an object of type A shared by n processes.

At a very high level, the intuition behind this theorem is simple: Processes use consensus to agree on the order in which to apply their operations on the object they implement. Between this intuition and an actual working algorithm that satisfies wait freedom, however, there is a significant gap. Herlihy's universal construction is an algorithm well worth studying carefully, and returning to every now and then!

2 Impact

The impact of the paper is accurately reflected by its citation count. A Google Scholar search conducted in July 2014 showed over 1400 citations for [10] and over 200 for [8]. Let us look beyond the numbers, however, into the specific ways in which Herlihy's paper on wait-free synchronisation has influenced the field of distributed computing.

Impact 1: The model

The model of asynchronous processes communicating via linearisable, wait-free shared objects that was articulated in a complete form in this paper has been a very influential one. As noted earlier, it is mathematically elegant but also pragmatic. It is certainly true that different aspects of this model appeared earlier, but I believe that this was the first paper that presented the complete package. It is nevertheless useful to trace the heritage.

Shared memory: The asynchronous shared memory model goes back to Dijkstra's seminal paper on mutual exclusion [3].

Wait freedom: The concept of wait-free implementations (though not under this name) originated in Lamport's and Peterson's work on implementations of shared registers [15, 19, 16, 17].

Linearisability: The concept of linearisability as the correctness criterion for the behaviour of shared objects was introduced by Herlihy and Wing [12, 13].

Impact 2: Lock-free data structures

The idea of synchronising access to data structures without relying on locks has had a significant impact on the practice of concurrent programming. Although locking is still (and may well remain) the predominant mechanism employed to coordinate access to data structures by multiple processes, Herlihy's paper helped highlight some of its shortcomings (potential for deadlock, unnecessary restrictions to concurrency, intolerance to even crash failures, priority inversions) and pointed the way to the possibility of synchronising without using locks. There is, by now, an extensive literature on so-called ***lock-free*** data structures. In this context, lock free doesn't necessarily mean wait free. It is a term that encompasses wait freedom as well as the weaker ***non-blocking*** property, which requires that progress be made by ***some*** non-faulty process, not necessarily ***every*** non-faulty process.[3]

Impact 3: Weaker liveness properties

Linearisable wait-free implementations tend to be complex, and one culprit seems to be wait freedom. The most intricate aspect of Herlihy's universal construction is the so-called helping mechanism, which ensures that "no process is left behind". If one is willing to settle for the less demanding non-blocking property, the universal construction becomes much simpler.

The observation that wait freedom seems to complicate things and that it is perhaps too strong a liveness property has led researchers to investigate other liveness properties, weaker than wait freedom, easier to implement, but hopefully still useful in practice. The following are some examples of objects with relaxed liveness requirements:

Obstruction-free objects: Every operation invoked by a nonfaulty process that eventually runs solo (i.e., without interference from other processes) terminates [4, 11].

"Pausable" objects: Every operation invoked by a live process eventually returns control to the caller, either by completing normally, or by aborting without taking effect, or by "pausing" so that another operation can run solo and terminate. An operation can abort or pause only if it encounters interference.

[3]The terms "lock free" and "non-blocking" are not used consistently in the literature; in some papers their meaning is as given here, in others it is reversed.

A nonfaulty process whose operation was paused is required to resume the paused operation and complete it (normally or by aborting) before it can do anything else [2].

Nondeterministic abortable objects: Every operation invoked by a nonfaulty process eventually returns to the caller either by completing normally or by aborting. An operation can abort only if it encounters interference. An aborted operation may or may not have taken effect, and the caller doesn't know which of these two possibilities is the case [1].

Abortable objects: Every operation invoked by a nonfaulty process eventually returns to the caller either by completing normally or by aborting. An operation can abort only if it encounters interference. An aborted operation is guaranteed not to have taken effect [7].

Impact 4: Structure of the "A implemented by B" relation

Though the consensus number of an object type A encapsulates much information about A's ability to implement other types, it does not tell the whole story. By Theorem 1.2, if A has consensus number n, it can support the implementation of any object shared by n processes; but what about the implementation of even "weak" objects, i.e., objects of types whose consensus number is no greater than n, ***shared by more than n processes***? In this setting, there are phenomena that run counter to the notion that the higher the consensus number of a type the greater its power to implement other types.

Consider the following question: Are all object types at the same level of the consensus hierarchy equivalent? That is, if A and B are two types at the same level n of the consensus hierarchy, can an object of type A, shared by ***any*** number m of processes, be implemented using objects of type B? Or, equivalently (in view of Theorem 1.2), can any object of a type with consensus number n, shared by ***any*** number of processes, be implemented using n-consensus? Herlihy himself proved that this is not the case for level 1: He demonstrated a type at level 1 that cannot be implemented from registers (which are also at level 1) [9]. Rachman proved that this is the case for every level [20]: For every positive integer n, he demonstrated a type T_n at level n of the consensus hierarchy such that an object of type T_n shared by $2n + 1$ processes cannot be implemented using n-consensus objects.[4] (In fact, Rachman's result is more general: for any positive integers n, m such that $m \leq n$, there is a type T_m at level m of the consensus hierarchy such that

[4]My account in this paragraph differs from my oral presentation in Paris, as a result of things I learned in the meanwhile — but should have known then!

an object of type T_m, shared by $2n + 1$ processes, cannot be implemented using n-consensus objects.)

A related set of investigations concern the matter of "robustness" of the consensus hierarchy. Consider a system with n processes. By the definition of consensus number, objects of a type with consensus number less than n cannot implement an n-consensus object. Is it possible, however, to use objects of ***multiple*** "weak" types (with consensus number less than n) to implement n-consensus? If this is possible, we say that the consensus hierarchy is ***not robust***. Jayanti was the first to identify and study the issue of robustness; he proved that under a restricted definition of implementation of one type by others, the consensus hierarchy is not robust [14]. Later, Schenk proved that under a restricted definition of wait freedom, the consensus hierarchy is not robust [21]. Lo and Hadzilacos proved that under the usual definitions of implementation and wait freedom, the consensus hierarchy is not robust [18].

Impact 5: Elevating the status of the bivalency argument

George Pólya and Gabor Szegö made a famous quip about the distinction between a trick and a method:

> "An idea that can be used only once is a trick. If one can use it more than once, it becomes a method." (*Problems and Theorems in Analysis*, 1972.)

Fischer, Lynch, and Paterson gave us the bivalency argument as a brilliant trick in their proof of the impossibility of consensus in asynchronous message-passing systems [5]. With his masterful use of the same argument to prove that consensus among n processes cannot be solved using objects of type B (for several choices of n and B), Herlihy elevated bivalency to the more exalted status of a method!

Impact 6: Design of multiprocessors?

I put a question mark for this impact, because here I am speculating: I do not really know why, in the late 1980s and early 1990s, multiprocessor architects abandoned operations with low consensus number in favour of universal ones. But the timing is such that I wouldn't be surprised to learn that these architects were influenced, at least in part, by Herlihy's discovery that, from the perspective of wait-free synchronisation, much more is possible with operations such as compare-and-swap or load-linked/store-conditional than with operations such as test-and-set or fetch-and-add.

Great papers answer important questions, but also open new ways of thinking, and perhaps even influence practice. Herlihy's paper on wait-free synchronisation delivers on all these counts!

Acknowledgements

I am grateful to Naama Ben-David and David Chan for their comments on this paper.

References

[1] Marcos K. Aguilera, Sven Frolund, Vassos Hadzilacos, Stephanie Horn, and Sam Toueg. Abortable and query-abortable objects and their efficient implementation. In *PODC '07: Proceedings of the 26th Annual ACM Symposium on Principles of Distributed Computing*, pages 23–32, 2007.

[2] Hagit Attiya, Rachid Guerraoui, and Petr Kouznetsov. Computing with reads and writes in the absence of step contention. In *DISC '05: Proceedings of the 19th International Symposium on Distributed Computing*, pages 122–136, 2005.

[3] Edgar W. Dijkstra. Solution of a problem in concurrent programming control. *Commununications of the ACM*, 8(9):569, 1965.

[4] Faith Fich, Maurice Herlihy, and Nir Shavit. On the space complexity of randomized synchronization. *Journal of the ACM*, 45(5):843–862, 1998.

[5] Michael Fischer, Nancy Lynch, and Michael Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.

[6] Rachid Guerraoui and Eric Ruppert. Linearizability is not always a safety property. In *Networked Systems - Second International Conference, NETYS 2014*, pages 57–69, 2014.

[7] Vassos Hadzilacos and Sam Toueg. On deterministic abortable objects. In *PODC '13: Proceedings of the 32nd ACM Symposium on Principles of Distributed Computing*, pages 4–12, 2013.

[8] Maurice Herlihy. Impossibility and universality results for wait-free synchronization. In *PODC '88: Proceedings of the 7th Annual ACM Symposium on Principles of Distributed Computing*, pages 276–290, 1988.

[9] Maurice Herlihy. Impossibility results for asynchronous PRAM. In *SPAA '91: Proceedings of the 3rd Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 327–336, 1991.

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

[11] Maurice Herlihy, Victor Luchangco, and Mark Moir. Obstruction-free synchronization: Double-ended queues as an example. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 522–529, Washington, DC, USA, 2003. IEEE Computer Society.

[12] Maurice Herlihy and Jeannette Wing. Axioms for concurrent objects. In *POPL '87: Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pages 13–26, New York, NY, USA, 1987. ACM Press.

[13] Maurice Herlihy and Jeannette Wing. Linearizability: A correctness condition for concurrent objects. *Transactions on Programming Languages and Systems*, 12(3):463–492, July 1990.

[14] Prasad Jayanti. On the robustness of Herlihy's hierarchy. In *PODC '93: Proceedings of the 12th Annual ACM Symposium on Principles of Distributed Computing*, pages 145–157, 1993.

[15] Leslie Lamport. On concurrent reading and writing. *Communications of the ACM*, 20(11):806–811, November 1977.

[16] Leslie Lamport. On interprocess communication. Part I: Basic formalism. *Distributed Computing*, 1(2):77–85, 1986.

[17] Leslie Lamport. On interprocess communication. Part II: Algorithms. *Distributed Computing*, 1(2):86–101, 1986.

[18] Wai-Kau Lo and Vassos Hadzilacos. All of us are smarter than any of is: wait-free hierarchies are not robust. In *STOC '97: In Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 579–588, 1997.

[19] Gary Peterson. Concurrent reading while writing. *ACM Transactions of Programming Languages and Systems*, 5(1):46–55, 1983.

[20] Ophir Rachman. Anomalies in the wait-free hierarchy. In *WDAG '94: Proceedings of the 8th International Workshop on Distributed Algorithms*, pages 156–163, 1994.

[21] Eric Schenk. The consensus hierarchy is not robust. In *PODC '97: In Proceedings of the 16th Annual ACM Symposium on Principles of Distributed Computing*, page 279, 1997.

Hardware Trends: Challenges and Opportunities in Distributed Computing

Tim Harris
Oracle Labs
Cambridge, UK
`timothy.l.harris@oracle.com`

This article is about three trends in computer hardware, and some of the challenges and opportunities that I think they provide for the distributed computing community. A common theme in all of these trends is that hardware is moving away from assumptions that have often been made about the relative performance of different operations (e.g., computation versus network communication), the reliability of operations (e.g., that memory accesses are reliable, but network communication is not), and even some of the basic properties of the system (e.g., that the contents of main memory are lost on power failure).

Section 1 introduces "rack-scale" systems and the kinds of properties likely in their interconnect networks. Section 2 describes challenges in systems with shared physical memory but without hardware cache coherence. Section 3 discusses non-volatile byte-addressable memory. The article is based in part on my talk at the ACM PODC 2014 event in celebration of Maurice Herlihy's sixtieth birthday.

1 Rack-Scale Systems

Rack-scale computing is an emerging research area concerned with how we design and program the machines used in data centers. Typically, these data centers are built from racks of equipment, with each rack containing dozens of discrete machines. Over the last few years researchers have started to weaken the boundaries between these individual machines, leading to new "rack-scale" systems. These architectures are being driven by the need to increase density and connectivity between servers, while lowering cost and power consumption.

Different researchers mean somewhat different things by "rack-scale" systems. Some systems are built from existing components. These are packaged together for a particular workload, providing appropriate hardware, and pre-installed software. Other researchers mean systems with internal disaggregation of components: rather than having a rack of machines each with its own network interface and disk, there might be a pool of processor nodes, disk nodes, and networking nodes, all connected over an internal intra-machine interconnect. The interconnect can be configured to connect sets of these resources together in different ways.

Initial commercial systems provide high-density processor nodes connected through an in-machine interconnect to storage devices or to external network interfaces. Two examples are the HP MoonShot [12] and AMD SeaMicro [22] single-box cluster computers. Many further ideas are now being explored in research projects—for instance, the use of custom system-on-chip (SoC) processors in place of commodity chips.

These systems should not just be seen as a way to build a faster data center. Communicating over a modern interconnect is different from communicating over a traditional packet-switched network. Some differences are purely trends in performance—a round-trip latency for over InfiniBand is around 1μs, not much longer than the time it takes to access data stored in DRAM on a large shared-memory multiprocessor. The Scale-Out NUMA architecture provides one example of how latencies may be reduced even further: it exposes the interconnect via a specialized "remote memory controller" (RMC) on a multi-core SoC [18]. Threads in one SoC can instruct the RMC to transfer data to or from memory attached to other processors in the system. Threads communicate with their RMC over memory-mapped queues (held in the SoC's local caches). These operations have much lower latency than accessing a traditional network interface over PCI-express. If network latencies continue to fall, while memory access latencies remain constant, then this will change the optimization goals when designing a protocol.

Other differences are qualitative: as with the Scale-Out NUMA RMC, the main programming interface in many rack-scale systems is RDMA (remote direct memory access). To software, RDMA appears as a transfer from a region of a sender's address space into a region in the receiver's address space. Various forms of control message and notification can be used—e.g., for a receiver to know when data has arrived, or for a sender to know when transmission is complete. Flow control is handled in hardware to prevent packet loss.

Some network devices provide low-latency hardware distribution of data to multiple machines at once (for instance, the ExaLINK matrix switches advertise

5ns latency multicasting data from an input port to any number of output ports [1]). Researchers are exploring how to use this kind of hardware as part of an atomic broadcast mechanism [7].

Research questions: What are the correct communication primitives to let applications benefit from low-latency communication within the system? What are the likely failure modes and how do we achieve fault tolerance? What is the appropriate way to model the guarantees provided by the interconnect fabric in a rack-scale system? How should the interconnect fabric be organized, and how should CPUs, DRAM, and storage be placed in it?

2 Shared Memory Without Cache Coherence

The second trend I will highlight is toward systems with limited support for cache coherence in hardware: Some systems provide shared physical memory, but rely on threads to explicitly flush and invalidate their local caches if they want to communicate through them. Some researchers argue that cache coherence will be provided within a chip, but not between chips [15].

This kind of model is not entirely new. For instance, the Cray T3D system distributed its memory across a set of processor nodes, providing each node with fast access to its local memory, and slower access to uncacheable remote memory [6]. This kind of model makes it important to keep remote memory accesses rare because they will be slow even in the absence of contention (for instance, lock implementations with local spinning are well suited in this setting [16]).

One motivation for revisiting this kind of model is to accommodate specialized processors or accelerators. The accelerator can transfer data to and from memory (and sometimes to and from the caches of the traditional processors) but does not need to participate in a full coherence protocol. A recent commercial example of this kind of system is the Intel Xeon Phi co-processor accessed over PCI-express [13].

A separate motivation for distributing memory is to provide closer coupling between storage and computation. The IRAM project explored an extreme version of this with the processor on the same chip as its associated DRAM [19]. Close coupling between memory and storage can improve the latency and energy efficiency of memory accesses, and permit the aggregate bandwidth to memory to grow by scaling the number of memory-compute modules.

Some research systems eschew the direct use of shared memory and instead focus on programming models based on message passing. Shared memory buffers can be used to provide a high-performance implementation of message passing (for instance, by using a block of memory as a circular buffer to carry messages).

This approach means that only the message passing infrastructure needs to be aware of the details of the memory system. Also, it means that software written for a genuinely distributed environment is able to run correctly (and hopefully more quickly) in an environment where messages stay within a machine.

Systems such as K2 [14] and Popcorn [4] provide abstractions to run existing shared-memory code in systems without hardware cache coherence, using ideas from distributed shared memory systems.

Conversely, the Barrelfish [5] and FOS [23] projects have been examining the use of distributed computing techniques within an OS. Barrelfish is an example of a *multikernel* in which each core runs a separate OS kernel, even when the cores operate in a single cache-coherent machine. All interactions between these kernels occur via message-passing. This design avoids the need for shared-memory data structures to be managed between cores, enabling a single system to operate across coherence boundaries. While it is elegant to rely solely on message passing, this approach seems better suited to some workloads than to others—particularly when multiple hardware threads share a cache, and could benefit from spatial and temporal locality in the data they are accessing.

Research questions: What programming models and algorithms are appropriate for systems which combine message passing with shared memory? To what extent should systems with shared physical memory (without cache coherence) be treated differently from systems without any shared memory at all?

3 Non-Volatile Byte-Addressable Memory

There are many emerging technologies that provide non-volatile byte-addressable memory (NV-RAM). Unlike ordinary DRAM, memory contents are preserved on power loss. Unlike traditional disks, locations can be read or written at a fine granularity—nominally individual bytes, although in practice hardware will transfer complete cache lines. Furthermore, unlike a disk, these reads and writes may be performed by ordinary memory access instructions (rather than using RDMA, or needing the OS to orchestrate block-sized transfers to or from a storage device).

This kind of hardware provides the possibility of an application keeping all of its data structures accessible in main memory. Researchers are starting to explore how to model NV-RAM [20]. Techniques from non-blocking data structures provide one starting point for building on NV-RAM. A power loss can be viewed as a failure of all of the threads accessing a persistent object. However, there are several challenges which complicate matters:

First, the memory state seen by the threads before the power loss is not necessarily the same as the state seen after recovery. This is because, although the

NV-RAM is persistent, the remainder of the memory system may hold data in ordinary volatile buffers such as processor caches and memory controllers. When power is lost, some data will transiently be in these volatile buffers. Aggressively flushing every update to NV-RAM may harm performance. Some researchers have explored flushing updates upon power-loss, but that approach requires careful analysis to ensure that there is enough residual power to do so [17].

The second problem is that applications often need to access several structures—for instance, removing an item from one persistent collection object, processing it, and adding it to another persistent collection. If there is a power loss during the processing step, then we do not want to lose the item.

Transactions provide one approach for addressing these two problems. It may be possible to optimize the use of cache flush/invalidate operations to ensure that data is genuinely persistent before a transaction commits, while avoiding many individual flushes while the transaction executes. As with transactional memory systems, transactions against NV-RAM would provide a mechanism for composing operations across multiple data structures [10]. What is less clear is whether transactions are appropriate for long-running series of operations (such as the example of processing an object when moving it between persistent collections).

Having an application's data structures in NV-RAM could be a double-edged sword. It avoids the need to define translations between on-disk and in-memory formats, and it avoids the time taken to load data into DRAM for processing. This time saving is significant in "big data" applications, not least when restarting a machine after a crash. However, explicit loading and saving has benefits as well as costs: It allows in-memory formats to change without changing the external representation of data. It allows external data to be processed by tools in a generic way without understanding its internal formats (backup, copying, de-duplication, etc.). It provides some robustness against transient corruption of in-memory formats by restarting an application and re-loading data.

It is difficult to quantify how significant these concerns will be. Earlier experience with persistent programming languages explored many of these issues [3]. Recent work on dynamic software updates is also relevant (e.g., Arnold and Kaashoek in an OS kernel [2], and Pina *et al.* in applications written in Java [21]).

Research questions: How should software manage data held in NV-RAM, and what kinds of correctness properties are appropriate for a data structure that is persistent across power loss?

4 Discussion

This article has touched on three areas where developments in computer hardware are changing some of the traditional assumptions about the performance and behavior of the systems we build on.

Processor clock rates are not getting significantly faster (and, many argue, core counts are unlikely to increase much further [9]). Nevertheless, there are other ways in which system performance can improve such as by integrating specialized cores in place of general-purpose ones, or by providing more direct access to the interconnect, or by removing the need to go through traditional storage abstractions to access persistent memory.

I think many of these trends reflect a continued blurring of the boundaries between what constitutes a "single machine" versus what constitutes a "distributed system". Reliable interconnects are providing hardware guarantees for message delivery, and in some cases this extends to guarantees about message ordering as well even in the presence of broadcast and multicast messages. Conversely, the move away from hardware cache coherence within systems means that distributed algorithms become used in systems which look like single machines—e.g., in the Hare filesystem for non-cache-coherent multicores [8].

Many of these hardware developments have been proceeding ahead of the advancement of formal models of the abstractions being built. Although the use of verification is widespread at low levels of the system – especially in hardware – I think there are important opportunities to develop new models of the abstractions exposed to programmers. There are also opportunities to influence the direction of future hardware evolution—perhaps as with how the identification of the consensus hierarchy pointed to the use of atomic compare and swap in today's multiprocessor systems [11].

References

[1] EXALINK Fusion (web page). Apr. 2015. `https://exablaze.com/exalink-fusion`.

[2] J. Arnold and M. F. Kaashoek. Ksplice: automatic rebootless kernel updates. In *Proc. 4th European Conference on Computer Systems (EuroSys)*, pages 187–198, 2009.

[3] M. Atkinson and M. Jordan. A review of the rationale and architectures of PJama: a durable, flexible, evolvable and scalable orthogonally persistent programming platform. Technical report, University of Glasgow, Department of Computing Science, 2000.

[4] A. Barbalace, M. Sadini, S. Ansary, C. Jelesnianski, A. Ravichandran, C. Kendir, A. Murray, and B. Ravindran. Popcorn: bridging the programmability gap in heterogeneous-ISA platforms. In *EuroSys '15: Proc. 10th European Conference on Computer Systems (EuroSys)*, page 29, 2015.

[5] A. Baumann, P. Barham, P.-E. Dagand, T. Harris, R. Isaacs, S. Peter, T. Roscoe, A. Schüpbach, and A. Singhania. The Multikernel: A new OS architecture for scalable multicore systems. In *SOSP '09: Proc. 22nd Symposium on Operating Systems Principles*, pages 29–44, 2009.

[6] Cray Research Inc. *CRAY T3D System Architecture Overview Manual*. 1993. `ftp://ftp.cray.com/product-info/mpp/T3D_Architecture_Over/T3D.overview.html`.

[7] M. P. Grosvenor, M. Fayed, and A. W. Moore. Exo: atomic broadcast for the rack-scale computer. 2015. `http://www.cl.cam.ac.uk/~mpg39/pubs/workshops/wrsc15-exo-abstract.pdf`.

[8] C. Gruenwald III, F. Sironi, M. F. Kaashoek, and N. Zeldovich. Hare: a file system for non-cache-coherent multicores. In *EuroSys '15: Proc. 10th European Conference on Computer Systems*, page 30, 2015.

[9] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 31(4):6–15, 2011.

[10] T. Harris, M. Herlihy, S. Marlow, and S. Peyton Jones. Composable memory transactions. In *PPoPP '05: Proc. 10th Symposium on Principles and Practice of Parallel Programming*, June 2005.

[11] M. Herlihy. Wait-free synchronization. *ACM Trans. Program. Lang. Syst.*, 13(1):124–149, Jan. 1991.

[12] HP Moonshot system: a new class of server. `http://www.hp.com/go/moonshot`, Accessed 9 July 2014.

[13] Intel Corporation. Intel Xeon Phi coprocessor system software developers guide. 2012. IBL Doc ID 488596.

[14] F. X. Lin, Z. Wang, and L. Zhong. K2: a mobile operating system for heterogeneous coherence domains. In *ASPLOS '14: Proc. Conference on Architectural Support for Programming Languages and Operating Systems*, pages 285–300, 2014.

[15] M. M. K. Martin, M. D. Hill, and D. J. Sorin. Why on-chip cache coherence is here to stay. *Commun. ACM*, 55(7):78–89, 2012.

[16] J. M. Mellor-Crummey and M. L. Scott. Algorithms for scalable synchronization on shared-memory multiprocessors. *ACM Transactions on Computer Systems*, 9(1):21–65, Feb. 1991.

[17] D. Narayanan and O. Hodson. Whole-system persistence. In *ASPLOS '12: Proc. Conference on Architectural Support for Programming Languages and Operating Systems*, pages 401–410, 2012.

[18] S. Novaković, A. Daglis, E. Bugnion, B. Falsafi, and B. Grot. Scale-Out NUMA. In *ASPLOS '14: Proc. 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, 2014.

[19] D. A. Patterson, K. Asanovic, A. B. Brown, R. Fromm, J. Golbus, B. Gribstad, K. Keeton, C. E. Kozyrakis, D. B. Martin, S. Perissakis, R. Thomas, N. Treuhaft, and K. A. Yelick. Intelligent RAM (IRAM): the industrial setting, applications and architectures. In *Proceedings 1997 International Conference on Computer Design: VLSI in Computers & Processors, ICCD '97, Austin, Texas, USA, October 12-15, 1997*, pages 2–7, 1997.

[20] S. Pelley, P. M. Chen, and T. F. Wenisch. Memory persistency. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, pages 265–276, Piscataway, NJ, USA, 2014. IEEE Press.

[21] L. Pina, L. Veiga, and M. Hicks. Rubah: DSU for Java on a stock JVM. In *OOPSLA '14: Proc. Conference on Object-Oriented Programming Languages, Systems, and Applications*, Oct. 2014.

[22] A. Rao. SeaMicro SM10000 system overview, June 2010. `http://www.seamicro.com/sites/default/files/SM10000SystemOverview.pdf`.

[23] D. Wentzlaff and A. Agarwal. Factored operating systems (FOS): the case for a scalable operating system for multicores. *SIGOPS Oper. Syst. Rev.*, 43(2):76–85, Apr. 2009.

Transactional Memory Today[1]

Michael Scott
Computer Science Department
University of Rochester, NY, USA
`scott@cs.rochester.edu`

It was an honor and a privilege to be asked to participate in the celebration, at PODC 2014, of Maurice Herlihy's many contributions to the field of distributed computing—and specifically, to address the topic of transactional memory, which has been a key component of my own research for the past decade or so.

When introducing transactional memory ("TM") to people outside the field, I describe it as a sort of magical merger of two essential ideas, at different levels of abstraction. First, at the language level, TM allows the programmer to specify that certain blocks of code should be atomic without saying how to *make* them atomic. Second, at the implementation level, TM uses speculation (much of the time, at least) to execute atomic blocks in parallel whenever possible. Each dynamic execution of an atomic block is known as a *transaction*. The implementation guesses that concurrent transactions will be mutually independent. It then monitors their execution, backing out and retrying if (and hopefully only if) they are discovered to conflict with one another.

The second of these ideas—the speculative implementation—was the focus of the original TM paper, co-authored by Maurice with Eliot Moss [22]. The first idea—the simplified model of language-level atomicity—is also due largely to Maurice, but was a somewhat later development.

[1]Based on remarks delivered at the Maurice Herlihy 60th Birthday Celebration, Paris, France, July 2014

1 Motivation

To understand the original motivation for transactional memory, consider the typical method of a nonblocking concurrent data structure. The code is likely to begin with a "planning phase" that peruses the current state of the structure, figuring out the operation it wants to perform, and initializing data—some thread-private, some visible to other threads—to describe that operation. At some point, a critical *linearizing instruction* transitions the operation from "desired" to "performed." In some cases, the identity of the linearizing instruction is obvious in the source code; in others it can be determined only by reasoning in hindsight over the history of the structure. Finally, the method performs whatever "cleanup" is required to maintain long-term structural invariants. Nonblocking progress is guaranteed because the planning phase has no effect on the logical state of the structure, the linearizing instruction is atomic, and the cleanup phase can be performed by any thread—not just the one that called the original operation.

Two issues make methods of this sort very difficult to devise. The first is the need to effect the transition from "desired" to "performed" with a single atomic instruction. The second is the need to plan correctly in the face of concurrent changes by other threads. By contrast, an algorithm that uses a coarse-grained lock faces neither of these issues: writes by other threads will never occur in the middle of its reads; reads by other threads will never occur in the middle of its writes.

2 The Original Paper

While Maurice is largely celebrated for his theoretical contributions, the original TM paper was published at ISCA, the leading architecture conference, and was very much a hardware proposal. We can see this in the subtitle—"Architectural Support for Lock-Free Data Structures"—and the abstract: "[TM is] ... intended to make lock-free synchronization as efficient (and easy to use) as conventional techniques based on mutual exclusion."

The core idea is simple: a transaction runs almost the same code as a coarse-grain critical section, but with special load and store instructions, and without the actual lock. The special instructions allow the hardware to track conflicts between concurrent transactions. A special end-of-transaction commit instruction will succeed (and make transactionally written values visible to other threads) only if no concurrent conflicting transaction has committed. Here "conflict" means that one transaction writes a cache line that another reads or writes. Within a transaction, a special validate instruction allows code to determine whether it still has a chance to commit successfully—and in particular, whether the loads it has performed to date remain mutually consistent. In response to a failed validate or commit, the typical transaction will loop back (in software) and start over.

Looking back with the perspective of more than 20 years, the original TM paper appears remarkably prescient. Elision of coarse-grain locks remains the principal use case for TM today, though the resulting algorithms are "lock-free" only in the informal sense of "no application-level locks," not in the sense of livelock-free. Like almost all contemporary TM hardware, Herlihy & Moss (H&M) TM was also a "best-effort-only" proposal: a transaction could fail due not only to conflict or to overflow of hardware buffers, but to a variety of other conditions—notably external interrupts or the end of a scheduling quantum. Software must be prepared to fall back to a coarse-grain lock (or some other hybrid method) in the event of repeated failures.

Speculative state (the record of special loads and stores) in the H&M proposal was kept in a special "transactional cache" alongside the "regular" cache (in 1993, processors generally did not have multiple cache layers). This scheme is still considered viable today, though commercial offerings vary: the Intel Haswell processor leverages the regular L1 data cache [40]; Sun's unreleased Rock machine used the processor store buffer [10]; IBM's zEC12 uses per-core private L2s [25].

In contrast with current commercial implementations, H&M proposed a "responder wins" coherence strategy: if transaction *A* requested a cache line that had already been speculatively read or written by concurrent transaction *B*, *B* would "win" and *A* would be forced to abort. Current machines generally do the opposite: "responder loses"—kill *B* and let *A* continue. Responder-loses has the advantage of compatibility with existing coherence protocols, but responder-wins turns out to be considerably less vulnerable to livelock. Nested transactions were not considered by H&M, but current commercial offerings address them only by counting, and subsuming the inner transactions in the outer: there is no way to abort and retry an inner transaction while keeping the outer one live.

Perhaps the most obvious difference between H&M and current TM is that the latter uses "modal" execution, rather than special loads and stores: in the wake of a special tm-start instruction, all ordinary memory accesses are considered speculative. In keeping with the technology of the day, H&M also assumed sequential consistency; modern machines must generally arrange for tm-start and commit instructions to incorporate memory barriers.

While designers of modern systems—both hardware and software—think of speculation as a fundamental design principle—comparable to caching in its degree of generality—this principle was nowhere near as widely recognized in 1993. In hindsight, the H&M paper (which doesn't even mention the term) can be seen not only as the seminal work on TM, but also as a seminal work in the history of speculation.

3 Subsequent Development

Within the architecture community, H&M TM was generally considered too ambitious for the hardware of the day, and was largely ignored for a decade. There was substantial uptake in the theory community, however, where TM-like semantics were incorporated into the notion of universal constructions [3, 5, 24, 28, 35]. In 1997, Shavit and Touitou coined the term "Software Transactional Memory," in a paper that shared with H&M the 2012 Dijkstra Prize [33].

And then came multicore. With the end of uniprocessor performance scaling, the difficulty of multithreaded programming became a sudden and pressing concern for researchers throughout academia and industry. And with advances in processor technology and transistor budgets, TM no longer looked so difficult to implement. Near-simultaneous breakthroughs in both software and hardware TM were announced by several groups in the early years of the 21st century.

Now, another decade on, perhaps a thousand TM papers have been published (including roughly a third of my own professional output). Plans are underway for the 10th annual ACM TRANSACT workshop. Hardware TM has been incorporated into multiple "real world" processors, including the Azul Vega 2 and 3 [7]; Sun Rock [10]; IBM Blue Gene/Q [36], zEnterprise EC12 [25], and Power8 [6]; and Intel Haswell [40]. Work on software TM has proven even more fruitful, at least from a publications perspective: there are many more viable implementation alternatives—and many more semantic subtleties—than anyone would have anticipated back in 2003. TM language extensions have become the synchronization mechanism of choice in the Haskell community [16], official extensions for C++ are currently in the works (a preliminary version [1] already ships in `gcc`), and research-quality extensions have been developed for a wide range of other languages.

4 Maurice's Contributions

Throughout the history of TM, Maurice has remained a major contributor. The paragraphs here touch on only a few of his many contributions. With colleagues at Sun, Maurice co-designed the DSTM system [18], one of the first software TMs with semantics rich enough—and overheads low enough—to be potentially acceptable in practice. Among its several contributions, DSTM introduced the notion of out-of-band *contention management*, a subject on which Maurice also collaborated with colleagues at EPFL [13, 14]. By separating safety and liveness, contention managers simplify both STM implementation and correctness proofs.

In 2005, Maurice collaborated with colleagues at Intel on mechanisms to virtualize hardware transactions, allowing them to survive both buffer overflows and context switches [30]. He also began a series of papers, with colleagues at Brown and Swarthmore, on transactions for energy efficiency [12]. With student Eric Koskinen, he introduced *transactional boosting* [20], which refines the notion of conflict to encompass the possibility that concurrent operations on abstract data types, performed within a transaction, may commute with one another at an abstract level—and thus be considered non-conflicting—even when they would appear to conflict at the level of loads and stores. With student Yossi Lev he explored support for debugging of transactional programs [21]. More recently, again with the team at Sun, he has explored the use of TM for memory management [11].

Perhaps most important, Maurice became a champion of the promise of transactions to simplify parallel programming—a promise he dubbed the "transactional manifesto" [19]. During a sabbatical at Microsoft Research in Cambridge, England, he collaborated with the Haskell team on their landmark exploration of *composability* [16]. Unlike locks, which require global reasoning to avoid or recover from deadlock, transactions can easily be combined to create larger atomic operations from smaller atomic pieces. While the benefits can certainly be oversold (and have been—though not by Maurice), composability represents a fundamental breakthrough in the creation of concurrent abstractions. Prudently employed, transactions can offer (most of) the performance of fine-grain locks with (most of) the convenience of coarse-grain locks.

5 Status and Challenges

Today hardware TM appears to have become a permanent addition to processor instruction sets. Run-time systems that use this hardware typically fall back to a global lock in the face of repeated conflict or overflow aborts. For the overflow case, hybrid systems that fall back to software TM may ultimately prove to be more appropriate. STM will also be required for TM programs on legacy hardware. The fastest STM implementations currently slow down critical sections (though not whole applications!) by factors of 3–5, and that number is unlikely to improve. With this present status as background, the future holds a host of open questions.

5.1 Usage Patterns

TM is not yet widely used. Most extant applications are actually written in Haskell, where the semantics are unusually rich but the implementation unusually slow. The most popular languages for research have been C and C++, but progress has been impeded, at least in part, by the lack of high quality benchmarks.

The biggest unknown remains the breadth of TM applicability. Transactions are clearly useful—from both a semantic and a performance perspective—for small operations on concurrent data structures. They are much less likely to be useful—at least from a performance perspective—for very large operations, which may overflow buffer limits in HTM, run slowly in STM, and experience high conflict rates in either case. No one is likely to write a web server that devotes a single large transaction to each incoming page request. Only experience will tell how large transactions can become and still run mostly in parallel.

When transactions *are* too big, and frequently conflict, programmers will need tools to help them identify the offending instructions and restructure their code for better performance. They will also need advances, in both theory and software engineering, to integrate transactions successfully into pre-existing lock-based applications.

5.2 Theory and Semantics

Beyond just atomicity, transactions need some form of condition synchronization, for operations that must wait for preconditions [16, 37]. There also appear to be cases in which a transaction needs some sort of "escape action" [29], to generate effects (or perhaps to observe outside state) in a way that is not fully isolated from action in other threads. In some cases, the application-level logic of a transaction may decide it needs to abort. If the transaction does not restart, but switches to some other code path, then information (the fact of the abort, at least) has "leaked" from code that "did not happen" [16]. Orthogonally, if large transactions prove useful in some applications, it may be desirable to parallelize them internally, and let the sub-threads share speculative state [4]. All these possibilities will require formalization.

A more fundamental question concerns the basic model of synchronization. While it is possible to define the behavior of transactions in terms of locks [27], with an explicit notion of abort and rollback, such an approach seems contrary to the claim that transactions are simpler than locks. An alternative is to make atomicity itself the fundamental concept [8], at which point the question arises: are aborts a part of the language-level semantics? It's appealing to leave them out, at least in the absence of a program-level abort operation, but it's not clear how such an approach would interact with operational semantics or with the definition of a data race.

For run-time–level semantics, it has been conventional to require that every transaction—even one that aborts—see a single, consistent memory state [15]. This requirement, unfortunately, is incompatible with implementations that "sandbox" transactions instead of continually checking for consistency, allowing doomed transactions to execute—at least for a little while—down logically impossible code paths. More flexible semantics might permit such "transactional zombies" while still ensuring forward progress [32].

5.3 Language and System Integration

For anyone building a TM language or system, the theory and semantic issues of the previous section are of course of central importance, but there are other issues as well. What should be the syntax of atomic blocks? Should there be atomic expressions? How should they interact with existing mechanisms like try blocks and exceptions? With locks?

What operations can be performed inside a transaction? Which of the standard library routines are on the list? If routines must be labeled as "transaction safe," does this become a "viral" annotation that propagates throughout a code base? How much of a large application must eschew transaction-unsafe operations?

In a similar vein, given the need to instrument loads and stores inside (but not outside) transactions, which subroutines must be "cloned"? How does the choice interact with separate compilation? How do we cope with the resulting "code bloat"?

Finally, what should be done about repeated aborts? Is fallback to a global lock acceptable, or do we need a hybrid HTM/STM system? Does the implementation need to adapt to observed abort patterns, avoiding fruitless speculation? What factors should influence adaptation? Should it be static or dynamic? Does it need to incorporate feedback from prior executions? How does it interact with scheduling?

5.4 Building and Using TM Hardware

With the spread of TM hardware, it will be increasingly important to use that hardware well. In addition to tuning and adapting, we may wish to restructure transactions that frequently overflow buffers. We might, for example—by hand or automatically—reduce a transaction's memory footprint by converting a read-only preamble into explicit (nontransactional) speculation [2, 39]. One of my students has recently suggested using advisory locks (acquired using nontransactional loads and stores) to serialize only the portions of transactions that actually conflict [38].

Much will depend on the evolution of hardware TM capabilities. Nontransactional (but immediate) loads and stores are currently available only on IBM Power machines, and there at heavy cost. Lightweight implementations would enable not only partial serialization but also ordered transactions (i.e., speculative parallelization of ordered iteration) and more effective hardware/software hybrids [9, 26]. As noted above, there have been suggestions for "responder-wins" coherence, virtualization, nesting, and condition synchronization. With richer semantics, it may also be desirable to "deconstruct" the hardware interface, so that features are available individually, and can be used for additional purposes [23, 34].

6 Concluding Thoughts

While the discussion above spans much of the history of transactional memory, and mentions many open questions, the coverage has of necessity been spotty, and the choice of citations idiosyncratic. Many, many important topics and papers have been left out. For a much more comprehensive overview of the field, interested readers should consult the book-length treatise of Harris, Larus, and Rajwar [17]. A briefer overview can be found in chapter 9 of my synchronization monograph [31].

My sincere thanks to Hagit Attiya, Shlomi Dolev, Rachid Guerraoui, and Nir Shavit for organizing the celebration of Maurice's 60th birthday, and for giving me the opportunity to participate. My thanks, as well, to Panagiota Fatourou and Jennifer Welch for arranging the subsequent write-ups for BEATCS and SIGACT News. Most of all, my thanks and admiration to Maurice Herlihy for his seminal contributions, not only to transactional memory, but to nonblocking algorithms, topological analysis, and so many other aspects of parallel and distributed computing.

References

[1] A.-R. Adl-Tabatabai, T. Shpeisman, and J. Gottschlich, editors. Draft Specification of Transaction Language Constructs for C++. Version 1.1, IBM, Intel, and Sun Microsystems, Feb. 2012.

[2] Y. Afek, H. Avni, and N. Shavit. Towards Consistency Oblivious Programming. In *Proc. of the 15th Intl. Conf. on Principles of Distributed Systems*, pages 65-79. Toulouse, France, Dec. 2011.

[3] Y. Afek, D. Dauber, and D. Touitou. Wait-Free Made Fast. In *Proc. of the 27th ACM Symp. on Theory of Computing*, 1995.

[4] K. Agrawal, J. Fineman, and J. Sukha. Nested Parallelism in Transactional Memory. In *Proc. of the 13th ACM Symp. on Principles and Practice of Parallel Programming*, Salt Lake City, UT, Feb. 2008.

[5] G. Barnes. A Method for Implementing Lock-Free Shared Data Structures. In *Proc. of the 5th ACM Symp. on Parallel Algorithms and Architectures*, Velen, Germany, June–July 1993.

[6] H. W. Cain, B. Frey, D. Williams, M. M. Michael, C. May, and H. Le. Robust Architectural Support for Transactional Memory in the Power Architecture. In *Proc. of the 40th Intl. Symp. on Computer Architecture*, Tel Aviv, Israel, June 2013.

[7] C. Click Jr. And now some Hardware Transactional Memory comments. Author's Blog, Azul Systems, Feb. 2009. `blogs.azulsystems.com/cliff/2009/02/and-now-some-hardware-transactional-memory-comments.html`.

[8] L. Dalessandro, M. L. Scott, and M. F. Spear. Transactions as the Foundation of a Memory Consistency Model. In *Proc. of the 24th Intl. Symp. on Distributed Computing*, Cambridge, MA, Sept. 2010. Earlier but expanded version available as TR 959, Dept. of Computer Science, Univ. of Rochester, July 2010.

[9] L. Dalessandro, F. Carouge, S. White, Y. Lev, M. Moir, M. L. Scott, and M. F. Spear. Hybrid NOrec: A Case Study in the Effectiveness of Best Effort Hardware Transactional Memory. In *Proc. of the 16th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, Newport Beach, CA, Mar. 2011.

[10] D. Dice, Y. Lev, M. Moir, and D. Nussbaum. Early Experience with a Commercial Hardware Transactional Memory Implementation. In *Proc. of the 14th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems*, Washington, DC, Mar. 2009.

[11] A. Dragojević, M. Herlihy, Y. Lev, and M. Moir. On The Power of Hardware Transactional Memory to Simplify Memory Management. In *Proc. of the 30th ACM Symp. on Principles of Distributed Computing*, San Jose, CA, June 2011.

[12] C. Ferri, A. Viescas, T. Moreshet, I. Bahar, and M. Herlihy. Energy Implications of Transactional Memory for Embedded Architectures. In *Wkshp. on Exploiting Parallelism with Transactional Memory and Other Hardware Assisted Methods (EPHAM)*, Boston, MA, Apr. 2008. In conjunction with *CGO*.

[13] R. Guerraoui, M. Herlihy, and B. Pochon. Polymorphic Contention Management in SXM. In *Proc. of the 19th Intl. Symp. on Distributed Computing*, Cracow, Poland, Sept. 2005.

[14] R. Guerraoui, M. Herlihy, and B. Pochon. Toward a Theory of Transactional Contention Managers. In *Proc. of the 24th ACM Symp. on Principles of Distributed Computing*, Las Vegas, NV, Aug. 2005.

[15] R. Guerraoui and M. Kapałka. On the Correctness of Transactional Memory. In *Proc. of the 13th ACM Symp. on Principles and Practice of Parallel Programming*, Salt Lake City, UT, Feb. 2008.

[16] T. Harris, S. Marlow, S. Peyton Jones, and M. Herlihy. Composable Memory Transactions. In *Proc. of the 10th ACM Symp. on Principles and Practice of Parallel Programming*, Chicago, IL, June 2005.

[17] T. Harris, J. R. Larus, and R. Rajwar. *Transactional Memory*, Synthesis Lectures on Computer Architecture. Morgan & Claypool, second edition, 2010.

[18] M. Herlihy, V. Luchangco, M. Moir, and W. N. Scherer III. Software Transactional Memory for Dynamic-sized Data Structures. In *Proc. of the 22nd ACM Symp. on Principles of Distributed Computing*, Boston, MA, July 2003.

[19] M. Herlihy. The Transactional Manifesto: Software Engineering and Non-blocking Synchronization. In *Invited keynote address, SIGPLAN 2005 Conf. on Programming Language Design and Implementation*, Chicago, IL, June 2005.

[20] M. Herlihy and E. Koskinen. Transactional Boosting: A Methodology for Highly-Concurrent Transactional Objects. In *Proc. of the 13th ACM Symp. on Principles and Practice of Parallel Programming*, Salt Lake City, UT, Feb. 2008.

[21] M. Herlihy and Y. Lev. tm_db: A Generic Debugging Library for Transactional Programs. In *Proc. of the 18th Intl. Conf. on Parallel Architectures and Compilation Techniques*, Raleigh, NC, Sept. 2009.

[22] M. Herlihy and J. E. B. Moss. Transactional Memory: Architectural Support for Lock-Free Data Structures. In *Proc. of the 20th Intl. Symp. on Computer Architecture*, San Diego, CA, May 1993. Expanded version available as CRL 92/07, DEC Cambridge Research Laboratory, Dec. 1992.

[23] M. D. Hill, D. Hower, K. E. Moore, M. M. Swift, H. Volos, and D. A. Wood. A Case for Deconstructing Hardware Transactional Memory Systems. Technical Report 1594, Dept. of Computer Sciences, Univ. of Wisconsin–Madison, June 2007.

[24] A. Israeli and L. Rappoport. Disjoint-Access Parallel Implementations of Strong Shared Memory Primitives. In *Proc. of the 13th ACM Symp. on Principles of Distributed Computing*, Los Angeles, CA, Aug. 1994.

[25] C. Jacobi, T. Slegel, and D. Greiner. Transactional Memory Architecture and Implementation for IBM System z. In *Proc. of the 45th Intl. Symp. on Microarchitecture*, Vancouver, BC, Canada, Dec. 2012.

[26] A. Matveev and N. Shavit. Reduced Hardware Transactions: A New Approach to Hybrid Transactional Memory. In *Proc. of the 25th ACM Symp. on Parallelism in Algorithms and Architectures*, Montreal, PQ, Canada, July 2013.

[27] V. Menon, S. Balensiefer, T. Shpeisman, A.-R. Adl-Tabatabai, R. L. Hudson, B. Saha, and A. Welc. Practical Weak-Atomicity Semantics for Java STM. In *Proc. of the 20th ACM Symp. on Parallelism in Algorithms and Architectures*, Munich, Germany, June 2008.

[28] M. Moir. Transparent Support for Wait-Free Transactions. In *Proc. of the 11th Intl. Wkshp. on Distributed Algorithms*, 1997.

[29] Y. Ni, V. S. Menon, A.-R. Adl-Tabatabai, A. L. Hosking, R. L. Hudson, J. E. B. Moss, B. Saha, and T. Shpeisman. Open Nesting in Software Transactional Memory. In *Proc. of the 12th ACM Symp. on Principles and Practice of Parallel Programming*, San Jose, CA, Mar. 2007.

[30] R. Rajwar, M. Herlihy, and K. Lai. Virtualizing Transactional Memory. In *Proc. of the 32nd Intl. Symp. on Computer Architecture*, Madison, WI, June 2005.

[31] M. L. Scott. *Shared-Memory Synchronization*. Morgan & Claypool, 2013.

[32] M. L. Scott. Transactional Semantics with Zombies. In *Invited keynote address, 6th Wkshp. on the Theory of Transactional Memory*, Paris, France, July 2014.

[33] N. Shavit and D. Touitou. Software Transactional Memory. *Distributed Computing*, 10(2):99-116, Feb. 1997. Originally presented at the *14th ACM Symp. on Principles of Distributed Computing*, Aug. 1995.

[34] A. Shriraman, S. Dwarkadas, and M. L. Scott. Implementation Tradeoffs in the Design of Flexible Transactional Memory Support. *Journal of Parallel and Distributed Computing*, 70(10):1068-1084, Oct. 2010.

[35] J. Turek, D. Shasha, and S. Prakash. Locking Without Blocking: Making Lock Based Concurrent Data Structure Algorithms Nonblocking. In *Proc. of the 11th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems*, Vancouver, BC, Canada, Aug. 1992.

[36] A. Wang, M. Gaudet, P. Wu, J. N. Amaral, M. Ohmacht, C. Barton, R. Silvera, and M. Michael. Evaluation of Blue Gene/Q Hardware Support for Transactional Memories. In *Proc. of the 21st Intl. Conf. on Parallel Architectures and Compilation Techniques*, Minneapolis, MN, Sept. 2012.

[37] C. Wang, Y. Liu, and M. Spear. Transaction-Friendly Condition Variables. In *Proc. of the 26th ACM Symp. on Parallelism in Algorithms and Architectures*, Prague, Czech Republic, June 2014.

[38] L. Xiang and M. L. Scott. Conflict Reduction in Hardware Transactions Using Advisory Locks. In *Proc. of the 27th ACM Symp. on Parallelism in Algorithms and Architectures*, Portland, OR, June 2015.

[39] L. Xiang and M. L. Scott. Software Partitioning of Hardware Transactions. In *Proc. of the 20th PPoPP*, San Francisco, CA, Feb. 2015.

[40] R. M. Yoo, C. J. Hughes, K. Lai, and R. Rajwar. Performance Evaluation of Intel Transactional Synchronization. In x. f. H.-P. Computing, editor, *Proc., SC2013: High Performance Computing, Networking, Storage and Analysis*, pages 1-11. Denver, Colorado, Nov. 2013.

The Distributed Computing Column

BY

Stefan Schmid

TU Berlin
T-Labs, Germany

stefan.schmid@tu-berlin.de

Fault-tolerant Distributed Systems in Hardware

Danny Dolev (Hebrew University of Jerusalem)
Matthias Függer (MPI for Informatics)
Christoph Lenzen (MPI for Informatics)
Ulrich Schmid (TU Vienna)
Andreas Steininger (TU Vienna)

Very large-scale integrated (VLSI) hardware designs can be seen as distributed systems at several levels of abstraction: from the cores in a multicore architecture down to the Boolean gates in its circuit implementation, hardware designs comprise of interacting computing nodes with non-negligible communication delays. The resulting similarities to classic large-scale distributed systems become even more accented in mission critical hardware designs that are required to operate correctly in the presence of component failures.

We advocate to act on this observation and treat fault-tolerant hardware design as the task of devising suitable distributed algorithms. By means of problems related to clock generation and distribution, we show that (i) design and analysis techniques from distributed computing can provide new and provably correct mission critical hardware solutions and (ii) studying such systems reveals many interesting and challenging open problems for distributed computing.

1 Introduction

Very large-scale integrated (VLSI) circuits bear several similarities with the systems studied by the distributed computing community:

- They are formed by an increasing number of interacting computing nodes.
- Communication delays are not negligible.
- The cost of communication, such as area and power consumption, is not negligible.

In fact, this view is correct at different levels of abstraction. We will elaborate on two such levels that significantly differ from each other with respect to the computational power of the system's nodes, their means of communication, and the problems they solve.

(I) Viewed from a low-level perspective, every digital circuit is a network of logic gates with delays, which continuously compute their current output state from their input history and interact via binary-valued, continuous-time signals. We stress the fact, however, that this is already a (convenient) abstraction, as real gates are electronic devices that process analog (continuous-valued) signals: A signal that is above a certain threshold voltage is considered high, otherwise low. Whereas analog signals (like a good clock signal) that swing fast from low voltages to high voltages are represented reasonably well by the resulting digital abstraction, this is obviously not the case for every signal: Just consider an analog signal that stays very close to the threshold voltage and just, e.g., due to very small noise, occasionally crosses it. It will turn out that this modeling inaccuracy causes serious problems both for synchronization and fault-tolerance, which must be considered explicitly.

Analogously to distributed computing, there are two fundamentally different ways to design digital circuits (i.e., algorithms in hardware), which correspond to synchronous and asynchronous algorithms in distributed computing.

The classic design paradigm relies on the synchronous computational model. It abstracts away the timing of gates and interconnects by considering gate outputs only at predetermined instants dictated by a central periodic clock signal. While this approach allows the designer to solely concentrate on the stable outputs of a network of gates, it relies critically on the guarantee that all signals have settled and all transients have vanished at the occurrence of the next clock transition. Inherently, such designs run at the speed of the clock period that is determined from worst-case bounds on gate and interconnect delays. Due to increasingly pronounced delay variations [52, 84] this results in highly conservative bounds and thus in considerable performance loss.

In contrast, designs that do not rely on the convenient discrete time abstraction provided by a clock signal are called *clockless* or asynchronous.[1] Such circuits must rely on different techniques to enforce some ordering between signal transitions. Suitable techniques range from aggressively timed circuits that explicitly use information on the delays along certain paths [80, 91, 97] to circuits that tolerate (certain) delay variations by means of some forms of handshaking. Prominent examples of the latter are *delay insensitive* (DI) circuits [72], speed-independent (SI) circuits and *quasi-delay insensitive* (QDI) circuits [74, 75]. While DI circuits are guaranteed to behave correctly in the presence of arbitrary gate and interconnect delay variations, SI resp. QDI circuits assume that all resp. certain signal forks in the interconnect are isochronic, i.e., have roughly equal propagation delays along all their fork teeth.

The robustness to delay variations in DI circuits comes at a high price, however: Martin [73] showed that the expressiveness of circuits that are DI at gate-level is severely limited. In fact, the only two-input gate allowed in such circuits is the C-Element, which is an AND gate for signal transitions; it produces a, say, rising transition at its output when it observed a rising transition at all its inputs. This clearly restricts the usability of DI circuits for real applications.

SI and QDI circuits, on the other hand, are Turing-complete [70]. Intuitively, the isochronic fork assumption guarantees that a gate whose output drives an isochronic fork implicitly performs a handshake with all its successor gates while just handshaking with one of its successors. A precise characterization of the conditions on the propagation delay that have to hold on certain paths in SI circuits was derived in [56].

(II) With the increasing number of computing nodes in *System-on-Chip* (SoC) and *Network-on-Chip* (NoC) architectures, problems that used to be relevant only in large-scale computer networks also become relevant within a single chip. Examples range from establishing a common time base over data communication and routing between nodes to load balancing.

In the hardware context, establishing a common time base among all nodes is of particular interest, because this sets the base for a synchronous computational model: Rather than being implemented entirely clockless, higher-level services like routing and load balancing could then also exploit synchrony properties. Unfortunately, however, the GHz clock speeds and sizes of modern VLSI circuits make it increasingly difficult to distribute a central clock signal throughout the whole circuit [43, 96]. Modern SoCs and NoCs hence typically rely on the *globally asynchronous locally synchronous* (GALS) approach [14], where different parts of a chip are clocked by different clock sources. Using inde-

[1]We will use the term "clockless" in the following, as such circuits do not always allow for arbitrarily large and unknown delays.

pendent and hence unsynchronized clock domains would give away the advantages of global synchrony and also requires non-synchronous cross-domain communication mechanisms or synchronizers [59, 58]. A promising alternative is mesochronous clocking [79] (sometimes also called *multi-synchronous* clocking [93]) as it guarantees some upper bound on the skew between clock domains. In this article, we will thus focus on discussing methods for providing a common time in GALS architectures.

Fault-tolerance. Besides an increasing number of components and non-negligible communication costs both at gate and system level, there is a further trend that advocates the application of distributed computing methods for designing VLSI chips: the increasing susceptibility to failures. Indeed, fault-tolerance has been identified as a key challenge in the International Technology Roadmap for Semiconductors [52] for years. Besides the increasing susceptibility of nanometer VLSI technology to permanent failures caused by manufacturing process variations and excessive operating conditions (supply voltage, temperature) [62], steadily decreasing feature sizes and signal voltage swings also led to dramatically increased transient failure rates [16], caused by ionizing particles hitting the junction of transistors [6], electro-magnetic cross-talk between signal wires and supply-voltage variations caused by simultaneous switching activities [78, 85].

Unfortunately, even relatively simple faults unveil the very limited ability of the convenient digital signal abstraction to properly describe reality. For example, an out-of-spec output driver of a gate that drives a fork to two different gate inputs may be able to reach the threshold voltage at one input but not at the other, causing those to interpret the gate output inconsistently. Similarly, a *single-event transient* (SET) [6] caused by an ionizing particle that hits the output driver of such a gate may be visible at one input but not at the other, depending on the latter's input capacitances. It is hence apparent that classic benign failure models from distributed computing, where a message is either lost or transmitted correctly, do not cover such faults. In fact, faulty gates have to be assumed to potentially behave arbitrarily, i.e., Byzantine [86].

While there is a huge body of work on fault mitigation techniques at the technological level (like *silicon-on-insulator* (SOI) technology [76]) and gate level (like the SET-tolerant DICE latch [83]), keeping the overall error rate acceptable [89, 22] despite the tremendously increasing number of gates/cores on a single chip demands for additional architectural solutions. At the same time, solutions that require central knowledge of the current system state (i) become infeasible due to the high communication costs and (ii) would themselves form a single point of failure. Algorithmic solutions that use only local knowledge, studied by the distributed computing community for decades, are hence promising in this context.

Classic architectural fault-tolerance approaches [87] like *Dual Modular Redundancy* (DMR) and *Triple Modular Redundancy* (TMR) fail in absence of a global time base, as it becomes unclear over which values to vote. Jang and Martin [53] adapted this method to QDI designs and applied it to build a microcontroller tolerating certain transient faults [54], in particular, *single-event upsets* (SEUs), where a state-holding device may flip its state due to a particle hit. Their solution duplicates each gate and adds two succeeding cross-coupled C-Elements whose inputs are connected to the outputs of the duplicated gates. In case of a spurious state transition of one of the duplicated gates, both C-Elements do not propagate the spurious output value until it is matched by the other gate's output also (which can be proved to eventually happen). While this method tolerates SEUs, it neither allows to tolerate SETs nor permanent faults.

Tolerating such faults necessarily requires to extend the circuit's control logic not to wait for all of its predecessors' outputs. In contrast to the AND-causality semantics of the C-Element, this requires OR-causality semantics. Interestingly, there has been research in this direction in a different context: In certain cases, a Boolean function's value can already be determined from a subset of its parameters. This fact can be used to speed up clockless circuits [17, 95]. Instead of waiting for all of a module's inputs to arrive, the module waits until its outcome is determined and then immediately produces a new output value. Care must be taken not to mix up current input data with lately arriving input data from a previous computation, however. The approach thus requires additional timing assumptions and ways to memorize which input values a module already took into account when computing its output.

A similar strategy can also be applied to design clockless circuits that tolerate a certain fraction of its input nodes to fail permanently. In particular, it has also been employed in the DARTS Byzantine fault-tolerant clock generation approach [49, 44] for mesochronous GALS architectures, which comprises a network of interconnected OR-causality gates whose outputs generate tightly synchronized clock pulses. The approach is discussed in more detail in Section 3.1.3.

The limit of the fault-tolerant hardware solutions discussed above is that they allow only a certain subset of the components to fail. Even if these components start to operate according to their specification again later on, their state may remain corrupted, preventing them from recommencing correct operation. For massive transient failures, which are likely to occur e.g. in space missions and may corrupt the entire system state, the above solutions are not adequate. To attack this problem, the concept of self-stabilizing distributed algorithms [33] has successfully been applied to digital circuits. A self-stabilizing circuit is guaranteed to eventually behave correctly again after its state was arbitrarily corrupted. For example, a self-stabilizing token passing algorithm implemented in clockless hardware was presented in [11], and S. Dolev and Haviv [34] built and proved correct a self-stabilizing microprocessor.

From a robustness point of view, a combination of resilience to permanent faults and self-stabilization is most desirable: Appropriate solutions operate correctly in the presence of not too many permanent faults and even recover from massive transient disruptions. In [27, 28] the fault-tolerant and self-stabilizing pulse generation algorithm FATAL and its hardware implementation were presented and proven correct. This solution is discussed in more detail in Section 3.1.4.

Additional challenges. While we highlighted major similarities between VLSI designs and classic distributed systems, there are also important differences. In most cases, these advise against a naive implementation of distributed algorithms in hardware.

First and foremost, this is the absence of computationally powerful atomic actions at the gate level in clockless circuits: Explicit means such as handshaking must be employed to synchronize activities in such a circuit, which is not only costly but also imperfect in terms of synchronization accuracy. This, in turn, is also a source for a unique problem called *metastability*, which arises when a circuit must handle input signals that bear no well-defined timing relation to its own state transitions. Consider a simple R/W register in a shared memory system that my be read by a processor at the time it is written by some other processor. Distributed computing models based on regular or even atomic registers assume that either the previous or the newly written value is returned by the read. In reality, the situation is even worse than assumed for safe registers, which allow an arbitrary return value in this case: The register may reach a metastable state, which moves its output voltage to a non-digital level for an unpredictable time!

Another unique problem arises from the imperfect coverage of the digital abstraction for analog signals in the case of failures. In distributed computing, Byzantine behavior is considered the worst a component can do to a system. Unfortunately, in digital circuits, generating an arbitrary binary-valued continuous-time signal is not the worst behavior of a component. Rather, a component may produce an arbitrary analog signal on its output, e.g., an output voltage that remains very close to the threshold voltage arbitrarily long, which is actually one manifestation of metastability (creeping metastability) [7, 13]. We will discuss these issues in more detail in Section 2.

Structure of this article. We start in Section 2 with a discussion on the peculiarities of SoCs in comparison to classic distributed systems, and the challenges arising in the definition of an appropriate distributed system model. In Section 3, we discuss the problem of obtaining a common time base for multi-synchronous GALS architectures, which is both fundamental to the solution of other problems and exemplarily captures the challenges of adapting distributed algorithms for use on a chip. The problem is divided into three parts: (i) Section 3.1 discusses the problem of generating synchronous pulses. (ii) Section 3.2 deals with establishing local counters. Here we provide more technical details, with the primary goal of illustrating how techniques from distributed computing find application in VLSI design. (iii) Section 3.3 finally is concerned with distributing the clock over a wider area. The work is concluded in Section 4.

2 Modeling Issues

While we pointed out the similarities of VLSI circuits and fault-tolerant distributed systems in Section 1, a simple migration of classic solutions in distributed computing is not favorable and most of the time even infeasible. The most prominent obstacles are:

(i) Gates continuously compute their output state from their input states. They generate events, i.e., binary transitions, in a fully parallel way and are capable of very simple computations, such as the logical AND of two binary inputs, only. Any kind of event sequencing and atomic actions that group several binary transitions into more powerful computations requires explicit synchronization between the concurrently operating gates, e.g., by handshaking or timing assumptions. Note that this includes even "simple" computations such as the sum or the product.

(ii) Communication and computation is costly, especially if the proposed solution is meant to "only" provide low-level services to the application running on top. For example, clock generation algorithms must not use more than a few wires between nodes to be able to compete with classic clock distribution networks. Exchange of data, even a few bits, requires parallel or serial coding and decoding logic and thus typically cannot be afforded for low-level services. Rather, solutions must resort to signaling a few status bits only. Exchanging data of more than, say, 32 bits, is usually also difficult for high-level services.

(iii) Non-digital low-level effects must be taken into account. Every binary valued model necessarily abstracts from the analog signals in real gate implementations. While it is perfectly valid to resort to binary abstractions most of the time, these models come to their limits when synchronization and failures enter the picture: Marino [71] showed that any bistable element, e.g., a binary memory cell, may get stuck arbitrarily long in between its two stable states. This may result in spontaneous, unpredictably late transitions on its output, and even in an inconsistently perceived input at multiple successor gates. While classic designs prevent these scenarios by ensuring that certain timing constraints on the input signals are not violated, this is not always possible and inherently cannot be assumed in fault-tolerant circuits.

In order to be able to predict the behavior of a circuit and reason formally about its correctness and performance at early design stages, i.e., before fabrication, a suitable circuit model is required. Clearly, any such model should be sufficiently simple to support fast predictions and formal analysis, while at the same time ensure that the results reflect reality sufficiently accurate. We will briefly sketch common approaches.

Discrete time state machines. Synchronously clocked circuits of any kind can be modeled by a discrete-time, discrete-value synchronous communicating state machine model, for which very efficient and fast timing prediction and formal analysis tools are available. Unfortunately, this abstraction does not cover all existing circuits. This is obvious for clockless circuits, but also for the timing analysis of clocked circuits, which is mandatory for validating the clock timing requirements for justifying the synchronous abstraction. The latter is particularly important with the advent of aggressively timed high-speed synchronous circuits, where clock speed must be traded against the increasing rate of manufacturing errors and other sources of timing failures. In that case one has to resort to continuous time models.

Continuous time models. Arguably, the most accurate models used in circuit design today are fully-fledged continuous-time analog valued ones as, e.g., instantiated by Spice [81]. However, excessive analog simulation times prohibit its use for analyzing more than a fairly small part of a VLSI circuit, over fairly short periods of simulated real-time. Discrete-value models, and specifically binary-valued ones, are hence an attractive alternative. Modern digital design approaches e.g. based on description languages such as VHDL [5] incorporate digital timing simulators that are typically based on zero-time Boolean gates interconnected by non-zero-delay channels. Popular instances are simple pure (i.e., constant-delay) and inertial delay channels (i.e., constant-delay channels that suppress short input pulses) [94], but also more elaborate ones like the Delay Degradation Model (DDM) [8] or the empirical Synopsis CCS Timing model [92]. Continuous time, discrete-value models can be either state-based or trace-based, as detailed in the following.

Clockless, state-based models. At the gate level, clockless circuits are typically modeled by a binary state vector representing the global circuit state and, potentially time-annotated, guard-action pairs [4, 72] that describe the gates. An execution, i.e., signal trace, of the circuit is a sequence of global states over time

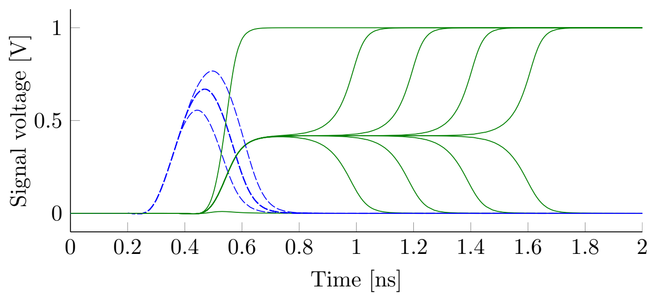


Figure 1: Analog simulation traces of a storage loop (e.g., a 2-input OR gate with the output fed back to one input) that can persistently memorize a high state of its (other) input. The blue dashed curves (the medium line actually corresponding to 8 almost identical pulses) show the real analog shape of short input pulse signals of different duration, the solid green ones give the corresponding output signals of the storage loop.

generated by a parallel execution of the guard-action pairs. Note that executions need not necessarily be unique, accounting for potential delay variations within the circuit. Like the models used in classic distributed computing, such as the Alur-Dill Timed Automata [2] and the Timed I/O Automata by Keynar et al. [55], these models all act on the explicitly given state-space of the underlying system.

While this view serves as a good level of abstraction in clockless circuits operated in closed environments, it comes to its limits when signals do not necessarily stabilize before they change again. For instance, consider a gate that produces a very short low-high-low pulse at its output: In reality, this means that the gate driver circuitry has only started to drive the output signal to high when it is turned off again. This results in a short, potentially non-full-swing waveform that may quite unpredictably affect the subsequent gate. An example is shown in the blue dashed pulse shape in Figure 1.

In this example, the subsequent gate is a *memory flag*, which persistently memorizes a high state at its input, until it is reset again to 0. A straightforward implementation is given by a *storage loop*, e.g. consisting of a 2-input OR gate with its output fed back to its other input. The solid green lines represent the output signals of the storage loop corresponding to the blue dashed inputs. The largest input pulse causes the storage loop to flip to the high state immediately, while the smallest one does not cause any effect on the initial low output. The medium input pulse, however, which actually represents 8 different ones that differ only marginally from each other, causes the loop to enter a metastable state: The input pulses are too short to allow the storage loop, which has some short but nonzero delay, to settle to a stable value. Depending on minor variations of the input pulses, the metastable state eventually resolves after some unpredictable and possibly large resolution time. The memory flag does not operate as intended during this time, possibly affecting the downstream logic in a similar way.

Such situations cannot be prevented from happening in open environments in which a circuit cannot control all of its inputs. The same holds in fault-tolerant circuits, where the signals provided by faulty nodes may be arbitrary. Thus they must be reasonably covered by an appropriate digital circuit model. Unfortunately, however, this is not the case for any model used in modern timing circuit simulators today. Besides the complete lack of modeling and analysis support for fault-tolerance, it was shown in [48] that none of the existing analytic channel models, including the popular pure and inertial delay channels [94] as well as the DDM model [8], faithfully model the propagation of short pulses in physical circuits. Specifically, it has been shown that these models are inconsistent with possibility and impossibility results concerning the implementation of a one-shot inertial delay channel: a channel that, like an inertial delay channel, suppresses short pulses, but is required to work correctly only in presence of a single input pulse (one-shot).

Recently, however, a candidate delay model [47] based on *involution channels* has been proposed that does not have this shortcoming: It is not only consistent with the theoretical results on the one-shot inertial delay problem [48], but also achieves a promising level of accuracy [82]. As a consequence, there is a prospect of eventually identifying a tractable delay model that can form the basis for a comprehensive modeling framework for digital circuits.

Clockless, trace-based models. Existing frameworks for designing clockless digital circuits also have shortcomings at higher abstraction levels. In particular, we are not aware of any modeling framework (except the one we proposed in [27]) that supports fault-tolerance. Instead of blowing up the state space of existing state-based models – like Alur-Dill Timed Automata [2], Lamport's TLA [63], Timed IO Automatons by Keynar et al. [55], discrete abstractions for hybrid systems [3], or state-space-based control theory [64] – with error states and/or using non-determinism or probabilistic state transitions for modeling faults, it advocates the use of solely trace-based models, which focus on the externally visible behavior of a circuit only.

Examples of earlier trace-based approaches are Ebergen's trace theory for clockless circuits [37] and Broy and Stølen's FOCUS [12]. In both approaches, a module is specified exclusively in terms of the output signal traces that it may exhibit in response to a given input signal trace, without referring to internal state. The trace-based approach introduced in Dolev et al. [27] allows to express timing conditions via (dense) real-time constraints relating input/output signal transitions, and supports fault-tolerance by allowing (sub-)modules to behave erroneously, i.e., deviate from their specified behavior according to some fault model (e.g. Byzantine behavior [86]). It provides concepts for expressing the composition and implementation of circuits by other circuits, which also allow to rigorously specify self-stabilizing [33] circuits. The model has been used to precisely specify the modules of the Byzantine fault-tolerant and self-stabilizing FATAL clock generation algorithm, which will be described in Section 3.1.4, at a level of abstraction that allows for a direct implementation in hardware.

Compared to state-based approaches, it may be more involved to apply a behavioral approach, in particular, in settings like fully synchronous or fully asynchronous systems, where state-space-based descriptions are reasonably simple. After all, in general, it is even difficult to decide whether behavioral specifications match at interface boundaries [21]. On the other hand, it is much better suited for systems with a complex evolution of the system state over time and/or in which the internal state of system components is too complex or even unknown, which is typically the case for self-stabilizing algorithms. Another attractive side-effect is the inherent support of hierarchical design using (pre-existing or yet to be built) building blocks, without the need to define interface state machines. One can easily build a system and/or its *model* in a modular way, by composing sub-components and/or their models, whose implementation can be provided (typically by different designers) and verified at a later stage.

Nevertheless, it is understood that behavioral constraints translate into appropriate constraints on the modules' states implicitly. Although making this relation explicit is not part of our modeling framework, this definitely is part of the effort to implement a module and to prove that it indeed exhibits the specified behaviors. The latter may of course involve any appropriate technique, e.g. timed automata [2] and related verification techniques [1].

Open problems. Although the model of [27] explicitly avoids metastable upsets in fault-free executions, it cannot deal explicitly with metastable upsets and metastability propagation. The work by Ginosar [51], which provides several examples of synchronizer circuits where current prediction models drastically underestimate the probability of metastable upsets, shows the importance for such an extension. The challenge here is to bound metastability resolution times and propagation effects, potentially in a probabilistic manner, to be able to quantify upset probabilities and stabilization times.

Besides the need to extend the model [27] by standard tools like simulation relations and abstractions, the integration with a faithful digital circuit model like [82] remains a challenge. The ultimate goal is a comprehensive modeling framework for modern digital circuits, which facilitates (semi-automated) formal verification of circuits, correctness proofs and accurate performance analysis as well as design parameter synthesis, ideally via supporting tools.

3 Generating and Distributing a System Clock

Simulating a synchronous system in the presence of both transient and permanent failures is a challenging task. The traditional approach to generating and distributing the clock signal, a *clock tree* [43], follows the master/slave principle: the signal of a single quartz oscillator is distributed to all logical gates by means of a tree topology. This approach is trivially self-stabilizing, but it must be abandoned due to the possibility of permanent faults; a failure of the tree's root (i.e., the oscillator) or a node close to the root breaks the entire system.

In the severe failure model considered in this article, this fundamental problem was first studied by S. Dolev and Welch [35, 36]. It was motivated by the observation that the assumption of only a fraction of the node being affected by transient faults is too optimistic for the typically long mission times (e.g., space missions) during which clock synchronization has to be provided.

The ultimate goal is to simulate synchronous rounds that are consistently labeled (at all correct nodes) by a round counter modulo $C \geq 2$, where C usually is fairly large. Dolev and Welch give a protocol that stabilizes in exponential time. While this does not seem very exciting at first glance, at this time the big surprise was that the problem was solvable at all!

For the sake of clarity of presentation, let us break down the task into three subproblems:

1. *Pulse Synchronization:* Simulating unlabeled synchronous rounds in a system with bounded communication delay and local clocks of bounded drift.

2. *Counting:* Computing consistent round counters in a synchronous system with unnumbered rounds.

3. *Clock Distribution:* Distributing pulses and/or round numbers efficiently, i.e., using a low-degree topology.

We remark that it is not imperative to follow this structure when solving the problem. However, the discussion will reveal why this is a fairly natural decomposition of the task.

3.1 Pulse Synchronization

In the pulse synchronization problem, we are given a fully connected system of n nodes, $f < n/3$ of which may be Byzantine faulty. Nodes communicate by messages that are delayed between 0 and 1 time units,[2] which also accounts for any delay incurred by local computations. Each node $i \in \{1, \dots, n\}$ is equipped with a local clock $C_i : \mathbb{R}_0^+ \to \mathbb{R}$ of bounded drift, i.e.,

$$\forall t > t' : \; t - t' \leq C_i(t) - C_i(t') \leq \vartheta(t - t')$$

for a constant $\vartheta > 1$. As we require self-stabilization, the initial states of the nodes, including the values of their local clocks, are arbitrary.

Pulse synchronization now requires nodes to regularly trigger *pulses* in a synchronized way. For a time $T \geq 0$, denote by $t_i^{(k)}$ and $i \in \{1, \dots, n\}$, $k \in \mathbb{N}$, the time when node i generates its k^{th} pulse at or after time T (we omit T from the notation). A pulse synchronization algorithm of precision Δ, accuracy bounds $A_{\min}$, $A_{\max}$, and stabilization time S satisfies in any execution that there is some time $T \leq S$ so that

precision: $\forall i, j, k : \; |t_i^{(k)} - t_j^{(k)}| \leq \Delta$ and

accuracy: $\forall i, k : \; A_{\min} \leq |t_i^{(k+1)} - t_i^{(k)}| \leq A_{\max}$.

Here it is implicit that indices i, j refer to correct nodes only, as we cannot make any guarantees on Byzantine nodes' behavior. Note that typically $A_{\min}$ will be a given parameter and the goal is to minimize $A_{\max} - A_{\min}$ as a function of $A_{\min}$ (or vice versa). Due to the drifting local clocks and delayed messages, indistinguishability arguments show that always $\Delta \geq 1$ and $A_{\max} \geq \max\{\vartheta A_{\min}, 1\}$.

3.1.1 Approaches by the Distributed Community

The results from [36] prompted the question whether pulse synchronization could also be solved efficiently, i.e., with a small stabilization time. In a series of papers, the stabilization time was first reduced to polynomial [20] and then linear [18, 29] in n.[3] These works also revealed that randomization is not essential to solve the problem: the latter algorithm is based on running multiple instances of deterministic consensus concurrently.

Together, these results indicated that the problem could admit solutions suitable for hardware implementation. However, none of the above algorithms was a

[2]This is a normalization. In all existing algorithms, the maximum delay affects stabilization times, etc. linearly.

[3]Linear stabilization time was claimed earlier [19], but the algorithm contained non-trivial errors that were fixed in [18].

good candidate, due to unacceptable stabilization time [36], the need for highly accurate local clocks [20], or message size $\Theta(n \log n)$ and too involved local computations [18, 29]. Malekpour provides an alternative linear-time solution [68, 69] with small messages and simple computations, but uses a simplified model (in particular, it is assumed that $\vartheta = 1$, i.e., there is no clock drift).

3.1.2 Approaches by the Hardware Community

Frequently, designers of fault-tolerant architectures consider the clocking mechanism sufficiently reliable and hence do not add any measures for fault-tolerance. The typical rationale is that a clock distribution network has very strong drivers and is therefore not susceptible to transient disturbances. Permanent defects, on the other hand, will make the system stop operation completely, which may be considered safe in some cases. In reality, however, the clock distribution infrastructure is already so complicated that a "partial" defect can easily occur (imagine a driver responsible for a sub-net failing). Moreover, considering that the clock tree is virtually always the most widespread network, showing the highest activity (in terms of transition frequency), it is not so clear why it should not be affected from transient faults as well. These arguments become even more dominant when talking about requirements of, e.g., a failure probability smaller than 10^{-9} per hour. For such a degree of reliability, it is unrealistic to assume that the system can be just "tried out" before being used, and the cost associated with a design error can be extremely high.

As a single clock source like a crystal oscillator constitutes a single point of failure, practitioners aiming for fault-tolerant clocking often turn to the alternative of using multiple clock sources. While this approach is indeed capable of solving the fault-tolerance issue, it at the same time introduces a new problem, namely that of synchronization. In the single-clock case we have a single timing domain to which all activities are synchronized.[4] Within this synchronous domain it is easy to perform communication based on known time bounds, to establish a clear ordering/precedence of events, and to avoid metastability caused by setup/hold time violations (i.e., too short input pulses) at storage elements. When using multiple clock sources, we immediately leave this safe area. It does not matter whether we use multiple clocks with the same nominal frequency or not – the only important distinction is whether the clocks are correlated (i.e., originate at the same source) or uncorrelated. In the latter case, one can never reason about their relative phase (which is essential for avoiding setup/hold time violations), which makes it mandatory to use explicit synchronizers that can, beyond causing performance

[4]The difficulty of providing this time information with the required phase accuracy all over a large system is, besides the fault-tolerance aspect, a key reason why this globally synchronous design paradigm is being challenged.

and area overheads, never attain complete protection from metastable upsets in the general case [60, 71].

With respect to the precision of existing approaches to synchronization, distinguishing "microticks" and "macroticks" has become common. Ultimately, this boils down to dealing with a large value of Δ by dividing the clock frequency (which is between $1/A_{\max}$ and $1/A_{\min}$ in our model) with respect to communication, so that $\Delta \ll 1/A_{\min}$. In other words, slowing down communication sufficiently, one can make the system work despite large Δ. However, this is obviously detrimental to performance, and one hence must strive for minimizing Δ. The software-based clock synchronization mechanisms found in practical applications like the Time-Triggered Protocol TTP [61] or FlexRay [42, 45] rely on adjusting local microtick counters appropriately to attain synchrony on macrotick level. However, both protocols are, essentially, variants of synchronous approximate agreement [32]. Hence, they require pre-established synchrony for correct operation, implying that they are not self-stabilizing.

Modern application-specific integrated circuits (ASICs) are typically composed of many internally synchronous function blocks that "solve" the issue of synchronization in an even simpler way. Instead of relying on any kind of synchronization between different clock domains, these blocks communicate without making any assumptions on timing (one needs still to avoid buffer overflows, however). This paradigm is called *globally asynchronous locally synchronous* (GALS) in the literature [14]. The intention here is mainly to mitigate the clock distribution problem, but this also provides a foundation for establishing fault-tolerance. Due to the consequent existence of multiple clock domains, such architectures feature copious amounts of synchronizers (to avoid metastability) and arbiters (to establish precedence). This works in practice, but comes with the associated performance penalties. Moreover, due to the current lack of tractable models accounting for metastability, there is no satisfying answer to the question "how many synchronizers suffice?"

What is needed to get rid of the performance and area overheads incurred by the GALS approach is correlated clocking all over the system, even if the phase is not perfectly matched in all places. Such clocks are called *mesosynchronous*. Probably the most natural implementation of such a distributed correlated clock source is a ring oscillator. The underlying principle is to use the delay along a cyclic path (gates plus interconnect) as a time reference. More specifically, such a path is implemented in an inverting fashion (odd number of inverting elements), such that the level is periodically inverted with the loop delay defining the half period of the oscillation. Examples of such approaches are the circuits presented by Maza et al, [77] and Fairbanks et al. [38]. They are ring oscillators in that they both exploit the fact that a circle formed by an odd number of inverters will oscillate, and the frequency of the produced clock is determined by circuit delays. In contrast to the simple basic ring oscillator scheme, multiple “rings” are connected to form meshes of inverters that distribute “clock waves”, thereby also generating new ones. In forming the mesh, output signals of inverters are joined by simply hardwiring them and forked by wire forks.

While these approaches can indeed provide a fast clock that is perceived as “correlated” all over the system, the clock is still not intended and claimed to be fault-tolerant by the authors.

3.1.3 DARTS

One may view the above hardware realizations of distributed clock generators as very simple distributed algorithms, in which the algorithmic steps are determined by the laws of superposition at the merging points. From a theoretical point of view, this results in extremely limited options for the designer of the algorithm. Thus, it is quite tempting to try out more sophisticated algorithms and prove strong(er) fault-tolerance properties. To this end, a suitable algorithm must be chosen and the hardwiring be replaced by an implementation based on logic gates.

This idea was at the heart of the DARTS project [50]. The overall goal of the project was to implement the fault-tolerant synchronization algorithm by Srikanth and Toueg [90] in hardware. The pseudo-code of the algorithm, given in Algorithm 1, is executed at each node of a fully connected network of $n > 3f$ nodes, where f is the number of Byzantine faulty nodes it can tolerate. The code is extremely simple, yet one should not be fooled: its implementation in hardware required to overcome non-trivial obstacles [44].

Algorithm 1 Pseudo-code of a node to synchronously generate round(k) messages.

```
Upon bootup
 1: $k \leftarrow 0$;
 2: broadcast round(0);
Upon reception of a message
 3: if received round($\ell$) from at least $f + 1$ distinct nodes with $\ell > k$ then
 4:     broadcast round($k + 1$), ..., round($\ell$);
 5:     $k \leftarrow \ell$;
 6: end if
 7: if received round($k$) from at least $2f + 1$ distinct nodes then
 8:     broadcast round($k + 1$);
 9:     $k \leftarrow k + 1$;
10: end if
```

According to the algorithm's assumptions, a fully connected communication structure was established, which also provides the highest possible degree of fault-tolerance. The implementation of the merging points, i.e., the actual algorithm, was done in clockless logic. This avoids the issue of having to synchronize the local clock source to the correlated "global" time provided by the algorithm (otherwise one would have to rely on synchronizers again), but in turn requires a careful algorithmic design and timing analysis of the entire system [49]. Interestingly, this means that the only timing sources in DARTS are lower and upper bounds on wire delays, without any formal local clocks. Thus, it is quite close in spirit to the solutions inspired by ring oscillators discussed previously. The hardware implementation of a DARTS node is shown in Figure 2.

The implementation of these hardware nodes, which were called "tick generation algorithm (TG-Alg) nodes," was a very insightful example for how difficult it is to map algorithms that were, at best, developed with a software implementation in mind, to hardware. Assumptions that seem simple at the level of an algorithm may turn out extremely painful when having to be realized in hardware. Examples here are the existence of unbounded counters (such as "k" in Algorithm 1), the request for mutual exclusive execution of tasks, the generous use of operators (multiplication is expensive to implement in hardware), etc.

The identification of relevant timing constraints was a challenging issue in the design of the DARTS prototype ASIC as well. Recall that metastability can, in principle, not be avoided (in the general case) for uncorrelated clock sources. However, one can show that in fault-free executions, metastability does not occur. This is not straight-forward due to the following cyclic dependencies: Under the assumption of proper function of the algorithm one can rely on its guaranteed properties (e.g. precision) when establishing the timing closure to avoid setup/hold time violations. In turn, the freedom from setup/hold time violations is a prerequisite for correct functionality.[5] Note that such timing guarantees are essential, as metastability, a possible result of setup/hold violations, results in unspecified behavior not covered by the analysis of the algorithm.

Several key techniques were applied for overcoming the above challenges:

- the use of difference counters for the cycle number, thus mitigating the problem of unlimited counting;

- the implementation of these counters through Muller pipelines, thus avoiding metastability issues that would arise from concurrent increase and decrease operations of the same counter;

[5]For DARTS, "only" an inductive argument was required. When turning to self-stabilization, establishing that the most basic timing assumptions eventually hold tends to be the most difficult aspect of the reasoning.

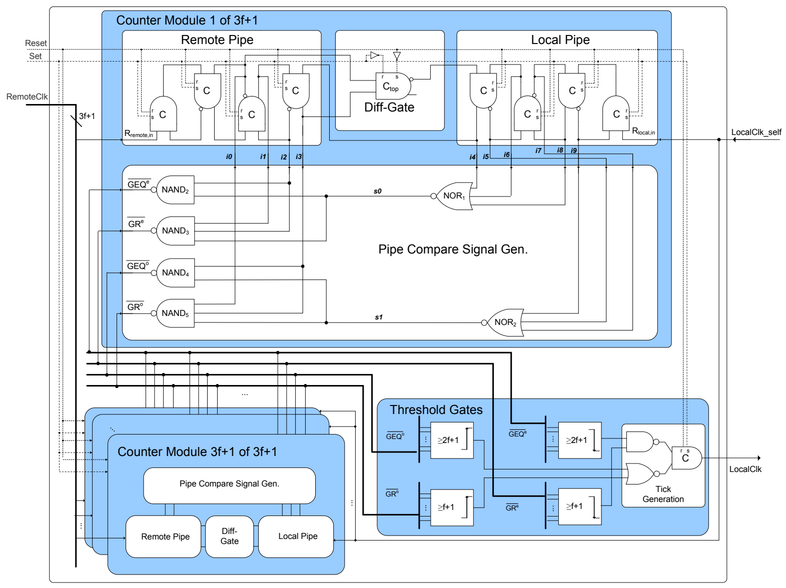


Figure 2: Hardware implementation of a DARTS node in clockless logic.

- a careful mix of event-based and state-based logic;
- the separated treatment of rising and falling signal edges in order to substantially relax the timing constraints.

The project succeeded in developing a working prototype chip, demonstrating that it was indeed feasible to use a distributed algorithm for generating a system-wide clock *and* prove that its *implementation* provides respective guarantees:

- bounded precision and accuracy,
- tolerance of Byzantine faults, and
- use of standard hardware libraries, with one exception: a C-Element must be added.

While the third property might seem like an oddball here, one should be aware that novel circuit components need to be designed on transistor level, layouted, characterized and validated (by simulation) as well. The existing standard libraries had to be augmented by a C-Element for DARTS.

While DARTS constituted a breakthrough in terms of bringing theory and practice closer to each other, the resulting solution exhibits a number of deficiencies calling for further research:

- full connectivity between nodes;
- lack of recovery from transient faults: even if only a minority of nodes undergoes transient faults at any time, there is no mechanism to recover to a correct state;
- too small frequency, limited by the propagation delay of a long loop;
- non-trivial initialization due to fairly strict demands on initial synchrony.

3.1.4 FATAL

In light of the theoretical results from Section 3.1.1 and the proof-of-concept that Byzantine fault-tolerance is viable in low-level hardware provided by DARTS, the obvious next question is whether self-stabilization can be added on the circuit level, too. This was answered affirmatively in [26].

The FATAL algorithm builds on the idea of adding a recovery mechanism to a pulse generation mechanism based on threshold voting. On an abstract level, the FATAL algorithm can be viewed as an implementation of the Srikanth-Toueg algorithm (c.f., Algorithm 1) that avoids having to keep book on tick/pulse numbers by making sure that the time between pulses is sufficiently large: instead of broadcasting round(k) messages, we simply broadcast round messages in Algorithm 1. Another interpretation is that of having a DARTS system that runs slow enough to operate with “pipes of length one”, i.e., simple memory cells.

The basic principle is illustrated in Figure 3, depicting a state machine each node runs a copy of. Circles represent states, arrows state transitions that happen when the condition next to the arrow is satisfied, and the box with “propose” on the state transition from *pulse* to *ready* indicates that the nodes’ memory flags are reset during this state transition. Each node continuously broadcasts whether it is in state propose or not, and when a node perceives another in this state according to this signal (including itself), its respective memory flag is set (i.e., each node has one memory cell for each node, including itself). The condition “3ϑ local time passed” means that node i will transition from *pulse* to *ready* at the time t when its local clock reads $C_i(t) = C_i(t') + 3\vartheta$, where t' is the time when it switched to *pulse*. Nodes generate a pulse when switching to *pulse*.

It is not hard to verify that, if all nodes start in *ready* with their memory flags cleared, this algorithm will solve pulse synchronization with $\Delta = 2$, $A_{\min} = 3 + 3\vartheta$, and $A_{\max} = 3 + 3\vartheta + 3\vartheta^2$. By making nodes wait longer in one of the transitions by

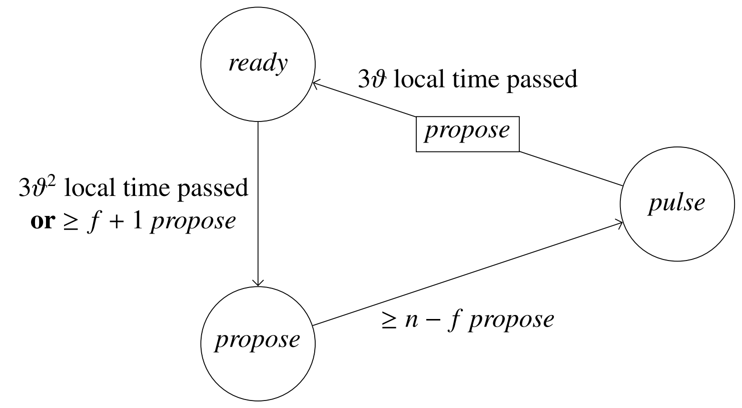

ready
3ϑ local time passed
propose
pulse
$3\vartheta^2$ local time passed
or $\geq f + 1$ *propose*
$\geq n - f$ *propose*
propose

Figure 3: Simple pulse synchronization requiring consistent initialization.

ϑx local time, we can actually have $A_{\min} = 3 + 3\vartheta + x$ and $A_{\max} = 3 + 3\vartheta + 3\vartheta^2 + \vartheta x$, for any $x \geq 0$, i.e., $A_{\max} \to \vartheta A_{\min}$ for $x \to \infty$.

We believe that the recovery mechanism developed for FATAL is a potential key building block of further improvements in the future. In [26], the above basic algorithm is modified so that the task of "stabilizing" the routine, i.e., getting it into a (global) state as if it had been initialized correctly, is reduced to generating a single pulse *by an independent algorithm*. More precisely, all correct nodes need to trigger a "start stabilization" event within a reasonably small time window and then not trigger another such event for $\Theta(1)$ time in order to guarantee stabilization.

The easier task of generating a single "helper pulse" for the purpose of recovery from transient faults is then solved by relying on randomization. The solution used in [25] generates such a pulse with probability $1 - 2^{-\Omega(n)}$ within $O(n)$ time, resulting in an overall stabilization time of $\Theta(n)$. Hence, the algorithm matches the best known stabilization time of $O(n)$. The improvement lies in the communication complexity and the amount of local computations: Apart from a few memory flags for each other node, each node runs a state machine with a constant number of states; each node continuously broadcasts only a few bits about its own state. Moreover, the algorithm can operate with arbitrary values of ϑ, permitting to use very simple oscillators as local clock sources.

In [25], the approach is implemented and evaluated in hardware. The experiments confirm the theoretical results from [26]. However, the algorithm cannot be used for clocking as-is, for several reasons:

- The algorithm generates pulses every $\Theta(1)$ time, but the involved constants are impractically large. Naive application of the approach would result in slowing down systems by several orders of magnitude.

- The pulses are anonymous, i.e., the counting problem discussed in the next section needs to be solved.
- The system is fully connected, which is infeasible in large circuits.

These issues will be discussed next.

3.2 Counting

Once pulse synchronization is solved, it can be used to simulate synchronous rounds: One adjusts the accuracy lower bound $A_{\min}$ such that it allows for the maximal sum of the communication delay between neighbors, the local computations for a round, and a safety margin proportional to Δ (recall that a pulse is not issued at all nodes precisely at the same instant of time). However, due to the strong fault model, it is non-trivial to achieve agreement on a round counter. Round counters are highly useful for, e.g., applying pre-determined time division multiple access (TDMA) schemes to shared resources (memory, communication network, etc.) or scheduling synchronized operations (measurements, snapshots, etc.) that are to be executed regularly.

We will now discuss how a self-stabilizing round counter can be constructed in a synchronous system with $f < n/3$ Byzantine faults. The problem of *C-counting*, where C is an integer greater than 2, is formalized as follows. In each round $r \in \mathbb{N}$, each node i outputs a counter $c_i(r) \in \{0, \ldots, C-1\}$. The algorithm stabilizes in $S \in \mathbb{N}$ rounds, if for all $r \geq S$ we have

agreement: $\forall i, j: \; c_i(r) = c_j(r)$ and

counting: $\forall i: \; c_i(r+1) = c_i(r) + 1 \bmod C$.

In this subsection, the discussion will be more technical, with the goal of illustrating how the fault-tolerance techniques that have been developed by the distributed computing community are canonical tools for designing fault-tolerant algorithms in hardware. We remark that the inclined reader should feel free to skip the technical proofs, as they are not essential to the remaining discussion in this article.

3.2.1 Equivalence to Consensus

The task of (binary) *consensus* requires that, given an input $b_i \in \{0, 1\}$ at each node at the beginning of round 1, each correct node computes an output o_i satisfying

agreement: $\forall i : o_i = o$ for some $o \in \{0, 1\}$ (we refer to o as the output), *validity:* if $\forall i : b_i = b$ then $o = b$, and

termination: all (correct) nodes eventually set their output (exactly once) and terminate.

In practice, one usually requires explicit bounds on when nodes terminate. By an R-round consensus algorithm, we will refer to an algorithm in which all correct nodes terminate by the end of round R.

The counting problem is equally hard as consensus with respect to asymptotic time complexity. We show this for deterministic algorithms and binary consensus algorithms here, but extensions to non-binary consensus and randomized algorithms are straightforward.

Lemma 3.1 (Counting solves consensus)**.** *Given an algorithm for C-counting stabilizing in R rounds, binary consensus (with $f < n/3$ Byzantine nodes) can be solved in R rounds.*

Proof. Denote by $\mathbf{x}(0)$ and $\mathbf{x}(1)$ state vectors such that the counting algorithm would properly count starting from 0 and 1, respectively (regardless of the subset of faulty nodes). Such states must exist, because after stabilization the algorithm will count modulo C and Byzantine nodes may pretend correct operation to avoid detection until after stabilization. Given an input vector $\mathbf{b} \in \{0, 1\}^n$, initialize each (correct) node i with state $x_i(b_i)$ and run the algorithm for R rounds. Then each node outputs $c_i(R) - R \bmod 2$.

Clearly, this algorithm terminates in R rounds and, by the agreement property of the counting protocol, all nodes output the same value. Hence it remains to show that this output value is valid, i.e., equals b if $b_i = b$ for all correct nodes. This follows from the choice of $\mathbf{x}(0)$ and $\mathbf{x}(1)$ and the counting property, which implies that, for all correct nodes i, $c_i(R) = R \bmod C$ if $b = 0$ and $c_i(R) = R + 1 \bmod C$ if $b = 1$. □

The other direction was shown in [30]. We present a simpler variant in the following lemma. It makes use of consensus for non-binary values, i.e., $b_i, o_i \in \{0, \ldots, C - 1\}$ (this case can be reduced to binary consensus in a straightforward manner).

Lemma 3.2 (Consensus solves counting)**.** *Given a synchronous consensus algorithm for inputs $0, \ldots, C-1$ terminating in R rounds that tolerates $f < n/3$ Byzantine nodes, C-counting can be solved with stabilization time $O(R)$.*

Proof. Given the consensus algorithm, we solve C-counting as follows. In each synchronous round, we start a new consensus instance that will generate an output value $c_i(r + R)$ at each node i exactly R rounds later (which will double as node

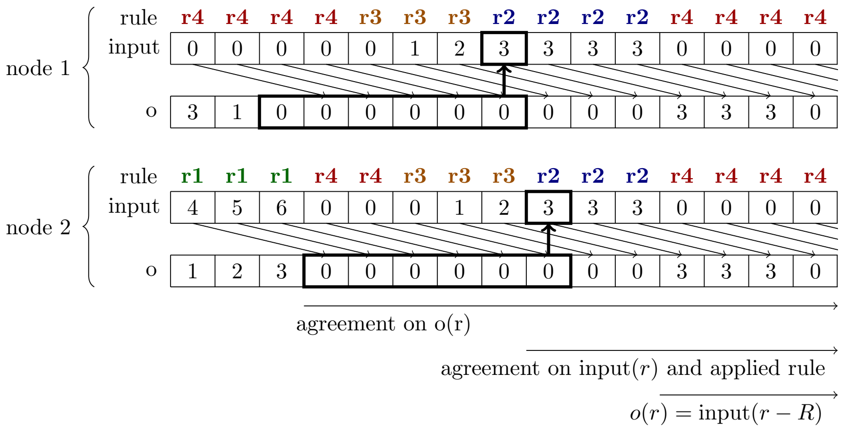


Figure 4: Part of an execution of two nodes running the C-counting algorithm given in the proof of Lemma 3.2, for $C = 8$ and $R = 3$. The execution progresses from left to right, each box representing a round. On top of the input field the applied rule (1 to 4) to compute the input is displayed. Displayed are the initial phases of stabilization: (i) after R rounds agreement on the output is guaranteed by consensus, (ii) then agreement on the input and the applied rule is reached, and (iii) another R rounds later the agreed upon outputs are the agreed upon inputs shifted by 3 rounds.

i's counter value). Note that, while we have no guarantees about the outputs in the first R rounds (initial states are arbitrary), in all rounds $r \geq R$ all correct nodes will output the same value $o(r) = o_i(r)$ (by the agreement property of consensus). Hence, if we define the input value $F_i(r)$ of node i as a function of the most recent $O(R)$ output values at node i, after $O(R)$ rounds all nodes will start using identical inputs $F(r) = F_i(r)$ and, by validity of the consensus algorithm, reproduce these inputs as output R rounds later (cf. Figure 4). In light of these considerations, it is sufficient to determine an input function F from the previous $O(R)$ outputs to values $0, \ldots, C-1$ so that counting starts within $O(R)$ rounds, assuming that the output of the consensus algorithm in round $r + R$ equals the input determined at the end of round r.

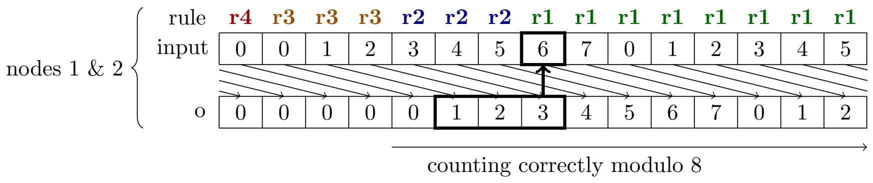


Figure 5: Extension of the execution shown in Figure 4. Nodes have already agreed upon inputs and outputs so that the latter just reproduce the inputs from R rounds ago. The rules now make sure that the nodes start counting modulo 8 in synchrony, always executing rule 1.

We define the following input function, where all values are taken modulo C:

$$\text{input}(r) := \begin{cases} c+R & \text{if} \quad (o(r-R+1),\dots,o(r)) = (c-R+1,\dots,c) \\ x+R & \text{if} \quad \begin{array}{l}(o(r-2R+1-x),\dots,o(r)) = (0,\dots,0,1,\dots,x) \\ \text{ for some } x \in \{0,\dots,R-1\}\end{array} \\ x & \text{if} \quad \begin{array}{l}(o(r-R+1-x),\dots,o(r)) = (0,\dots,0) \\ \text{ for maximal } x \in \{0,\dots,R-1\}\end{array} \\ 0 & \text{else .} \end{cases}$$

In the setting discussed above, it is straightforward to verify the following properties of input:

- Always exactly one of the rules applies, i.e., input is well-defined.
- If the outputs counted modulo C for $2R$ consecutive rounds, they will do so forever (by induction, using the first rule); c.f. Figure 5.
- If this does not happen within $O(R)$ rounds, there will be R consecutive rounds where input 0 will be used (by the third and the last rule), c.f. Figure 5.
- Once R consecutive rounds with input 0 occurred, inputs $1,\dots,2R$ will be used in the following $2R$ rounds (by the second and third rule).
- Finally, the algorithm will commence counting correctly (by the first rule).

Overall, if each node i computes its input $F_i(r)$ from its local view of the previous outputs using input, the algorithm will start counting correctly within $S \in O(R)$ rounds. □

These two lemmas imply that there is no asymptotic difference in the round complexity of consensus and the stabilization time of counting. However, note that the conversion of a consensus algorithm into a counting algorithm given by Lemma 3.2 is very "lossy" in terms of communication complexity and computational efficiency, as R instances of consensus run concurrently! Hence, the main question is whether there are fast solutions to counting that are efficient in terms of communication and computation as well.

3.2.2 Counting Using Shared Coins

The pulse synchronization algorithm by S. Dolev and Welch [36] is conceptually based on a counting algorithm given in the same article, yielding an exponential time randomized solution.

This was improved by Ben-Or et al. [9]. We explain a simpler variant of the algorithm here. The solution is based on *shared coins*. A (weak) shared coin guarantees that there are probabilities $p_0, p_1 > 0$ so that with at least probability p_0, all correct nodes output 0, and with at least probability p_1, all correct nodes output 1. We call $p := \min\{p_0, p_1\}$ the *defiance* of the shared coin. Moreover, we require that the value of the coin is not revealed before the round in which the outputs are generated, so that faulty nodes cannot exploit this knowledge to prevent stabilization.

Observe that we neither require $p_0 + p_1 = 1$ nor that all correct nodes always output the same value. In particular, a trivial shared coin with defiance 2^{-n} is given by each node flipping a coin independently. The algorithm from [36] essentially makes use of this trivial shared coin, which results in its expected exponential stabilization time.

The first step of the algorithm from [9] is to solve 2-counting.

Lemma 3.3 (2-counting from shared coin)**.** *Given a stream of weak shared coins with constant defiance,* 2*-counting can be solved with constant expected stabilization time.*

Proof. In each round r, each node i

1. broadcasts $c_i(r)$;
2. if it received at least $n - f$ times value $c_i(r) - 1 \bmod 2$ in round $r - 1$, it sets $c_i(r + 1) := c_i(r) + 1 \bmod 2$; and
3. otherwise, $c_i(r + 1)$ is set to the output of the shared coin at i in round r.

Before we prove the claim, note that Step 2 depends on messages that were broadcasted in round $r - 1$ instead of messages broadcasted in step 1 during the same round r. The reason for this is to avoid that the faulty nodes exploit so-called *rushing*: As the value of the coin for round r is revealed in round r, faulty nodes may exploit this information to affect the outcome of the broadcast (in terms of what correct nodes observes) exactly so that in Step 3 the "wrong" action is taken by correct nodes relying on the coin. By referring to the broadcast of the previous round instead, the faulty nodes are forced to commit to an outcome of the broadcast before the coin is revealed, making sure that with probability at least p the "right" action is taken by correct nodes in Step 3.

To see that the algorithm indeed stabilizes, observe first that in cannot happen that, in the same broadcast, a correct node sees value 0 at least $n - f$ times and another correct node sees value 1 at least $n - f$ times: this implies that at least $n - 2f$ correct nodes each have $c_i(r) = 0$ and $c_i(r) = 1$, respectively, but we have only $n - f < n - f + (n - 3f) = 2(n - 2f)$ correct nodes (here we used that $f < n/3$). Assume that $c \in \{0, 1\}$ is the unique value such that some node receives c at least $n - f$ times in round $r - 1$. If there is no such value, choose c arbitrarily. With probability at least p_c, all correct nodes set $c_i(r + 1) := c + 2 \bmod 2 = c$ in round r. Similarly, in round $r + 1$ all nodes set $c_i(r + 2)$ to $c + 1 \bmod 2$ with probability at least p_{1-c}. Once this happened, the clocks of correct nodes will start counting deterministically, as always Step 2 will be executed. Hence, the algorithm stabilizes with (independent) probability $p_0 p_1 \geq p^2$ every other round. □

Once 2-counting is available, the generalization to C-counting is achieved by a similar approach. The key difference is that we now use a two-round protocol controlled by the output of the 2-counting algorithm to solve C-counting. We remark that the algorithm essentially performs a gradecast ([40]) followed by achieving a consistent choice with constant probability using the shared coin.

Lemma 3.4 (C-counting from shared coin and 2-counting)**.** *Given a stream of weak shared coins with constant defiance and a 2-counter, C-counting can be solved with constant expected stabilization time.*

Proof. In each round r, each node i performs the following steps.

1. If the 2-counter reads 0:
   (a) broadcast $c_i(r)$;
   (b) if received value $c \neq \perp$ at least $n - f$ times, set helper variable $h_i(r) := c$, otherwise $h_i(r) := \perp$;
   (c) if $b_i(r-1) = 1$ or the shared coin shows 1 at i in round r, set $c_i(r+1) := c_i(r) + 1 \bmod C$, and otherwise $c_i(r + 1) := 0$.
2. If the 2-counter reads 1:
   (a) broadcast $h_i(r - 1)$;

(b) if received value $c \neq \bot$ at least $n - f$ times, set $c_i(r + 1) = c + 1 \bmod C$ and $b_i := 1$;

(c) else if received value $c \neq \bot$ at least $n - 2f$ times, set $c_i(r + 1) = c + 1 \bmod C$ and $b_i(r) := 0$;

(d) else set $c_i(r + 1) := 1$ and $b_i(r) := 0$.

To see why this stabilizes with constant probability within $O(1)$ rounds, observe that the following holds once the 2-counter stabilized:

- If in an even round r all correct nodes agree on the clock value and have $b_i(r - 1) = 1$, the algorithm will count correctly forever.
- The same holds if they agree on the clock and the shared coin shows 1 at all nodes in round r.
- As $f < n/3$, there can be at most one value $c \neq \bot$ with correct nodes setting $h_i(r) := c$ in an even round r.
- If any correct node i receives this unique value c at least $n - f$ times in the subsequent odd round $r + 1$, all correct nodes receive c at least $n - 2f$ times. Hence, *either* it holds that (b) or (c) applies at all correct nodes *or* (c) or (d) apply at all nodes.
- In the first case, all correct nodes have the same clock value. Hence, the shared coin showing 1 in round $r + 2$ guarantees stabilization.
- In the second case, all correct nodes set $b_i(r + 1) := 0$. Hence, if the coin shows 0 at all nodes in round $r+2$, they all set $c_i(r+3) := 0$ and, subsequently $c_i(r + 4) := 1$. If the coin shows 1 at all nodes in round $r + 4$, this implies stabilization.

Hence, the algorithm stabilizes with (independent) probability $\min\{p_1, p_0 p_1\} \geq p^2$ within every 4 rounds once the 2-counter counts correctly. □

Composing the two algorithms yields a C-counter with expected constant stabilization time. We stress the similarity of the routine to solutions to consensus based on shared coins [88]; the ideas and concepts developed for consensus translate directly to the counting problem, even if it might be harder in terms of the required communication.

Unfortunately, this approach to solving counting suffers from the same problem as consensus algorithms based on shared coins: theoretically sound protocols that generate shared coins based on the private randomness of the nodes are highly expensive in terms of communication and computation.

3.2.3 Constructing Large Counters from Small Counters

There are several techniques for constructing large counters from small counters, indicating that the key challenge is to obtain a 2-counter. One is given by Lemma 3.4, which however necessitates the presence of a shared coin. Another one is very basic, but inefficient time-wise, as C enters the stabilization time as a factor.

Lemma 3.5 (C-counting from 2-counting [31])**.** *Given a* 2*-counting algorithm with stabilization time S, for any $k \in \mathbb{N}$ we can solve 2^k-counting with stabilization time $2^k S$ and at most factor* 2 *more communication.*

Proof. The proof goes by induction over k, the base case being covered by assumption. For the step, we simply execute the 2-counting algorithm slower, by performing one round when the 2^k-counter switches to 0. This way, concatenating the clock bit of the slow 2-counter and the 2^k-counter, we obtain a 2^{k+1}-counter. The stabilization time is $2^k S$ for the slowed-down 2-counter plus the stabilization time of the 2^k-counter, yielding by induction a total stabilization time of $\sum_{l=1}^{k} 2^l S < 2^{k+1} S$. The communication bounds of the 2-counting algorithm together with the slow-down yield the claim concerning the amount of communication.[6] □

A simple variant on the theme achieves faster stabilization at the cost of increased message size.

Lemma 3.6 (C-counting from 2-counting, faster stabilization)**.** *Assuming we are given a* 2*-counting algorithm with stabilization time S, for any $k \in \mathbb{N}$ we can solve 2^k-counting with stabilization time $2^k + kS$ and at most factor k more communication.*

Proof. The proof goes by induction over k, the base case being covered by assumption. For the step, we execute another copy of the 2-counting algorithm with a minor change: If the already constructed 2^k-counter reads 0, we skip a round of the 2-counting algorithm. Thus, the 2-counter will proceed by $2^k - 1 \bmod 2 = 1$ every 2^k rounds. The 2^{k+1}-counter is now given by the 2^k-counter and an additional leading bit, which is the value the 2-counter had when the 2^k-counter most recently was 0. By the above considerations, the 2^{k+1}-counter will count correctly once (i) both counters stabilized and (ii) the 2^k-counter had value 0 once after this happened.

[6]Note that one can also ensure that the maximum message size does not increase by more than factor 2, by shifting the communication pattern so that no more than 2 instances of the 2-counting algorithm communicate in the same round.

The stabilization time bound now follows: once the 2^k-counter is correctly operating, the 2-counter stabilizes within $S+1$ rounds, and the 2^k-counter will become 0 again within another 2^k rounds; summation yields $\sum_{l=0}^{k} 2^l + S < 2^{k+1} + (k+1)S$ rounds for stabilization of the constructed 2^{k+1}-counter. The communication bound is trivially satisfied. □

Even if 2-counting can be solved efficiently, these techniques are slow if C is large. Motivated by this issue, in [46] large clocks are constructed from small ones by encoding clock values over multiple rounds. This enables to increase the clock range exponentially. Specifically, the paper provides two main results. The first is essentially a reduction to consensus (with only one instance running at any given time), and it is similar to the approach taken in Lemma 3.4. The key changes are the following:

1. To enable 1-bit messages, broadcasts of clock values are replaced by $\lceil \log C \rceil$ rounds each in which the clock bits are transmitted sequentially.
2. Instead of relying on shared coins, nodes run an instance of consensus with the variables b_i determined in odd "rounds" as input. The output of the consensus algorithm is used by all nodes to decide whether c (shifted by the number of rounds passed) is the next clock value or 0.
3. In all other rounds, clock values are locally increased by 1 modulo C.

Due to the use of consensus, the correctness argument becomes simpler. If the consensus algorithm outputs 1, there must be a node that used input 1 and therefore received $n-f$ times c in the second broadcast. This implies that all nodes received $n-2f \geq f+1$ times c and therefore can determine the new clock value. Otherwise, the algorithm is certain to default to resetting clocks to 0.

This approach replaces the need for a shared coin with the need for an efficient consensus algorithm and a sufficiently large counter. We instantiate the result for the phase king protocol [10] in the following corollary.

Corollary 3.7 (Large counters from small counters and consensus [46])**.** *Given a C-counter for $C \in O(n)$ sufficiently large, a $2^{\Omega(C)}$-counter with stabilization time $O(n)$ can be constructed deterministically using 1-bit broadcast messages.*

Note that one can combine this corollary with Lemma 3.5 or Lemma 3.6 to construct large counters from 2-counters. In [46], a randomized alternative to these lemmas is given that constructs larger counters from an $O(1)$-counter at smaller overhead. Using either of the two lemmas to obtain the required $O(1)$-counter, the following corollary can be derived.

Corollary 3.8 (Large counters from 2-counters using randomization [46])**.** *Given a* 2*-counter, C-counting can be solved with expected stabilization time $O(n+\log C)$ and $O(\log^* C)$ broadcasted bits per node and round.*

3.2.4 Counting from Pulse Synchronization

Ironically, the obstacle of solving 2-counting disappears if it is feasible to remove one level of abstraction and exert some control over how (perfect) synchrony is simulated. More concretely, in all existing pulse synchronization algorithms one can increase $A_{\min}$ (the minimum time between consecutive pulses) at will, so that $A_{\max}$ grows proportionally. In particular, this can be used to allow for more than a single synchronous round to be simulated in between pulses. Initializing the simple (non-self-stabilizing) pulse synchronization algorithm given in Figure 3 consistently, we thus can allow for sufficient time to generate a tunable number C of "sub-pulses" before the next pulse occurs. Counting locally modulo C by re-initializing the counter to 0 at each pulse and increasing it by 1 on each sub-pulse, we can use the sub-pulses as round delimiters for simulating synchronous rounds with a self-stabilizing C-counter that comes "for free".

To be precise, this counter does not come entirely for free; apart from the additional logic, increasing $A_{\min}$ may also result in increasing the stabilization time of the pulse synchronization algorithm. However, one can obtain a 2-counter or, in fact, any $O(1)$-counter, without asymptotically affecting the stabilization time of the underlying pulse synchronization algorithm. The techniques for constructing larger counters based on small counters given in [46] then can take it from there.

From an abstract perspective, this can be seen as an implementation of the Srikanth-Toueg algorithm [90] that counts only up to $O(1)$ and then is restarted. This approach is followed by FATAL$^+$, an extended version of FATAL analyzed in [26] and implemented and evaluated in [25]. Owing to the simplicity of the algorithm given in Figure 3, the sub-pulses can actually be produced at a higher frequency and with better precision than offered by the basic FATAL algorithm.

We remark that the method of coupling the two algorithms in FATAL$^+$ may be of independent interest. We expect that it can also be applied to couple FATAL or FATAL$^+$ to non-stabilizing pulse synchronization protocols based on approximate agreement, like the earlier discussed TTP and FlexRay protocols. This bears the promise of obtaining a pulse synchronization protocol that (i) can run at even higher frequencies (i.e., $A_{\min}$ is smaller) and (ii) achieves a precision in the order of the *uncertainty* of the communication delay, i.e., if messages are underway between $1-\varepsilon$ and 1 time unit, then $\Delta \in O(\varepsilon)$. This is to be contrasted to algorithms like DARTS or FATAL, which use explicit voting for resynchronization at each pulse and therefore have $\Delta \geq 1$ even if there is a lower bound on the communication delay. Note that the uncertainty of the communication delay is known to be a lower bound on the worst-case precision of any clock synchronization protocol [67], implying that $\Delta \in \Omega(\varepsilon)$ is unavoidable.

3.2.5 Constructing Counters from Scratch

Modifying the system on such a deep level as how the clock signal is provided may not always be possible, e.g., when one designs a submodule in a larger circuit. In this case, one may still have to solve 2-counting directly.

Recent research has started to tackle this issue. In [31], efficient solutions for the special case of $f = 1$ are designed and proved optimal in terms of the trade-off between stabilization time and number of local states using computer-aided design. However, as the space of algorithms to consider is huge, this method does not scale; even the case $f = 2$ is currently beyond reach.

In [66], a recursive approach is taken to avoid that each node participates in $\Theta(R)$ concurrent instances of an R-round consensus algorithm used for establishing agreement on clock values. The target is to, in each step of the recursion, boost the *resilience* of the protocol to faults, while increasing neither the number of required nodes nor the time for stabilization too much. The result is an algorithm of slightly suboptimal resilience $f < n^{1-o(1)}$, but linear stabilization time $O(f)$ and only $O(\log^2 n/\log\log n + \log C)$ state bits per node. These state bits are broadcasted to all other nodes in each round. For deterministic algorithms, this implies an exponential improvement in communication complexity as compared to the solution from Lemma 3.2, since deterministic consensus algorithms satisfy that $R > f$ (see [41]).

To sketch the idea of the approach, consider a system of n nodes. Each node i runs a 0-resilient C_i-counter (for some C_i we will determine shortly). This is nothing but a local counter modulo C_i: it is increased in each round, and it works correctly so long as i does not fail. We use these counters to let nodes determine temporary leaders that will assist with stabilization if required; once the system is stabilized, the corresponding communication will be ignored. The current leader's local counter is used to provide a temporarily working counter to all nodes. This counter is used to run an $O(f)$-round consensus algorithm, the phase king protocol [10], to agree on the counter values. It is straightforward to show that agreement cannot be destroyed by this mechanism once it is achieved, even if the temporary counter produces garbage later on.

In short, this mechanism reduces the task to ensuring that eventually a correct leader will emerge and persist for $R \in O(f)$ rounds. We achieve this as follows: Node 1 will cycle through all possible leaders, where it keeps "pointing" to the same node for $\Theta(R)$ consecutive rounds. Node 2 does the same, albeit slower by a factor of $2n$. This guarantees that, for any other node j, nodes 1 and 2 eventually consider it the leader for R consecutive rounds. We proceed inductively, slowing down the "leader pointer" of node j by a factor of $(2n)^{j-1}$ compared to the one of node 1. Clearly, eventually all correct nodes will point to a correct node for R consecutive rounds.

The downside of this approach is that the stabilization time is exponential, since the slowest pointer takes $R \cdot (2n)^n$ rounds to complete a single cycle. Here the recursion comes into play. Instead of using single nodes, on each level of the recursion one groups the nodes into a small number $k \in O(1)$ of clusters. Each cluster runs an f-resilient counter that is used to determine to which leader the node currently points. The "leaders" now are also clusters, meaning that the slowest clock takes $R \cdot (2k)^k \in O(R)$ rounds for a cycle. Now the same principle can be applied, assuming that we can also convince correct nodes in blocks with more than f faults to point to the "correct" leader the blocks with at most f faults will eventually choose. Requiring that fewer than half of the k blocks have more than f faults, this is ensured by an additional majority vote. The resilience of the compound algorithm therefore becomes $\lceil k/2 \rceil (f + 1) - 1$ (one fault less than required to make half of the blocks contain $f + 1$ faults). Crunching numbers and tuning parameters, one obtains the claimed result.

Maybe the most interesting aspect of this construction is its resemblance to recursive approaches for improving the communication complexity of consensus protocols [10, 15, 57]. The additional challenge here is the lack of a common clock, which is overcome by relying on the guarantees from the lower levels together with the simple leader election mechanism explained above. From this point of view, the construction can be interpreted as a natural generalization of the recursive phase king algorithm given in [10]. Accordingly, one may hope that also for counting, it is possible to achieve optimal resilience in a similar way.

3.3 Clock Distribution

All the algorithms so far assume full connectivity, which is unrealistic if the number of nodes is not quite small. In principle, one could seek for solutions to the pulse synchronization and counting problems in low-degree topologies directly. However, it is much easier to solve these tasks in a small, fully connected "core" and then distribute the results redundantly using a sparse interconnection topology. The advantage is that for the distribution problem, it now is sufficient to have pre-defined master/slave relations, i.e., a pre-defined direction of propagation of the clock signal throughout the system. This greatly simplifies the job, as it circumvents the need for reaching any sort of agreement: the clock information is plainly dictated by the nodes further upstream.

When using a sparse network, we must also decrease our expectations in terms of fault-tolerance. Solving clock synchronization (or consensus, for that matter) in the presence of f faults requires minimum node degrees of $2f + 1$, as otherwise

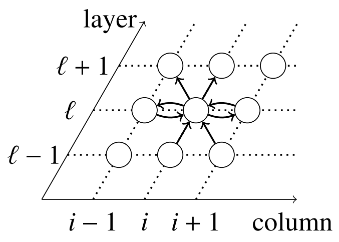


Figure 6: Node i in layer ℓ of a HEX grid and its incident links. The node propagates a pulse when having received it from both nodes on the previous layer. If one fails, the second signal is provided by one of its neighbors in the same layer.

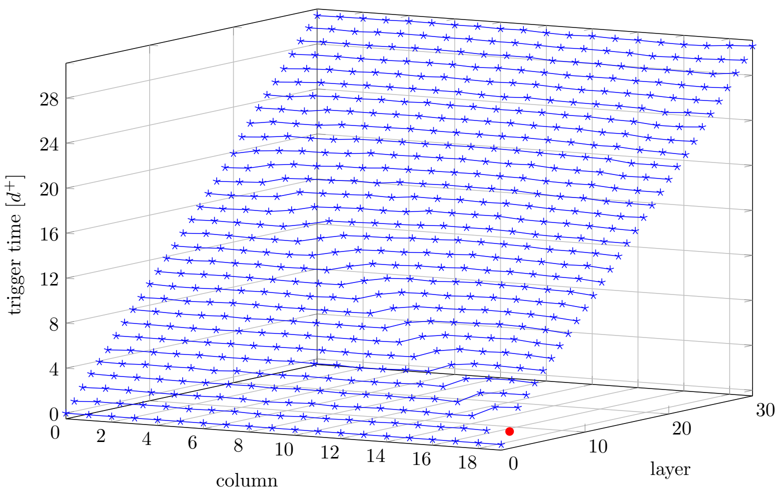


Figure 7: Pulse propagation in HEX with a single Byzantine node. The figure shows pulse trigger times of nodes in a grid: initially nodes (0 to 19) in layer 0 generate pulses in synchrony, feeding these pulses into the grid. The Byzantine faulty node 19 in layer 1 generates a "ripple" in trigger times that is clearly visible in the next layer, but smoothes out over the following layers.

it may happen that a correct node does not have a majority of correct neighbors, rendering it impossible to falsify a claim jointly made by its faulty neighbors [23]. Respecting this, we require only that, for a given parameter f, the system tolerates up to f faulty nodes in the neighborhood of correct nodes.

Following these two paradigms – local fault-tolerance and directed clock propagation – and requiring a "nice" interconnect topology (planarity, connections to physically close nodes) led to HEX [24]. In a HEX grid, each node has 6 neighbors arranged in a hexagon, and the clock information spreads along the layers of the grid, cf. Figure 6. Nodes simply wait for the second signal indicating the current clock pulse, where nodes in the same layer send redundant signals in case one of the nodes in the preceding layer fails. Accordingly, a HEX grid tolerates $f = 1$ local fault, while having small node degrees and a planar topology.[7]

In order to prove precision bounds for HEX, we assumed that communication delays vary by at most $\varepsilon \ll 1$. Clearly, this implies that the precision cannot deteriorate by more than ε per layer. Surprisingly, a much stronger bound of $1 + \Delta_0 + O(\varepsilon^2 W)$ can be shown on the precision of adjacent nodes, where Δ_0 reflects the precision of the core and W is the width of the HEX grid.

This is an example of a non-trivial *gradient property*, a concept introduced by Fan and Lynch [39]. Finding clock distribution mechanisms that are suitable for hardware realization, fault-tolerant, and provide non-trivial gradient properties is of great interest, as a strong gradient property enables adjacent chip components to communicate at smaller delays in a synchronous, clock-driven fashion. In particular, it directly affects the time required to simulate a synchronous round and hence the operational frequency of the entire system.

Unfortunately, the worst-case precision of HEX deteriorates by $\Theta(f)$ in the presence of f faults, cf. Figure 7. While the simulations show that this is most likely overly pessimistic [65], the adverse effects of even a single fault are problematic in comparison to the surprisingly good performance of the system in absence of faults. We hope that topologies that feature at least $2f + 1$ links to the preceding layer will offer much better precision in face of faults; the idea is illustrated in Figure 8.

Open problems. In Section 3.2.2 we discussed efficient approaches to construct 2-counters from shared coins. While generating shared coins assuming Byzantine failures is prohibitively costly in terms of communication, it is interesting whether there are efficient shared coin protocols under weaker failure assumptions that are realistic for the considered hardware setting.

Clearly, the search for clock distribution topologies that can be implemented

[7]Clearly, the principle can be generalized to larger values of f adding edges, but degrees increase and planarity is lost.

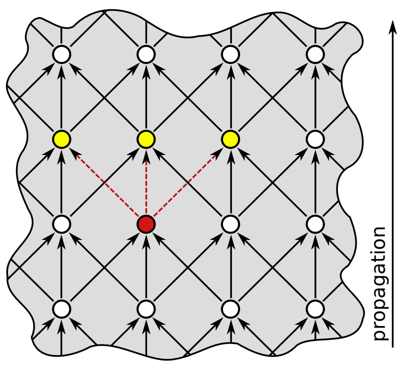


Figure 8: Local structure of a clock propagation approach similar too HEX. Using three connections from the previous layer helps to further mitigate the effect of faults on precision, as the redundant third clock signal does not propagate along a longer path.

with a small number of layers, balanced link delays and sufficiently high degree to tolerate more than 1 local node failure is of central interest. It is also not clear of how to adapt the pulse triggering rules in these HEX-variants to obtain optimal guaranteed precision bounds.

Improving the precision of fault-tolerant self-stabilizing approaches to clocking is an important challenge to achieve utility in practice. As mentioned earlier, it is promising to couple existing solutions with weak precision bounds to algorithms based on approximate agreement to combine high precision and self-stabilization.

Last but not least, an important question is how to verify the correctness of designs prior to production. Striving for algorithms that are sufficiently simple to be implemented in practice bears the promise of enabling formal verification of (parts of) the resulting systems. Given that suitable models can be devised, a grand challenge is the full verification of a fault-tolerant clocking mechanism bottom to top, from gates and wires up to the synchronous abstraction.

4 Conclusion

Due to the continuously increasing scale and complexity of today's VLSI circuits, it becomes insufficient to ensure their reliability by fault mitigation techniques at technological and gate level only, as manufactures will not be able to support the combination of exponentially growing numbers of transistors and decreasing feature size indefinitely. Error correction, on the other hand, is restricted to storing information, neglecting the issue of dependable computation. Hence, one must strive for algorithmic fault-tolerance above the gate level, but below the abstraction of a synchronous, computationally powerful machine.

The distributed computing community has developed many concepts and algorithmic ideas that can be applied to VLSI circuits once we see them as what they truly are: distributed systems in their own right. The main challenges are

- to adapt and extend the existing theory beyond the abstraction of computationally powerful nodes;
- to devise models of computation that account for faults and metastability in a tractable manner;
- to come up with simple, yet efficient algorithms suitable for hardware implementation; and
- to reason about their correctness in ways ensuring that produced chips *will* work.

We believe that the examples given in this article demonstrate that the existing techniques are essential tools in tackling these challenges. The task that lies ahead is to fill the gap between fault-tolerance in theory and the design of practical, dependable hardware.

References

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The Algorithmic Analysis of Hybrid Systems. *Theoretical computer science*, 138(1):3–34, Feb. 1995.

[2] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[3] R. Alur, T. Henzinger, G. Lafferriere, and G. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[4] J. H. Anderson and M. G. Gouda. A new explanation of the glitch phenomenon. *Acta Informatica*, 28(4):297–309, 1991.

[5] P. J. Ashenden. *The designer's guide to VHDL*, volume 3. Morgan Kaufmann, 2010.

[6] R. Baumann. Radiation-Induced Soft Errors in Advanced Semiconductor Technologies. *IEEE Transactions on Device and Materials Reliability*, 5(3):305–316, Sept. 2005.

[7] S. Beer, R. Ginosar, J. Cox, T. Chaney, and D. M. Zar. Metastability challenges for 65nm and beyond; simulation and measurements. In *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1297–1302. IEEE, 2013.

[8] M. Bellido, J. Chico, and M. Valencia. *Logic-timing Simulation And the Degradation Delay Model*. Imperial College Press, 2006.

[9] M. Ben-Or, D. Dolev, and E. N. Hoch. Fast Self-Stabilizing Byzantine Tolerant Digital Clock Synchronization. In *27th Symposium on Principles of Distributed Computing (PODC)*, pages 385–394, 2008.

[10] P. Berman, J. A. Garay, and K. J. Perry. *Bit Optimal Distributed Consensus*, pages 313–321. Plenum Press, New York, NY, USA, 1992.

[11] G. Brown, M. Gouda, and C. lin Wu. Token systems that self-stabilize. *IEEE Transactions on Computers*, 38(6):845–852, Jun 1989.

[12] M. Broy and K. Stølen. *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer-Verlag New York, Inc., 2001.

[13] T. J. Chaney and C. E. Molnar. Anomalous behavior of synchronizer and arbiter circuits. *IEEE Transactions on Computers*, 22(4):421–422, 1973.

[14] D. M. Chapiro. *Globally-Asynchronous Locally-Synchronous Systems*. PhD thesis, Stanford University, 1984.

[15] B. A. Coan and J. L. Welch. Modular Construction of a Byzantine Agreement Protocol with Optimal Message Bit Complexity. *Information and Computation*, 97(1):61–85, 1992.

[16] C. Constantinescu. Trends and Challenges in VLSI Circuit Reliability. *IEEE Micro*, 23(4):14–19, 2003.

[17] J. Cortadella and M. Kishinevsky. Synchronous Elastic Circuits with Early Evaluation and Token Counterflow. In *44th Annual Design Automation Conference (DAC)*, pages 416–419, New York, NY, USA, 2007. ACM.

[18] A. Daliot and D. Dolev. Self-Stabilizing Byzantine Pulse Synchronization. *Computing Research Repository*, abs/cs/0608092, 2006.

[19] A. Daliot, D. Dolev, and H. Parnas. Linear Time Byzantine Self-Stabilizing Clock Synchronization. In *7th International Conference on Principles of Distributed Systems (OPODIS)*, volume 3144 of *LNCS*, pages 7–19. Springer Verlag, Dec 2003. A revised version appears in Cornell ArXiv: *http://arxiv.org/abs/cs.DC/0608096*.

[20] A. Daliot, D. Dolev, and H. Parnas. Self-Stabilizing Pulse Synchronization Inspired by Biological Pacemaker Networks. In *6th Symposium on Self-Stabilizing Systems (SSS)*, pages 32–48, 2003.

[21] L. de Alfaro, T. A. Henzinger, and M. Stoelinga. Timed Interfaces. In *Embedded Software (EMSOFT)*, pages 108–122, 2002.

[22] A. Dixit and A. Wood. The Impact of New Technology on Soft Error Rates. In *IEEE Reliability Physics Symposium (IRPS)*, pages 5B.4.1–5B.4.7, Apr 2011.

[23] D. Dolev. The Byzantine Generals Strike Again. *Journal of Algorithms*, 3:14–30, 1982.

[24] D. Dolev, M. Függer, C. Lenzen, M. Perner, and U. Schmid. HEX: Scaling Honeycombs is Easier than Scaling Clock Trees. In *25th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2013.

[25] D. Dolev, M. Függer, C. Lenzen, M. Posch, U. Schmid, and A. Steininger. Rigorously Modeling Self-Stabilizing Fault-Tolerant Circuits: An Ultra-Robust Clocking Scheme for Systems-on-Chip. *Journal of Computer and System Sciences*, 80(4):860–900, 2014.

[26] D. Dolev, M. Függer, C. Lenzen, and U. Schmid. **F**ault-tolerant **A**lgorithms for **T**ick-generation in **A**synchronous **L**ogic: Robust Pulse Generation. *Journal of the ACM*, 61(5):30:1–30:74, 2014.

[27] D. Dolev, M. Függer, M. Posch, U. Schmid, A. Steininger, and C. Lenzen. Rigorously modeling self-stabilizing fault-tolerant circuits: An ultra-robust clocking scheme for systems-on-chip. *Journal of Computer and System Sciences*, 80(4):860–900, 2014.

[28] D. Dolev, M. Függer, U. Schmid, and C. Lenzen. Fault-tolerant Algorithms for Tick-generation in Asynchronous Logic: Robust Pulse Generation. *Journal of the ACM*, 61(5):30:1–30:74, Sept. 2014.

[29] D. Dolev and E. Hoch. Byzantine Self-Stabilizing Pulse in a Bounded-Delay Model. In *9th Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, volume 4280, pages 350–362, 2007.

[30] D. Dolev and E. Hoch. On Self-stabilizing Synchronous Actions Despite Byzantine Attacks. In *21st Symposium on Distributed Computing (DISC)*, pages 193–207, 2007.

[31] D. Dolev, J. H. Korhonen, C. Lenzen, J. Rybicki, and J. Suomela. Synchronous Counting and Computational Algorithm Design. In *15th Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 237–250, 2013.

[32] D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching Approximate Agreement in the Presence of Faults. *Journal of the ACM*, 33:499–516, 1986.

[33] S. Dolev. *Self-Stabilization*. MIT Press, 2000.

[34] S. Dolev and Y. Haviv. Self-stabilizing microprocessor: analyzing and overcoming soft errors. *IEEE Transactions on Computers*, 55(4):385–399, April 2006.

[35] S. Dolev and J. L. Welch. Self-Stabilizing Clock Synchronization in the Presence of Byzantine Faults (Abstract). In *14th Symposium on Principles of Distributed Computing (PODC)*, page 256, 1995.

[36] S. Dolev and J. L. Welch. Self-Stabilizing Clock Synchronization in the Presence of Byzantine Faults. *Journal of the ACM*, 51(5):780–799, 2004.

[37] J. C. Ebergen. A formal approach to designing delay-insensitive circuits. *Distributed Computing*, 5(3):107–119, 1991.

[38] S. Fairbanks and S. Moore. Self-Timed Circuitry for Global Clocking. In *11th International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 86–96, 2005.

[39] R. Fan and N. Lynch. Gradient Clock Synchronization. In *23rd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 320–327, 2004.

[40] P. Feldman and S. Micali. Optimal algorithms for Byzantine agreement. In *ACM Symposium on Theory of Computing*, pages 148–161, 1988.

[41] M. J. Fischer and N. A. Lynch. A Lower Bound for the Time to Assure Interactive Consistency. *Information Processing Letters*, 14:183–186, 1982.

[42] FlexRay Consortium et al. FlexRay communications system-protocol specification. *Version 2.1*, 2005.

[43] E. G. Friedman. Clock Distribution Networks in Synchronous Digital Integrated Circuits. *Proceedings of the IEEE*, 89(5):665–692, 2001.

[44] G. Fuchs and A. Steininger. VLSI Implementation of a Distributed Algorithm for Fault-Tolerant Clock Generation. *Journal of Electrical and Computer Engineering*, 2011(936712), 2011.

[45] M. Függer, E. Armengaud, and A. Steininger. Safely Stimulating the Clock Synchronization Algorithm in Time-Triggered Systems – A Combined Formal and Experimental Approach. *IEEE Transactions on Industrial Informatics*, 5(2):132–146, 2009.

[46] M. Függer, M. Hofstätter, C. Lenzen, and U. Schmid. Efficient Construction of Global Time in SoCs despite Arbitrary Faults. In *16th Conference on Digital System Design (DSD)*, pages 142–151, 2013.

[47] M. Függer, R. Najvirt, T. Nowak, and U. Schmid. Towards Binary Circuit Models That Faithfully Capture Physical Solvability. In *Design, Automation, and Test in Europe (DATE)*, 2015.

[48] M. Függer, T. Nowak, and U. Schmid. Unfaithful Glitch Propagation in Existing Binary Circuit Models. In *19th International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 191–199, 2013.

[49] M. Függer and U. Schmid. Reconciling Fault-Tolerant Distributed Computing and Systems-on-Chip. *Distributed Computing*, 24(6):323–355, 2012.

[50] M. Függer, U. Schmid, G. Fuchs, and G. Kempf. Fault-Tolerant Distributed Clock Generation in VLSI Systems-on-Chip. In *6th European Dependable Computing Conference (EDCC)*, pages 87–96, 2006.

[51] R. Ginosar. Fourteen Ways to Fool Your Synchronizer. In *9th International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 89–96, 2003.

[52] International Technology Roadmap for Semiconductors, 2012. http://www.itrs.net.

[53] W. Jang and A. J. Martin. SEU-Tolerant QDI Circuits. In *11th International Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 156–165, 2005.

[54] W. Jang and A. J. Martin. A soft-error-tolerant asynchronous microcontroller. In *13th NASA Symposium on VLSI Design*, 2007.

[55] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager. *The Theory of Timed I/O Automata*. Morgan & Claypool Publishers, 2006.

[56] S. Keller, M. Katelman, and A. J. Martin. A Necessary and Sufficient Timing Assumption for Speed-Independent Circuits. In *15th Symposium on Asynchronous Circuits and Systems (ASYNC)*, pages 65–76, 2009.

[57] V. King and J. Saia. Breaking the $O(N^2)$ Bit Barrier: Scalable Byzantine Agreement with an Adaptive Adversary. *Journal of the ACM*, 58(4):18:1–18:24, 2011.

[58] D. J. Kinniment. *Synchronization and Arbitration in Digital Systems*. Wiley, Chichester, 2008.

[59] D. J. Kinniment, A. Bystrov, and A. V. Yakovlev. Synchronization Circuit Performance. *IEEE Journal of Solid-State Circuits*, SC-37(2):202–209, 2002.

[60] L. Kleeman and A. Cantoni. On the Unavoidability of Metastable Behavior in Digital Systems. *IEEE Transactions on Computers*, C-36(1):109–112, 1987.

[61] H. Kopetz and G. Bauer. The time-triggered architecture. *Proceedings of the IEEE*, 91(1):112–126, 2003.

[62] I. Koren and Z. Koren. Defect tolerance in VLSI circuits: Techniques and yield analysis. *Proceedings of the IEEE*, 86(9):1819–1838, Sep 1998.

[63] L. Lamport. The Temporal Logic of Actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872–923, 1994.

[64] E. A. Lee and P. Varaiya. *Structure and Interpretation of Signals and Systems*. LeeVaraiya.org, 2nd edition, 2011.

[65] C. Lenzen, M. Perner, M. Sigl, and U. Schmid. Byzantine Self-Stabilizing Clock Distribution with HEX: Implementation, Simulation, Clock Multiplication. In *6th Conference on Dependability (DEPEND)*, 2013.

[66] C. Lenzen, J. Rybicki, and J. Suomela. Towards Optimal Synchronous Counting. In *34th Symposium on Principles of Distributed Computing (PODC)*, 2015.

[67] J. Lundelius and N. Lynch. An Upper and Lower Bound for Clock Synchronization. *Information and Control*, 62(2-3):190–204, 1984.

[68] M. Malekpour. A Byzantine-Fault Tolerant Self-stabilizing Protocol for Distributed Clock Synchronization Systems. In *9th Conference on Stabilization, Safety, and Security of Distributed Systems (SSS)*, pages 411–427, 2006.

[69] M. Malekpour. A Self-Stabilizing Byzantine-Fault-Tolerant Clock Synchronization Protocol. Technical report, NASA, 2009. TM-2009-215758.

[70] R. Manohar and A. J. Martin. Quasi-delay-insensitive circuits are turing-complete. Technical report, Pasadena, CA, USA, 1995.

[71] L. Marino. General Theory of Metastable Operation. *IEEE Transactions on Computers*, C-30(2):107–115, 1981.

[72] A. J. Martin. Compiling communicating processes into delay-insensitive VLSI circuits. *Distributed Computing*, 1(4):226–234, 1986.

[73] A. J. Martin. The Limitations to Delay-insensitivity in Asynchronous Circuits. In *Sixth MIT Conference on Advanced Research in VLSI*, AUSCRYPT '90, pages 263–278, Cambridge, MA, USA, 1990. MIT Press.

[74] A. J. Martin. Synthesis of asynchronous VLSI circuits. Technical report, DTIC Document, 2000.

[75] A. J. Martin and M. Nystrom. Asynchronous Techniques for System-on-Chip Design. *Proceedings of the IEEE*, 94(6):1089–1120, Jun 2006.

[76] D. Mavis and P. Eaton. SEU and SET Modeling and Mitigation in Deep Submicron Technologies. In *45th Annual IEEE International Reliability physics symposium*, pages 293–305, April 2007.

[77] M. Maza and M. Aranda. Interconnected Rings and Oscillators as Gigahertz Clock Distribution Nets. In *14th Great Lakes Symposium on VLSI (GLSVLSI)*, pages 41–44, 2003.

[78] M. S. Maza and M. L. Aranda. Analysis of Clock Distribution Networks in the Presence of Crosstalk and Groundbounce. In *International IEEE Conference on Electronics, Circuits, and Systems (ICECS)*, pages 773–776, 2001.

[79] D. G. Messerschmitt. Synchronization in Digital System Design. *IEEE Journal on Selected Areas in Communications*, 8(8):1404–1419, 1990.

[80] C. Myers and T. H. Y. Meng. Synthesis of timed asynchronous circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 1(2):106–119, June 1993.

[81] L. W. Nagel and D. Pederson. SPICE (Simulation Program with Integrated Circuit Emphasis). Technical Report UCB/ERL M382, EECS Department, University of California, Berkeley, 1973.

[82] R. Najvirt, M. Függer, T. Nowak, U. Schmid, M. Hofbauer, and K. Schweiger. Experimental Validation of a Faithful Binary Circuit Model. 2015. (appears in Proc. GLSVLSI'15).

[83] R. Naseer and J. Draper. DF-DICE: A scalable solution for soft error tolerant circuit design. *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2006.

[84] S. Nassif, K. Bernstein, D. Frank, A. Gattiker, W. Haensch, B. Ji, E. Nowak, D. Pearson, and N. Rohrer. High Performance CMOS Variability in the 65nm Regime and Beyond. In *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*, pages 569–571, Dec 2007.

[85] A. K. Palit, V. Meyer, W. Anheier, and J. Schloeffel. Modeling and Analysis of Crosstalk Coupling Effect on the Victim Interconnect Using the ABCD Network Model. In *19th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT)*, pages 174–182, Oct 2004.

[86] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal of the ACM*, 27:228–234, 1980.

[87] M. Peercy and P. Banerjee. Fault Tolerant VLSI Systems. *Proceedings of the IEEE*, 81(5):745–758, May 1993.

[88] M. O. Rabin. Randomized byzantine generals. In *24th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 403–409, 1983.

[89] P. Shivakumar, M. Kistler, S. Keckler, D. Burger, and L. Alvisi. Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic. *International Conference on Dependable Systems and Networks (DSN)*, pages 389–398, 2002.

[90] T. K. Srikanth and S. Toueg. Optimal Clock Synchronization. *Journal of the ACM*, 34(3):626–645, 1987.

[91] K. Stevens, R. Ginosar, and S. Rotem. Relative timing [asynchronous design]. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(1):129–140, Feb 2003.

[92] Synopsis. CCS Timing. Technical white paper v2.0, 2006.

[93] P. Teehan, M. Greenstreet, and G. Lemieux. A Survey and Taxonomy of GALS Design Styles. *IEEE Design and Test of Computers*, 24(5):418–428, 2007.

[94] S. H. Unger. Asynchronous Sequential Switching Circuits with Unrestricted Input Changes. *IEEE Transactions on Computers*, 20(12):1437–1444, 1971.

[95] A. Yakovlev, M. Kishinevsky, A. Kondratyev, L. Lavagno, and M. Pietkiewicz-Koutny. On the models for asynchronous circuit behaviour with OR causality. *Formal Methods in System Design*, 9(3):189–233, 1996.

[96] C. Yeh, G. Wilke, H. Chen, S. Reddy, H. Nguyen, T. Miyoshi, W. Walker, and R. Murgai. Clock Distribution Architectures: a Comparative Study. In *7th Symposium on Quality Electronic Design (ISQED)*, pages 85–91, 2006.

[97] T. Yoneda and C. Myers. Synthesis of Timed Circuits Based on Decomposition. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(7):1177–1195, July 2007.

The Education Column

BY

Juraj Hromkovič

Department of Computer Science
ETH Zürich
Universitätstrasse 6, 8092 Zürich, Switzerland
`juraj.hromkovic@inf.ethz.ch`

Informatics – New Basic Subject

Walter Gander
Department of Computer Science
ETH Zürich
gander@inf.ethz.ch

Abstract

Informatics, as Computer Science is called in Europe, has become a leading science. It is high time that it be adopted as a basic subject in schools like mathematics or physics. We discuss in this article some recent developments in Europe concerning informatics in schools.

1 Computers have been invented for *computing!*

The first computers were calculating machines designed to solve engineering problems faster and with fewer errors. Consider for instance two typical representatives of computer pioneers:

1. Howard Aiken (1900-1973), a physicist, who encountered a system of differential equations during his PhD studies in 1939 which could not be solved analytically. He therefore needed to compute a numerical approximation, a tedious work by hand calculations.

   He envisioned an electro-mechanical computing device that could do much of the tedious work for him. This computer was originally called the ASCC (Automatic Sequence Controlled Calculator) and later renamed Harvard Mark I. With engineering, construction, and funding from IBM, the machine was completed and installed at Harvard in February, 1944.[1]

2. Konrad Zuse (1910-1995), civil engineer, had to solve linear equations for static calculations. This tedious calculations motivated him to think about constructing a machine to do this work. Unlike Aiken he did not look for a sponsor but installed 1936 a workshop for constructing a computer in the living room of his parents! [8]

[1]http://en.wikipedia.org/wiki/Howard_H._Aiken

> *His greatest achievement was the world's first programmable computer; the functional program-controlled Turing-complete Z3 became operational in May 1941. Thanks to this machine and its predecessors, Zuse has often been regarded as the inventor of the modern computer.*[2]

2 From Numerical Computing to Information Processing

After World War II the interest in developing and using computers grew. Pioneers in several countries started to built their own machines: in USA, in the UK and also in Switzerland and in Germany. ETH Zürich built 1950-1956 the ERMETH, TUM the Technical University in Munich developed 1952-1956 the PERM. Heinz Rutishauser (ETH) and Friedrich L. Bauer (TUM) developed together with GAMM[3] and ACM[4] members ALGOL60 [2, 3]. ALGOL60 was focussed on numerical calculations. It can be considered as the "Latin of the programming languages" since many features of ALGOL60 have been inherited into modern programming languages.

Computers are not only fast calculators, capable of performing millions of operations per second, but have also the capability to store vast amounts of data, coded as bit-strings. Manipulating such data by storing and retrieving information, sorting and interpreting their contents required new programming languages and finally changed the calculating machines to information processing agents. George Forsythe, the founder of the Computer science Department in Stanford and one of the fathers of Silicon Valley wrote already in 1963 [4]:

> *Machine-held strings of binary digits can simulate a great many kinds of things, of which numbers are just one kind. For example,* they can simulate automobiles on a freeway, chess pieces, electrons in a box, musical notes, Russian words, patterns on a paper, human cells, colors, electrical circuits, and so on. *To think of a computer as made up essentially of numbers is simply a carryover from the successful use of mathematical analysis in studying models*
> ...
> Enough is known already of the diverse applications of computing for us to recognize the birth of a coherent body of technique, which I call computer science.

[2]`http://en.wikipedia.org/wiki/Konrad_Zuse`

[3]International Association of Applied Mathematics and Mechanics `https://www.gamm-ev.de/`

[4]Association for Computing Machinery `https://www.acm.org/`

Today, some 50 years later, *computers determine our lives*, we indeed live in a *digital world*. All our communication by cell-phone, e-mail, sms, through social networks is digital and based on computers. We are writing using text-processing programs and spreadsheets and use presentation tools for our lectures. Books are available as electronic files: the *Digital Book Index*, for instance, provides links to more than 165'000 full-text digital books from more than 1'800 commercial and non-commercial publishers, universities, and various private sites. More than 140'000 of these books, texts, and documents are available free[5]. Also the way we store music has changed. Vinyl records have been replaced, music is digitized and a memory stick can store the music of a whole former collection of records. Radio and television have also been digitized. It became possible to listen or watch missed emissions on the Internet. The video portal *YouTube* added another dimension by providing a platform for uploading digital files and making available everybody's own videos. Photography has changed completely: software has replaced chemically processed films. Finally, search machines have revolutionized the way we acquire our information – *Wikipedia* has become our encyclopedia, libraries and archives are digitized and became accessible on-line.

3 CS Education in Switzerland

In spite of the impressive digital revolution, computer science was for a long time not recognized by academia as a new basic science in Switzerland. It took years till the pressure from industry was large enough to convince the ETH management to finally introduce 1981 a curriculum for computer science studies at ETH.

In schools the computer was first seen as calculator. Pocket calculators replaced gradually the slide rule. It was only with the advent of the personal computer (PC) around 1984 that those responsible for education showed interest to introduce computers in schools. Committees were formed to develop teaching material and to study how to integrate computer science as a new subject in the STEM-oriented tracks of the Swiss Gymnasia. At that time a PC was essentially bare of software, thus it was necessary to develop applications by own programs, often written in Pascal [5, 6]. Spirit of discovery and creativity inspired the students, though at the same time many teachers were frustrated by frequent breakdowns and system changes. Many of those enthusiastic former high-school students – the first generation educated in computer science – are now successful computer scientists.

In the following years many applications were developed. Computers became ubiquitous, easier to handle and there was no need to program own applications.

[5]`http://www.digitalbookindex.com/about.htm`

The Internet was born and connected the world. Therefore a strong movement emerged 1995 in Switzerland that teaching programming in schools was no longer necessary. Instead one should concentrate on teaching skills to make good use of the computers. As the applications became more sophisticated and more complex, teachers had to be trained first. The computer industry, in particular Microsoft and Intel, made agreements with whole countries to train the schools on their products[6].

This paradigm change, shifting away from constructing programs toward just consuming and applying programs had a disastrous effect. It lead to an complete wrong image what computer science is. Skills were overemphasized and fundamentals of the discipline completely neglected. Many freshman entered ETH for computer science studies only to drop out soon because of their completely wrong picture of informatics as a science. Unease grew in industry and academia.

ICTSwitzerland[7] the umbrella organization of the Swiss IT-industry urged in 2010 with a memorandum to introduce real computer science in schools not only to teach how to use computers. The Hasler-Foundation[8] supported with the project *FIT in IT* in the last ten years the introduction of computer science as basic science in schools. In Fall 2014 finally the ministers of educations of the 21 German speaking Cantons in Switzerland decided to introduce informatics as a basic subject in all schools.

4 European Initiative

Computer science education developed differently in Europe. Eastern Europe kept focusing on fundamentals and on programming while Western states concentrated on just using computers with similar consequences as in Switzerland.

Informatics Europe[9] and *ACM Europe*[10] created a common task force to study that problem. The task force completed a report in 2013 on Informatics Education in Schools with the title *Informatics education: Europe cannot afford to miss the boat*[1].

At first it was necessary to define terms since many persons equate computer science with using new media. Fruitless discussions and disagreements are un-

[6]See for instance `http://www.saarland.de/17657.htm`

[7]`http://www.ictswitzerland.ch`

[8]The purpose of the Hasler Foundation is to promote information and communications technology (ICT) for the well-being and benefit of Switzerland as an intellectual and industrial centre.`http://www.haslerstiftung.ch/en/home`

[9]The association of computer science departments and research laboratories in Europe and neighbouring areas. `http://www.informatics-europe.org/`

[10]European Chapter of ACM, the world's largest educational and scientific computing society. `http://europe.acm.org/`

avoidable when people use the same words for different meanings, contents and in different contexts. The task force defined therefore

Computer Science in Schools = *Digital Literacy* + *Informatics*

and specified:

- *Digital literacy* covers fluency with computer tools and the Internet.
- *Informatics* covers the science behind information technology. Informatics is a distinct science, characterized by its own concepts, methods, body of knowledge and open issues. It has emerged, in a role similar to that of mathematics, as a cross-discipline field underlying today's scientific, engineering and economic progress.

The report makes the following observations:

- Informatics is a major enabler of technology innovation, the principal resource for Europe's drive to become an information society, and the key to the future of Europe's economy.
- European countries are making good progress in including digital literacy in the curriculum. The teaching of this topic should emphasize the proper use of information technology resources and cover matters of ethics such as privacy and plagiarism.
- Informatics education, unlike digital literacy education, is sorely lacking in most European countries. The situation has paradoxically worsened since the 70s and 80s.
- Not offering appropriate informatics education means that Europe is harming its new generation of citizens, educationally and economically.
- Unless Europe takes resolute steps to change that situation, it will turn into a mere consumer of information technology and miss its goal of being a major player.

Therefore the report makes the following recommendations

1. All students should benefit from education in digital literacy, starting from an early age and mastering the basic concepts by age 12. Digital literacy education should emphasize not only skills but also the principles and practices of using them effectively and ethically.

2. All students should benefit from education in informatics as an independent scientific subject, studied both for its intrinsic intellectual and educational value and for its applications to other disciplines.

3. A large-scale teacher training program should urgently be started. To bootstrap the process in the short term, creative solutions should be developed involving school teachers paired with experts from academia and industry.

4. The definition of informatics curricula should rely on the considerable body of existing work on the topic and the specific recommendations of the present report.

The report triggered similar activities in France and Germany. The *Académie des Sciences* published in May 2013 a report with the title "*L'enseignement de l'informatique en France. Il est urgent de ne plus attendre*".[11] A resolution was taken in November 2013 at the *Fakultätentag Informatik* in Germany with the title "*Informatik in der Schulbildung: Wir dürfen den Anschluss nicht verlieren!*".[12]

A parallel development happened in the UK. Michael Gove, the Secretary of State for Education, lamented the state of computer science education in his famous speech of January 2012[13]

> The UK had been let down by an ICT curriculum that *neglects the rigorous computer science and programming skills* which high-tech industries need.
> ...
> In short, just at the time when technology is bursting with potential, teachers, professionals, employers, universities, parents and pupils are all telling us the same thing: *ICT in schools is a mess.*

Michael Gove urges to reform the computer science education in the UK and refers to the PC-area when the computer inspired discovery and creativity of the students who were fascinated by using and exploring this new tool:

> With minimal memory and no disk drives, the Raspberry Pi computer can operate basic programming languages, handle tasks like spread sheets, word-processing and games, and connect to wifi via a dongle – all for between £16 and £22. This is a great example of the cutting edge of education technology happening right here in the UK. It could bring the same excitement as the BBC Micro did in the 1980s.

[11] `http://www.academie-sciences.fr/activite/publi.htm`
[12] `https://www.ft-informatik.de/52.html`
[13] `https://www.gov.uk/government/speeches/michael-gove-speech-at-the-bett-show-2012`

Since Fall 2014 big changes are going on in the UK in informatics education. Simon Peyton Johnes describes in his TED talk with the title "*Teaching creative computer science*"[14] on YouTube the changes and the motivation for teaching informatics. On his slides he make the distinction between *skills*=ICT and *discipline*=informatics. The new concept for teaching is illustrated in this table:

What we want instead

Ideas	as well as	technology
Create	as well as	consume
Write	as well as	read
Understand	as well as	use
Knowledge	rather than	magic

5 Why Informatics as Basic Subject in Schools?

The goals of informatics as a basic subject is

1. To teach the students to understand the principles and functioning of today's digital world.

2. To train the students in constructive problem solving.

Jan Cuny, Larry Snyder, and Jeannette M. Wing coined the term "Computational Thinking"[15]. What they mean is:

> Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent.

Jeannette M. Wing wrote in [7]:

> It [Computational Thinking] represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use.

Computational thinking is a methodology for anyone to be used to solve problems. Especially it applies to problems solving with computers. It involves the following steps

- *Analyze a task or problem*, model and formalize it.

[14]https://www.youtube.com/watch?v=Ia55clAtdMs
[15]`http://www.cs.cmu.edu/~CompThink/`

- Search for a way to solve it, find or design an *algorithm*.
- *Program*.
- *Run the program*: let the computer work, maybe correct, modify the program, *interpret the results*.

Programming is an essential step in this process. It is an important activity for all the students in general education. It is *creative* since there are often many ways to solve a problem. It is *constructive*. The solution has to be constructed like an engineer constructs a machine. Running the program is like starting a virtual machine. Programming finally teaches *precise working* since any small error prevents the solution and trains *computational thinking*.

6 Conclusions

Computer Science is the leading science of the 21st century. It is used like mathematics in all sciences. It has to become part of general knowledge in education.

Informatics Europe and ACM Europe convincingly state that computer science in the school must consist of two parts:

1. Learn to make good use of IT and its devices. This is called *Digital Literacy*, often also ICT. These skills are short living knowledge, they change with technology.
2. Learn the fundamentals of computer science which are essential to understand our digital world. This is called *Informatics*. It is long living knowledge which lasts forever and does not change with technology.

References

[1] Informatics Europe and ACM Europe: *Informatics education: Europe cannot afford to miss the boat*. Report of the joint Informatics Europe & ACM Europe Working Group on Informatics Education, *April 2013* `http://www.informatics-europe.org/images/documents/informatics-education-europe-report.pdf`

[2] F. L. Bauer, H. Bottenbruch, H. Rutishauser, K. Samelson. *Proposal for a universal language for the description of computing processes*. In: J. W. Carr (ed.), Computer Programming and Artificial Intelligence, University of Michigan Summer School 1958, pages 355-373.

[3] J. W. Backus, F. L. Bauer, J. Green, C. Katz, J. McCarthy, P. Naur, A .J. Perlis, H. Rutishauser, K. Samelson, B. Vauquois, J. H. Wegstein, A. van Wijngaarden, M. Woodger; edited by Peter Naur. *Revised Report on the Algorithmic Language ALGOL 60.*

[4] George Forsythe. *Educational implications of the computer revolution*. Applications of Digital Computers, W. F. Freiberger and William Prager (eds.), Ginn, Boston, 1963, pp. 166-178.

[5] Kathleen Jensen, Niklaus Wirth. *Pascal User Manual and Report*. ISO Pascal Standard. Springer-Verlag, 4th ed. 1991,

[6] `http://www.emsps.com/oldtools/borpasv.htm`

[7] Jeannette M. Wing: Computational Thinking, COMMUNICATIONS OF THE ACM, Vol. 49, No. 3, (2006)

[8] Zuse, Konrad: The Computer – My Life. Springer-Verlag, (1993)

The Formal Language Theory Column

by

Giovanni Pighizzini

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

AVERAGE SIZE OF AUTOMATA CONSTRUCTIONS FROM REGULAR EXPRESSIONS[*]

Sabine Broda[†]
sbb@dcc.fc.up.pt

António Machiavelo[†]
ajmachia@fc.up.pt

Nelma Moreira[†]
nam@dcc.fc.up.pt

Rogério Reis[†]
rvr@dcc.fc.up.pt

Abstract

Because of their succinctness and clear syntax, regular expressions are the common choice to represent regular languages. Deterministic finite automata are an excellent representation for testing equivalence, containment or membership, as these problems are easily solved for this model. However, minimal deterministic finite automata can be exponentially larger than the associated regular expression, while the corresponding nondeterministic finite automata can be linearly larger. The worst case of both the complexity of the conversion algorithms, and of the size of the resulting automata, are well studied. However, for practical purposes, estimates for the average case can provide much more useful information. In this paper we review recent results on the average size of automata resulting from several constructions and suggest several directions of research. Most results were obtained within the framework of analytic combinatorics.

1 Introduction

The methods to convert regular expressions (REs) into equivalent automata can be divided in three major classes, depending on whether the resulting automaton is deterministic (DFA), nondeterministic without ε-transitions (NFA) or nondeterministic with ε-transitions (ε-NFA). Paradigmatic methods of each class are Brzozowski's [13], Glushkov's [21] and Thompson's [47] constructions, respectively.

[*]This work was partially supported by CMUP (UID/MAT/00144/2013), which is funded by FCT (Portugal) with national (MEC) and European structural funds through the programs FEDER, under the partnership agreement PT2020.

[†]CMUP, DCC & DM, Faculdade de Ciências da Universidade do Porto, Portugal.

Brzozowski's method introduces the notion of derivative of a regular expression with respect to a symbol, a syntactic equivalent of the notion of left quotient for languages. It is well-known that regular languages have a finite number of left-quotients. To obtain a finite number of derivatives, it is necessary to consider regular expressions modulo some equational axioms, namely the associativity, commutativity and idempotence of union (ACI). A nondeterministic version of derivatives was introduced by Mirkin [35] and Antimirov [2]. Instead of a derivative being a regular expression, a set of regular expressions (partial derivatives) is considered. This avoids the necessity of using derivatives modulo equational axioms, but the associated construction (partial derivative automata) yields NFAs instead of DFAs.

Glushkov construction uses the positions of the letters occurring in a regular expression. The partial derivative automaton is a quotient of the Glushkov (or position) automaton, and very often is its smallest quotient [15, 22, 31]. If ε-transitions are eliminated from the Thompson automaton, the Glushkov automaton is obtained. Finally, the determination of the Glushkov automaton (by subset construction) produces the McNaughton-Yamada DFA [34].

The worst case of both the complexity of the conversion algorithms, and of the size of the resulting automata, are well studied [12, 16, 15] (see also [25] for a survey). However, for practical purposes, an estimate for the average case constitute a much more useful information.

In this paper we review recent results on the average size of automata resulting from several constructions, and suggest several directions of research. Most results were obtained within the framework of analytic combinatorics, a powerful tool for asymptotic average analysis, by relating the enumeration of combinatorial objects to the algebraic and complex analytic properties of generating functions. An introduction to this method, and a derivation of the asymptotic average size of several conversions between regular expressions and ε-NFAs, can be found in Broda *et al* [9]. Another approach to average complexity is to consider uniform random generators and to perform statistically significant experiments. The drawback of this approach is that it only gives results for a small range of object sizes and, due to their combinatorial nature, only modest ranges can usually be considered. However, whenever we refer to experimental results we mean results obtained in this limited context. Both in experimental and analytic results we consider the average with respect to the uniform distribution.

In the conversions from regular expressions to NFAs without ε-transitions, although position based methods can provide recursive definitions that endow analytic analysis, we will emphasise the role of derivatives when considering extended regular expressions, or other algebraic structures such as Kleene algebras with tests.

We briefly review some basic definitions about regular expressions and finite automata. For more details, we refer the reader to Kozen [27] or Sakarovitch [44]. The set R of *regular expressions* over a finite alphabet Σ is the smallest set containing $\emptyset$ and all the expressions generated by the following grammar:

$$\alpha \quad := \quad \varepsilon \mid \sigma_1 \mid \cdots \mid \sigma_k \mid (\alpha + \alpha) \mid (\alpha \cdot \alpha) \mid \alpha^\star \tag{1}$$

where $\sigma_i \in \Sigma$ are *letters* and the operator $\cdot$ (concatenation) is often omitted. The language $\mathcal{L}(\alpha) \subseteq \Sigma^\star$ associated to α is inductively defined as $\mathcal{L}(\varepsilon) = \{\varepsilon\}$, $\mathcal{L}(\sigma) = \{\sigma\}$ for $\sigma \in \Sigma$, $\mathcal{L}((\alpha + \beta)) = \mathcal{L}(\alpha) \cup \mathcal{L}(\beta)$, $\mathcal{L}((\alpha \cdot \beta)) = \mathcal{L}(\alpha) \cdot \mathcal{L}(\beta)$, and $\mathcal{L}(\alpha^\star) = \mathcal{L}(\alpha)^\star$. Also, $\mathcal{L}(\emptyset) = \emptyset$. The *size* $|\alpha|$ of $\alpha \in \mathsf{R}$ is the number of symbols in α, where parentheses are not counted; the *alphabetic size* $|\alpha|_\Sigma$ is its number of letter occurrences. We define $\varepsilon(\alpha)$ as $\varepsilon(\alpha) = \varepsilon$ if $\varepsilon \in \mathcal{L}(\alpha)$, and $\varepsilon(\alpha) = \emptyset$ otherwise. Two regular expressions α and β are *equivalent* if $\mathcal{L}(\alpha) = \mathcal{L}(\beta)$, and we write $\alpha = \beta$. With this interpretation, the algebraic structure $(\mathsf{R}, +, \cdot, \emptyset, \varepsilon)$ constitutes an idempotent semiring, and with the unary operator $\star$, a Kleene algebra (KA). Given a language $L \subseteq \Sigma^\star$ and a word $w \in \Sigma^\star$, the *left-quotient* of L w.r.t. w is the language $w^{-1}L = \{\ x \mid wx \in L\ \}$. A *nondeterministic finite automaton* (NFA) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ where Q is a finite set of states, Σ is the alphabet, $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ is the transition relation, $I \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of final states. When $I = \{q_0\}$, we just write q_0 instead of $\{q_0\}$. The *size* of an NFA, $\mathcal{A}$, is $|\mathcal{A}| = |Q| + |\delta|$, the number of states $|\mathcal{A}|_Q = |Q|$, and the number of transitions $|\mathcal{A}|_\delta = |\delta|$. An NFA that has transitions labelled with ε is an ε-NFA. An NFA is *deterministic* (DFA) if $|I| = 1$ and for each pair $(q, \sigma) \in Q \times \Sigma$ there exists at most one q' such that $(q, \sigma, q') \in \delta$. The *language* accepted by an automaton $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) = \{\ w \in \Sigma^* \mid \delta(I, w) \cap F \neq \emptyset\ \}$. An equivalence relation E over Q is *right invariant* (or a *bisimulation*) if $E \subseteq (Q \setminus F)^2 \cup F^2$, and for any $p, q \in Q$, $\sigma \in \Sigma$ if $p\ E\ q$, then[1] $\delta(p, \sigma)/_E = \delta(q, \sigma)/_E$. The quotient automaton $\mathcal{A}/_E = (Q/_E, \Sigma, \delta_E, [q_0]_E, F/_E)$, where $\delta_E = \{\ ([p]_E, \sigma, [q]_E) \mid (p, \sigma, q) \in \delta\ \}$, satisfies $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}/_E)$. The largest bisimulation, *i.e.* the union of all bisimulation relations on Q, is called *bisimilarity* ($\equiv_b$).

2 Generating Functions and Analytic Methods

A *combinatorial class* C is a set of objects on which a non-negative integer function (size) $|\cdot|$ is defined, and such that for each $n \geq 0$, the number of objects of size n, c_n, is finite. The *generating function* $C(z)$ of C is the formal power series $C(z) = \sum_{c \in \mathsf{C}} z^{|c|} = \sum_{n=0}^{\infty} c_n z^n$. We let $[z^n]C(z)$ denote the coefficient of z^n, c_n. The symbolic method [19] is a framework that allows to express a combinatorial class C in terms of simpler ones, $\mathsf{B}_1, \ldots, \mathsf{B}_n$, by means of specific operations, yielding

[1]Denoting by $S/_E$ the set $\{\ [s]_E \mid s \in S\ \}$.

the generating function $C(z)$ as a function of the generating functions $B_i(z)$ of B_i, for $1 \leq i \leq n$. For example, given two disjoint combinatorial classes A and B, with generating functions $A(z)$ and $B(z)$, respectively, the union $A \cup B$ is a combinatorial class whose generating function is $A(z) + B(z)$. Other usual admissible operations are the cartesian product and the Kleene closure.

Multivariate generating functions are used in order to obtain estimates about the asymptotic behaviour of parameters associated to combinatorial classes. Considering t weighting functions, $p_i : C \to \mathbb{C}$, for $1 \leq i \leq t$, let $c_{k_1,\ldots,k_t,n}$ be the number of objects c of size n with $p_1(c) = k_1, \ldots, p_t(c) = k_t$, one has the following multivariate weighting generating function

$$C(u_1, \ldots, u_t, z) = \sum_{n,k_1,\ldots,k_t \geq 0} c_{k_1,\ldots,k_t,n}\, u_1^{k_1} \cdots u_t^{k_t} z^n.$$

The functions $(p_i)_i$ give the respective weights of t features under consideration. Note that since C is a combinatorial class, the number of objects with a given size is finite, and therefore, for a fixed n, there is only a finite number of $c_{k_1,\ldots,k_t,n}$ which are different from 0. Also,

$$\left.\frac{\partial C(u_1, \ldots, u_t, z)}{\partial u_i}\right|_{u_i=1} = \sum_{\substack{n,k_j \geq 0 \\ j \neq i}} \left(\sum_{k_i \geq 0} k_i c_{k_1,\ldots,k_t,n} \right) u_1^{k_1} \cdots u_{i-1}^{k_{i-1}} u_{i+1}^{k_{i+1}} \cdots u_t^{k_t} z^n,$$

where $\sum_{k_i \geq 0} k_i c_{k_1,\ldots,k_t,n}$ accounts for the cumulative presence of weight p_i in the objects of size n.

2.1 Analytic Asymptotics

Generating functions can be seen as complex analytic functions, and the study of their behaviour around their dominant singularities gives us access to an asymptotic estimate for their coefficients. We refer the reader to Flajolet and Sedgewick for an extensive study on this topic. Here we only state the results relevant for this paper. For $\rho \in \mathbb{C}$, $R > 1$ and $0 < \phi < \pi/2$, consider the domain $\Delta(\rho, \phi, R) = \{\ z \in \mathbb{C} \mid |z| < R,\ z \neq \rho, \text{ and } |Arg(z - \rho)| > \phi\ \}$, where $Arg(z)$ denotes the argument of $z \in \mathbb{C}$. A region is a Δ-*domain* at ρ if it is a $\Delta(\rho, \phi, R)$, for some R and ϕ. The generating functions we consider have always a unique dominant singularity, and satisfy one of the two conditions of the following proposition, used by Nicaud [37].

Proposition 1. *Let $f(z)$ be a function that is analytic in some Δ-domain at $\rho \in \mathbb{R}^+$. If at the intersection of a neighborhood of ρ and its Δ-domain,*

1. *$f(z) = a - b\sqrt{1 - z/\rho} + o\left(\sqrt{1 - z/\rho}\right)$, with $a, b \in \mathbb{R}$, $b \neq 0$, then*

$$[z^n]f(z) \sim \frac{b}{2\sqrt{\pi}}\rho^{-n}n^{-3/2}.$$

2. *$f(z) = \frac{a}{\sqrt{1-z/\rho}} + o\left(\frac{1}{\sqrt{1-z/\rho}}\right)$, with $a \in \mathbb{R}$, and $a \neq 0$, then*

$$[z^n]f(z) \sim \frac{a}{\sqrt{\pi}}\rho^{-n}n^{-1/2}.$$

2.2 Generating Functions for Regular Expressions

For the regular expressions given in (1), an average case analysis of different descriptional measures, including the number of letters, has been presented by Nicaud [36, 37]. Here we introduce some of those results, deriving them in a slight different way and using a parametrised weighted generating function based on Broda et al. [9].

Using the recursive definition of R given by (1), the associated generating function $R_k(z)$ satisfies $R_k(z) = (k+1)z + zR_k(z) + 2zR_k(z)^2$. Solving this equation for $R_k(z)$, and considering that $R_k(0) = a_0 = 0$, one obtains $R_k(z) = \frac{1-z-\sqrt{\Delta_k(z)}}{4z}$, where $\Delta_k(z) = 1 - 2z - (7+8k)z^2$. The zeros of $\Delta_k(z)$ are $\rho_k = \frac{1}{1+\sqrt{8k+8}}$ and $\bar{\rho}_k = \frac{1}{1-\sqrt{8k+8}}$. The coefficients of the series of $\tilde{R}_k(z) = 4zR_k(z) + z = 1 - \sqrt{\Delta_k(z)}$, have the same asymptotical behaviour of the ones of $R_k(z)$. Now $\Delta_k(z) = (7+8k)(z-\bar{\rho}_k)\rho_k(1-z/\rho_k)$, and since $(7+8k)(\rho_k - \bar{\rho}_k) = 4\sqrt{2k+2}$, one has

$$\tilde{R}_k(z) = 1 - \sqrt{\Delta_k(z)} = 1 - 2\sqrt[4]{2k+2}\sqrt{\rho_k}\sqrt{1 - z/\rho_k} + o\left(\sqrt{1 - z/\rho_k}\right).$$

By Proposition 1, one obtains

$$[z^n](4zR_k(z) + z) \sim \frac{\sqrt[4]{2k+2}\sqrt{\rho_k}}{\sqrt{\pi}}\rho_k^{-n}n^{-3/2},$$

$$[z^n]R_k(z) \sim \frac{\sqrt[4]{2k+2}\sqrt{\rho_k}}{4\sqrt{\pi}}\rho_k^{-(n+1)}(n+1)^{-3/2}, \tag{2}$$

where $[z^n]R_k(z)$ is the number of regular expressions α with $|\alpha| = n$.

To obtain estimates for the average value relative to some other measures on regular expressions, one can consider parameters $(c_\varepsilon, c_\sigma, c_+, c_\bullet, c_\star)$, where $c_\varkappa$ is the contribution of the operation $\varkappa$ expressed in the measure under consideration. This allows to consider a parametrized weighted generating function and an asymptotic estimation of its coefficients. For each set of parameters, one obtains estimations for the associated measure, such as number of letters, number of operators of a given type (concatenation, star, etc) and, as we will see in Section 3.1, the size of some automata constructions.

The general bivariate generating function, corresponding to those parameters, satisfies the following equation

$$C_k(u,z) = (u^{c_\varepsilon} + ku^{c_\sigma})z + (u^{c_+} + u^{c_\bullet})zC_k(u,z)^2 + u^{c_\star}zC_k(u,z).$$

Solving this equation for $C_k(u,z)$, and choosing the root that has positive coefficients, one sees that

$$C_k(u,z) = \frac{1 - u^{c_\star}z - \sqrt{(1 - u^{c_\star}z)^2 - 4(ku^{c_\sigma} + u^{c_\varepsilon})(u^{c_+} + u^{c_\bullet})z^2}}{2(u^{c_+} + u^{c_\bullet})z}.$$

Deriving in order to u, and taking $u = 1$, the cumulative generating function obtained is

$$C_k(z) = \frac{a_k(z)\sqrt{\Delta_k(z)} + b_k(z)}{8z\sqrt{\Delta_k(z)}}, \tag{3}$$

where $\Delta_k(z)$ is as above, and

$$\begin{aligned} a_k(z) = {} & (c_+ + c_\bullet - 2c_\star)\,z - (c_+ + c_\bullet) \\ b_k(z) = {} & (4\,(2c_\sigma - c_+ - c_\bullet)\,k + 8c_\varepsilon - 3(c_+ + c_\bullet) - 2c_\star)\,z^2 + \\ & + 2\,(c_\star - c_+ - c_\bullet)\,z + c_+ + c_\bullet. \end{aligned}$$

Considering $G_k(z) = 8z\,C_k(z) - a_k(z) = \frac{b_k(z)}{\sqrt{\Delta_k(z)}}$, proceeding in a similar way to what was done to $R_k(z)$, and applying Proposition 1, one obtains

$$[z^n]G_k(z) \sim \frac{b_k(\rho_k)}{2\sqrt[4]{2k+2}\,\sqrt{\rho_k}\,\sqrt{\pi}}\,\rho_k^{-n} n^{-\frac{1}{2}}. \tag{4}$$

One concludes that for $n \geq 2$,

$$\begin{aligned} [z^n]C_k(z) \quad & \sim \quad \frac{b_k(\rho_k)}{16\sqrt[4]{2k+2}\,\sqrt{\rho_k}\,\sqrt{\pi}}\,\rho_k^{-(n+1)}(n+1)^{-\frac{1}{2}} & (5) \\ & = \quad \frac{c_\star\sqrt{2k+2} + (c_+ + 2c_\sigma + c_\bullet)k + (c_+ + 2c_\varepsilon + c_\bullet)}{4\sqrt{\pi}\sqrt[4]{2k+2}}\rho_k^{\frac{1}{2}-n}(n+1)^{-\frac{1}{2}}. & (6) \end{aligned}$$

The following expression gives, for $n \geq 2$, the parametrised asymptotic estimate for the average size, for a given measure, for regular expressions of size n.

$$\begin{aligned} \frac{[z^n]C_k(z)}{[z^n]R_k(z)} \quad & \sim \quad \frac{b_k(\rho_k)}{4\rho_k\sqrt{2k+2}}\,(n+1) = \\ & = \quad \rho_k\left(c_\star + (c_+ + c_\bullet)\sqrt{\frac{k+1}{2}} + (c_\sigma k + c_\varepsilon)\sqrt{\frac{2}{k+1}}\right)(n+1). \end{aligned}$$

Thus, for the considered measure, the average of its value per character of the original regular expression is, asymptotically,

$$\lim_{n\to\infty} \frac{[z^n]C_k(z)}{n[z^n]R_k(z)} = \rho_k \left(c_\star + (c_+ + c_\bullet)\sqrt{\frac{k+1}{2}} + (c_\sigma k + c_\varepsilon)\sqrt{\frac{2}{k+1}} \right). \tag{7}$$

The generating function for the number of letters occurring in a regular expression, $Let_k(z)$, can be obtained from (3) by making $c_\sigma = 1$ and null all the other parameters, giving

$$Let_k(z) = \frac{kz}{\sqrt{\Delta_k(z)}}, \tag{8}$$

and (6) yields

$$[z^n]Let_k(z) \sim \frac{k\sqrt{\rho_k}}{2\sqrt{\pi}\sqrt[4]{2k+2}}\rho_k^{-n}(n+1)^{-\frac{1}{2}}, \tag{9}$$

from which it follows that

$$\frac{[z^n]Let_k(z)}{n[z^n]R_k(z)} \sim \frac{2k\rho_k}{\sqrt{2k+2}} \xrightarrow[k\to\infty]{} \frac{1}{2}. \tag{10}$$

In the same manner one can easily obtain approximate values for the number of concatenations, disjunctions and stars.

3 Regular Expressions to NFAs

Because of their succinctness and clear syntax, regular expressions are the common choice to represent regular languages. DFAs are an excellent representation for testing equivalence, containment or membership, as these problems are easily solved for this model. For instance, recognition of a word w is $O(|w|)$ for DFAs, while it is $O(|w| \cdot |Q|^2)$ for NFAs. However, minimal DFAs can be exponentially larger than the associated REs, while the corresponding NFAs can be linearly larger. Since NFA minimisation is PSPACE-complete, the aim is to obtain directly from the REs small NFAs usable for practical purposes. In this section we summarise the known results on the average size of different NFA constructions from REs.

3.1 Average Size of ε-NFAs

We consider here three constructions, introduced respectively by Thompson in 1968 [47] ($\mathcal{A}_T$), by Sippu and Soisalon-Soininen in 1990 [46] ($\mathcal{A}_{SSS}$), and by Ilie and Yu in 2003 [26] ($\mathcal{A}_{\varepsilon-fol}$), that transform a regular expression α into an equivalent ε-NFA. Generically, denoting the result by $\mathcal{N}_\alpha$, all three algorithms associate with the (atomic) regular expressions ε and σ the same ε-NFAs, as given in Figure 1.

$\mathcal{N}_\varepsilon$:
ε
$\mathcal{N}_\sigma$:
σ

Figure 1: ε-NFAs for atomic expressions

Thus, for all three constructions we have

$$\begin{aligned} |\mathcal{N}_\varepsilon| &= |\mathcal{N}_\varepsilon|_Q + |\mathcal{N}_\varepsilon|_\delta &&= 2+1 &&= 3\\ |\mathcal{N}_\sigma| &= |\mathcal{N}_\sigma|_Q + |\mathcal{N}_\sigma|_\delta &&= 2+1 &&= 3. \end{aligned}$$

The ε-NFA's for compound regular expressions are constructed inductively from the automata corresponding to their subexpressions. In Thompson's construction, the automaton $\mathcal{N}_{\beta_1+\beta_2}$ (in Figure 2) is built from $\mathcal{N}_{\beta_1}$ and $\mathcal{N}_{\beta_2}$ introducing a new initial state with ε-transitions to the initial states of both $\mathcal{N}_{\beta_1}$ and $\mathcal{N}_{\beta_2}$, as well as a new final state and ε-transitions from the final states of $\mathcal{N}_{\beta_1}$ and $\mathcal{N}_{\beta_2}$. It follows that this construction introduces exactly 2 new states and 4 new transitions for each + operator. Thus, we have

$$\begin{aligned} |\mathcal{N}_{\beta_1+\beta_2}|_Q &= |\mathcal{N}_{\beta_1}|_Q + |\mathcal{N}_{\beta_2}|_Q + 2\\ |\mathcal{N}_{\beta_1+\beta_2}|_\delta &= |\mathcal{N}_{\beta_1}|_\delta + |\mathcal{N}_{\beta_2}|_\delta + 4, \end{aligned} \tag{11}$$

and consequently,

$$|\mathcal{N}_{\beta_1+\beta_2}| = |\mathcal{N}_{\beta_1}| + |\mathcal{N}_{\beta_2}| + 6. \tag{12}$$

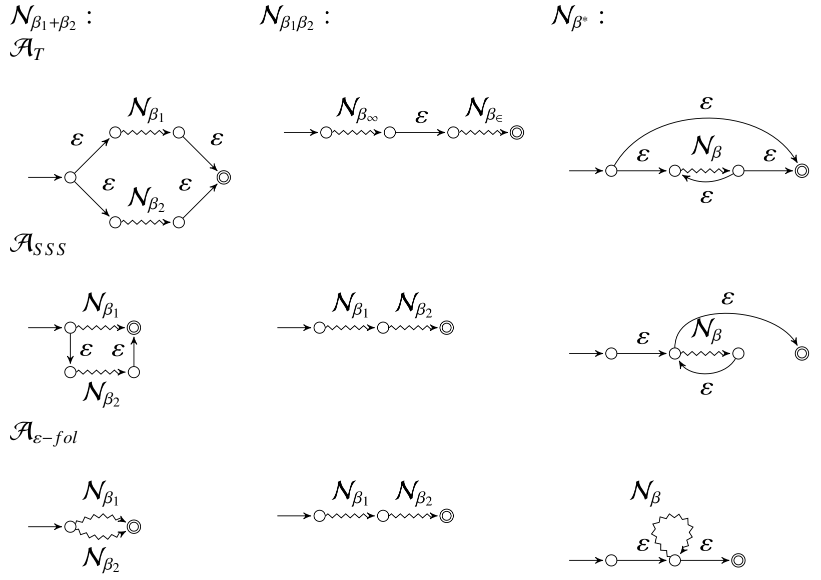
The remaining construction cases are presented in Figure 2.

To obtain the weighted generating function for the size of the resulting automata, according to a given measure, we consider the parameters $(c_\varepsilon, c_\sigma, c_+, c_\bullet, c_\star)$ represented in Table 1.

ε-NFAs	States	Transitions	Combined Size
$\mathcal{A}_T$	$(2,2,2,0,2)$	$(1,1,4,1,4)$	$(3,3,6,1,6)$
$\mathcal{A}_{SSS}$	$(2,2,0,-1,2)$	$(1,1,2,0,3)$	$(3,3,2,-1,5)$
$\mathcal{A}_{\varepsilon-fol}$	$(2,2,-2,-1,1)$	$(1,1,0,0,2)$	$(3,3,-2,-1,3)$

Table 1: Parameters for the 3 constructions

Now, note that for all the constructions here considered, the worst-case complexity is reached for expressions with only one letter and $n-1$ stars. For such an expression (which has size n), the size of the corresponding $\mathcal{A}_T$, $\mathcal{A}_{SSS}$ and $\mathcal{A}_{\varepsilon-fol}$ automaton is, respectively, $6n-3$, $5n-2$ and $3n$. In Table 2, we illustrate



$\mathcal{N}_{\beta_1+\beta_2}$:
$\mathcal{N}_{\beta_1\beta_2}$:
$\mathcal{N}_{\beta^*}$:
$\mathcal{A}_T$
$\mathcal{A}_{SSS}$
$\mathcal{A}_{\varepsilon-fol}$

Figure 2: $\mathcal{A}_T$, $\mathcal{A}_{SSS}$ and $\mathcal{A}_{\varepsilon-fol}$ constructions

the discrepancy between the average and the worst case, for the combined size, by presenting the values of the expression in Equation (7) for different values of k, the limit as k goes to infinity, and the size of the worst case, which does not depend on k. As the alphabet grows, the size of the obtained ε-NFA's is much smaller, asymptotically and on average, than in the worst case. For instance, in the case of the $\mathcal{A}_{\varepsilon-fol}$, the ratio between these values is 0.25. This construction also exhibits the best behaviour of the three.

k	2	10	50	100	∞	worst-case
$\mathcal{A}_T$	3.72	3.51	3.38	3.34	3.25	6
$\mathcal{A}_{SSS}$	2.30	2.06	1.90	1.86	1.75	5
$\mathcal{A}_{\varepsilon-fol}$	1.13	0.97	0.86	0.83	0.75	3

Table 2: Average vs. worst-case combined size

3.2 Average Size of the Glushkov Automata

Let $\text{Pos}(\alpha) = \{1, 2, \ldots, |\alpha|_\Sigma\}$ be the set of positions for $\alpha \in \mathsf{R}$, and let $\text{Pos}_0(\alpha) = \text{Pos}(\alpha) \cup \{0\}$. We consider the expression $\overline{\alpha}$ obtained by marking each letter with its position in α, *i.e.* $\mathcal{L}(\overline{\alpha}) \in \overline{\Sigma}^\star$, where $\overline{\Sigma} = \{\, \sigma^i \mid \sigma \in \Sigma, 1 \leq i \leq |\alpha|_\Sigma \,\}$. For $\alpha \in \mathsf{R}$ and $i \in \mathrm{Pos}(\alpha)$, let the sets *first*, *last* and *follow* be $\mathsf{ft}(\alpha) = \{\, i \mid \exists w \in \overline{\Sigma}^\star, \sigma^i w \in \mathcal{L}(\overline{\alpha}) \,\}$, $\mathsf{lt}(\alpha) = \{\, i \mid \exists w \in \overline{\Sigma}^\star, w\sigma^i \in \mathcal{L}(\overline{\alpha}) \,\}$ and $\mathsf{fw}(\alpha, i) = \{\, j \mid \exists u, v \in \overline{\Sigma}^\star, u\sigma^i\sigma^j v \in \mathcal{L}(\overline{\alpha}) \,\}$, respectively. These sets can be inductively defined as follows:

$$\begin{array}{lcl} \mathsf{ft}(\emptyset) & = & \mathsf{ft}(\varepsilon) = \emptyset \\ \mathsf{ft}(\sigma_i) & = & \{i\} \\ \mathsf{ft}(\alpha^\star) & = & \mathsf{ft}(\alpha) \end{array} \quad \begin{array}{lcl} \mathsf{ft}(\alpha + \beta) & = & \mathsf{ft}(\alpha) \cup \mathsf{ft}(\beta) \\ \mathsf{ft}(\alpha\beta) & = & \begin{cases} \mathsf{ft}(\alpha) \cup \mathsf{ft}(\beta) & \text{if } \varepsilon(\alpha) = \varepsilon \\ \mathsf{ft}(\alpha) & \text{otherwise,} \end{cases} \end{array} \tag{13}$$

$$\begin{aligned} \mathsf{fw}(\emptyset) &= \mathsf{fw}(\varepsilon) = \mathsf{fw}(\sigma_i) = \emptyset \\ \mathsf{fw}(\alpha\beta) &= \mathsf{fw}(\alpha) \cup \mathsf{fw}(\beta) \\ \mathsf{fw}(\alpha\beta) &= \mathsf{fw}(\alpha) \cup \mathsf{fw}(\beta) \cup \mathsf{lt}(\alpha) \times \mathsf{ft}(\beta) \\ \mathsf{fw}(\alpha^\star) &= \mathsf{fw}^\star(\alpha) \\ \mathsf{fw}^\star(\emptyset) &= \mathsf{fw}^\star(\varepsilon) = \emptyset \\ \mathsf{fw}^\star(\sigma_i) &= \{(i,i)\} \\ \mathsf{fw}^\star(\alpha + \beta) &= \mathsf{fw}^\star(\alpha) \cup \mathsf{fw}^\star(\beta) \cup \mathsf{cs}(\alpha,\beta) \\ \mathsf{fw}^\star(\alpha\beta) &= \begin{cases} \mathsf{fw}^\star(\alpha) \cup \mathsf{fw}^\star(\beta) \cup \mathsf{cs}(\alpha,\beta) & \text{if } \varepsilon(\alpha) = \varepsilon(\beta) = \varepsilon \\ \mathsf{fw}^\star(\alpha) \cup \mathsf{fw}(\beta) \cup \mathsf{cs}(\alpha,\beta) & \text{if } \varepsilon(\beta) = \varepsilon \\ \mathsf{fw}(\alpha) \cup \mathsf{fw}^\star(\beta) \cup \mathsf{cs}(\alpha,\beta) & \text{if } \varepsilon(\alpha) = \varepsilon \\ \mathsf{fw}(\alpha) \cup \mathsf{fw}(\beta) \cup \mathsf{cs}(\alpha,\beta) & \text{otherwise} \end{cases} \\ \mathsf{fw}^\star(\alpha^\star) &= \mathsf{fw}^\star(\alpha), \end{aligned} \tag{14}$$

with $\mathsf{cs}(\alpha,\beta) = \mathsf{lt}(\alpha) \times \mathsf{ft}(\beta) \cup \mathsf{lt}(\beta) \times \mathsf{ft}(\alpha)$. The *Glushkov automaton* for α is $\mathcal{A}_{\mathrm{pos}}(\alpha) = (\mathrm{Pos}_0(\alpha), \Sigma, \delta_{\mathrm{pos}}, 0, F)$, with $\delta_{\mathrm{pos}} = \{\, (0, \overline{\sigma}^j, j) \mid j \in \mathsf{ft}(\alpha) \,\} \cup \{\, (i, \overline{\sigma}^j, j) \mid j \in \mathsf{fw}(\alpha, i) \,\}$ and $F = \mathsf{lt}(\alpha) \cup \{0\}$ if $\varepsilon(\alpha) = \varepsilon$, and $F = \mathsf{lt}(\alpha)$, otherwise.

The definition of lt is almost identical, differing only in the case of concatenation, where the branches are swapped. The definition of the fw function here presented deviates from the usual one in the case of the $\star$ operator. This version ensures that the unions are all disjoint. The same result can be obtained if the regular expression is first transformed into star normal form, *i.e.* such that for all subexpressions β^* one has $\varepsilon \notin \mathcal{L}(\beta)$ [12].

The number of states of $\mathcal{A}_{pos}(\alpha)$ is exactly $n + 1$ where $n = |\alpha|_\Sigma$. Thus, the average number of its states coincides with the average number of letters determined in Equation (10), *i.e.* asymptotically and on average the number of states of $\mathcal{A}_{pos}(\alpha)$ is half the size of α. On the other hand, the number of transitions in $\mathcal{A}_{pos}(\alpha)$ is, in the worst case, $n^2 + n$. Nicaud's main result in [37] is that, on average, the number of transitions is $O(|\alpha|)$. However, his computation of the number of transitions was not exact because the definition used for the fw function did not take into account the possible non-disjoint unions of its results. The version of the fw function presented above allows for an exact counting.

The generating function for the number of transitions is $T_k(z) = F_k(z) + E_k(z)$, where $F_k(z)$ and $E_k(z)$ are the generating functions associated with ft and fw, respectively. By symmetry, the generating function $L_k(z)$ for lt is the same as $F_k(z)$, which was computed by Nicaud: $L_k(z) = F_k(z) = \frac{kz}{1-z-3zR_k(z)-zR_{k,\varepsilon}(z)}$, where $R_{k,\varepsilon}(z)$ denotes the generating function for regular expressions whose languages contain ε and is given by $R_{k,\varepsilon}(z) = \frac{z+zR_k(z)}{1-2zR_k(z)}$. An estimate for the number of transitions of $\mathcal{A}_{pos}$ was given in [7] as

$$[z^n]T_k(z) \sim \frac{(1+\rho_k)(2+16\rho_k+10\rho_k^2-12\rho_k^3)}{8\rho_k\sqrt{\pi}(1-5\rho_k^2)\sqrt{2-2\rho_k}}\rho_k^{-n}n^{-\frac{1}{2}}. \tag{15}$$

Hence, an estimate for the average number of transitions *per* state is

$$\frac{[z^n]T_k(z)}{[z^n]Let_k(z)} \sim \frac{(1+\rho_k)(2+16\rho_k+10\rho_k^2-12\rho_k^3)}{(1-2\rho_k-7\rho_k^2)(1-5\rho_k^2)}. \tag{16}$$

An estimate for the average number of transitions *per* regular expression is:

$$\frac{[z^n]T_k(z)}{[z^n]R_k(z)} \sim \frac{(1+\rho_k)(1+8\rho_k+5\rho_k^2-6\rho_k^3)}{(1-\rho_k)(1-5\rho_k^2)}n. \tag{17}$$

Since ρ_k tends to 0 as k goes to ∞, it follows that for large values of k, the average number of transitions *per* state is approximately 2, while the average number of transitions *per* automaton is approximately the size of the original regular expression.

3.3 Average Size of the Partial Derivative Automata

The partial derivative automaton of a regular expression was introduced, independently and through two distinct approaches, by Mirkin [35] and Antimirov [2]. Champarnaud and Ziadi [18] proved that the two formulations are equivalent. For a RE α and a symbol $\sigma \in \Sigma$, the set of partial derivatives of α w.r.t. σ is defined inductively as follows:

$$\begin{array}{lcl} \partial_\sigma(\emptyset) & = & \partial_\sigma(\varepsilon) = \emptyset \\ \partial_\sigma(\sigma') & = & \begin{cases} \{\varepsilon\}, & \text{if } \sigma' = \sigma \\ \emptyset, & \text{otherwise} \end{cases} \end{array} \quad \begin{array}{lcl} \partial_\sigma(\alpha+\beta) & = & \partial_\sigma(\alpha) \cup \partial_\sigma(\beta) \\ \partial_\sigma(\alpha\beta) & = & \partial_\sigma(\alpha)\beta \cup \varepsilon(\alpha)\partial_\sigma(\beta) \\ \partial_\sigma(\alpha^\star) & = & \partial_\sigma(\alpha)\alpha^\star, \end{array} \tag{18}$$

where, for any $S \subseteq \mathsf{R}, \beta \in \mathsf{R}$, $S\emptyset = \emptyset S = \emptyset$, $S\varepsilon = \varepsilon S = S$, and $S\beta = \{\ \alpha\beta \mid \alpha \in S\ \}$ if $\beta \neq \emptyset$, and $\beta \neq \varepsilon$. The definition of partial derivative can be naturally extended to sets of regular expressions, words, and languages. One has $\bigcup_{\tau\in\partial_w(\alpha)}\mathcal{L}(\tau) = w^{-1}\mathcal{L}(\alpha)$, and the set of all partial derivatives of α w.r.t. words is denoted by $\mathrm{PD}(\alpha) = \bigcup_{w\in\Sigma^\star}\partial_w(\alpha)$. Note that the set $\mathrm{PD}(\alpha)$ is always finite [2], as opposed to the set of Brzozowski's derivatives, which is only finite modulo ACI.

The partial derivative automaton is defined by $\mathcal{A}_{pd}(\alpha) = (\mathrm{PD}(\alpha), \Sigma, \delta_{pd}, \alpha, F_{pd})$, where $\delta_{pd} = \{\ (\tau, \sigma, \tau') \mid \tau \in \mathrm{PD}(\alpha) \wedge \tau' \in \partial_\sigma(\tau)\ \}$, and $F_{pd} = \{\ \tau \in \mathrm{PD}(\alpha) \mid \varepsilon(\tau) = \varepsilon\ \}$. Mirkin's construction of the $\mathcal{A}_{pd}(\alpha)$ is based on solving a system of equations of the form $\alpha_i = \sigma_1\alpha_{i1} + \ldots + \sigma_k\alpha_{ik}$ if $\varepsilon(\alpha_i) = \emptyset$, and $\alpha_i = \sigma_1\alpha_{i1} + \ldots + \sigma_k\alpha_{ik} + \varepsilon$ otherwise, with $\alpha_0 \equiv \alpha$ and α_{ij}, $1 \le j \le k$, linear combinations the α_i, $0 \le i \le n$, $n \ge 0$. A solution (called the *support* of α) $\pi(\alpha) = \{\alpha_1, \ldots, \alpha_n\}$ can be obtained recursively on the structure of α as follows:

$$\begin{array}{rclrcl}
\pi(\emptyset) & = & \emptyset & \pi(\alpha\cup\beta) & = & \pi(\alpha)\cup\pi(\beta)\\
\pi(\varepsilon) & = & \emptyset & \pi(\alpha\beta) & = & \pi(\alpha)\beta\cup\pi(\beta)\\
\pi(\sigma) & = & \{\varepsilon\} & \pi(\alpha^\star) & = & \pi(\alpha)\alpha^\star.
\end{array} \tag{19}$$

Champarnaud and Ziadi [18] proved that $\mathrm{PD}(\alpha) = \pi(\alpha) \cup \{\alpha\}$ and that the two constructions lead to the same automaton. As noted by Broda et al. [7], Mirkin's algorithm to compute $\pi(\alpha)$ also provides an recursive definition of the set of transitions of $\mathcal{A}_{pd}(\alpha)$. Let $\varphi(\alpha) = \{\ (\sigma, \gamma) \mid \gamma \in \partial_\sigma(\alpha), \sigma \in \Sigma\ \}$ and $\lambda(\alpha) = \{\ \alpha' \mid \alpha' \in \pi(\alpha),\ \varepsilon(\alpha') = \varepsilon\ \}$, where both sets can be recursively defined using (18) and (19). We have, $\delta_{pd} = \{\alpha\} \times \varphi(\alpha) \cup F(\alpha)$ where the result of the $\times$ operation is seen as a set of triples and the set F is defined inductively by:

$$\begin{array}{rcl}
F(\emptyset) & = & F(\varepsilon) = F(\sigma) = \emptyset,\ \sigma\in\Sigma\\
F(\alpha+\beta) & = & F(\alpha)\cup F(\beta)\\
F(\alpha\beta) & = & F(\alpha)\beta\cup F(\beta)\cup\lambda(\alpha)\beta\times\varphi(\beta)\\
F(\alpha^\star) & = & F(\alpha)\alpha^\star\cup(\lambda(\alpha)\times\varphi(\alpha))\alpha^\star.
\end{array} \tag{20}$$

For all $\alpha \in \mathsf{R}$, $\mathcal{A}_{pd}(\alpha) = (\pi(\alpha) \cup \{\alpha\}, \Sigma, \{\alpha\} \times \varphi(\alpha) \cup F(\alpha), \alpha, \lambda(\alpha) \cup \varepsilon(\alpha)\{\alpha\})$.

In his original paper, Mirkin showed that $|\pi(\alpha)| \le |\alpha|_\Sigma$. Since $\mathcal{A}_{pd}(\alpha)$ is a quotient of the Glushkov automaton [17], we know that it has at most $|\alpha|_\Sigma+1$ states. But this upper bound is reached if and only if, at every step during the computation of $\pi(\alpha)$, all unions are disjoint. There are however two cases in which this clearly does not happen. Whenever $\varepsilon \in \pi(\beta) \cap \pi(\gamma)$, $|\pi(\beta + \gamma)| = |\pi(\beta) \cup \pi(\gamma)| \le |\pi(\beta)| + |\pi(\gamma)| - 1$ and also $|\pi(\beta\gamma^\star)| = |\pi(\beta)\gamma^\star \cup \pi(\gamma^\star)| = |\pi(\beta)\gamma^\star \cup \pi(\gamma)\gamma^\star| \le |\pi(\beta)| + |\pi(\gamma)| - 1$. These observations lead to the computation of a lower bound of the number of state mergings [6]. The respective generating function is $I_k(z) = \frac{(z+z^2)R_{\pi,k}(z)^2}{\sqrt{\Delta_k(z)}}$, where $R_{\pi,k}(z)$ is the generating function of $\alpha \in \mathsf{RE}$ such that $\varepsilon \in \pi(\alpha)$. The asymptotic estimate of the (cumulative) number of mergings is

$$[z^n]I_k(z) \sim \frac{1+\rho_k}{64}\frac{\left(a_k(\rho_k) + b(\rho_k)^2 - 2b(\rho_k)\sqrt{a_k(\rho_k)}\right)}{\sqrt{\pi}\sqrt{2-2\rho_k}}\rho_k^{-(n+1)}n^{-1/2}, \tag{21}$$

where now $a_k(z) = 16z^4 - 24z^3 + (64k + 1)z^2 + 6z + 1$ and $b(z) = -4z^2 + 3z + 1$.

From (2) and (21) one easily gets the following asymptotic estimate for the average number of mergings

$$\frac{[z^n]I_k(z)}{[z^n]R_k(z)} \sim \lambda_k\, n, \tag{22}$$

where $\lambda_k = \frac{(1+\rho_k)}{16(1-\rho_k)}\left(a_k(\rho_k) + b(\rho_k)^2 - 2b(\rho_k)\sqrt{a_k(\rho_k)}\right)$. Using again the fact that $\lim\limits_{k\to\infty} \rho_k = 0$, and that $\lim\limits_{k\to\infty} a_k(\rho_k) = 9$, while $\lim\limits_{k\to\infty} b(\rho_k) = 1$, one gets that $\lim_{k\to\infty} \lambda_k = \frac{1}{4}$. This means that, for a RE of size n, the average number of state mergings is, asymptotically, about $\frac{n}{4}$.

In order to obtain a lower bound for the reduction in the number of states of the $\mathcal{A}_{pd}$ automaton, as compared to the ones of the $\mathcal{A}_{pos}$ automaton, it is enough to compare the number of mergings for an expression α with the number of letters in α. From (9) and (22) one gets

$$\frac{[z^n]I_k(z)}{[z^n]L_k(z)} \sim \frac{1-\rho_k}{4k\rho_k^2}\lambda_k. \tag{23}$$

It is easy to see that $\lim_{k\to\infty} \frac{1-\rho_k}{4k\rho_k^2}\lambda_k = \frac{1}{2}$. In other words, asymptotically, the average number of states of the $\mathcal{A}_{pd}$ automaton is about one half of the number of states of the $\mathcal{A}_{pos}$ automaton, and about one quarter of the size of the corresponding RE.

In [7] the same technique was used for the estimation of the number of transitions of $\mathcal{A}_{pd}$. Observing the equations (20), in this case one first estimates the number of mergings that occur in $\lambda(\alpha)$ and then the corresponding number of transition mergings. Letting It_z being the generating function for the number of transitions mergings, one has

$$[z^n]It_k(z) \sim \frac{(1+\rho_k)\left(a(\rho_k)\sqrt{b(\rho_k)} + c(\rho_k)\right)}{16\sqrt{\pi}\rho_k\sqrt{2-2\rho_k}\,(1-5\rho_k^2)d(\rho_k)}\rho_k^{-n}n^{-\frac{1}{2}} \tag{24}$$

where $a(z)$, $b(z)$, $c(z)$, and $d(z)$ are some fixed polynomials. Therefore, a lower bound for the average number of mergings per transition of the Glushkov automaton is given by

$$[z^n]\frac{It_k(z)}{T_k(z)} \sim \frac{a(\rho_k)\sqrt{b(\rho_k)} + c(\rho_k)}{4(1 + 8\rho_k + 5\rho_k^2 - 6\rho_k^3)d(\rho_k)}. \tag{25}$$

Because $lim_{k\to\infty}[z^n]\frac{It_k(z)}{T_k(z)} = \frac{1}{2}$, asymptotically with respect to k, the number of transitions in $\mathcal{A}_{pd}$ is at most half the number of transitions in $\mathcal{A}_{pos}$.

3.4 Other NFAs and Open Problems

Another well-known quotient of the Glushkov automaton is the *follow automaton*, $\mathcal{A}_f$, which can also be obtained by eliminating the ε-transitions from the $\mathcal{A}_{\varepsilon-fol}$ [26]. It is known that if the regular expression is in star normal form, the $\mathcal{A}_{pd}$ is always smaller than or equal to $\mathcal{A}_f$. The conversion to star normal form is linear and experimental results suggest that, on average, regular expressions are in that form [22]. Although it is an open problem to theoretically show that this is the case, we believe that on average the $\mathcal{A}_f$ is not smaller than $\mathcal{A}_{pd}$. Experimental results also suggest that on average the $\mathcal{A}_{pd}$ almost coincides with the bisimilarity of $\mathcal{A}_{pos}$, $\mathcal{A}_{pos}/_{\equiv_b}$. Maia et al. [31] characterised, for finite languages, the graph properties of $\mathcal{A}_{pd}$, and determined under which conditions $\mathcal{A}_{pd} \simeq \mathcal{A}_{pos}/_{\equiv_b}$. It is an open problem to obtain an asymptotic average behaviour of these two constructions.

There are some related constructions of automata from regular expressions. Instead of left quotients one can consider right quotients. Given a language L, the *right-quotient* of L w.r.t. a word w is the language $Lw^{-1} = \{\ x \mid xw \in L\ \}$. It is not difficult to verify that $Lw^{-1} = (w^R)^{-1}L^R$, where $(\ )^R$ represents the reversal operation. Then sets of right-partial derivatives and the *right-partial derivative automaton* ($\overleftarrow{\mathcal{A}}_{pd}$) can be naturally defined. However, as $(\mathcal{A}_{pd}(\alpha^R))^R \simeq \overleftarrow{\mathcal{A}}_{pd}$, we can conclude that the number of states of $\overleftarrow{\mathcal{A}}_{pd}$ are, asymptoticaly and on average, half of the number of states of $\mathcal{A}_{pos}$. Yamamoto [48] introduced the *prefix automaton* of a RE, $\mathcal{A}_{pre}$, as a quotient of the Thompson automaton that corresponds also to the quotient of the Glushkov automaton by a right-invariant relation that identifies states of $\mathcal{A}_{pos}$ with the same left language. Maia et al. [32] characterised the $\mathcal{A}_{pre}$ automaton as a solution of a system of equations, and presented a recursive definition of $\mathcal{A}_{pre}$ akin to the one given for $\mathcal{A}_{pd}$ in equations (19) and (20). Using the framework of analytic combinatorics and techniques similar to the ones described in Section 3.3, it was shown that, as the size of alphabet grows, the average number of states of the $\mathcal{A}_{pre}$ automaton approaches the number of states of the $\mathcal{A}_{pos}$ automaton.

Finally, given the set of positions of a RE α, $\mathrm{Pos}(\alpha)$, instead of considering the fw function as in the $\mathcal{A}_{pos}$ automaton, one can consider the pr function, that gives the set of positions that can precede a given position *i.e.* $\mathsf{pr}(\alpha, j) = \{\ i \mid ua_ia_jv \in \mathcal{L}(\overline{\alpha})\ \}$, for $j \in \mathrm{Pos}(\alpha)$. The *previous automaton* $\mathcal{A}_{pre}$ is defined akin to $\mathcal{A}_{pos}$, but considering a unique distinct final state. It follows that $\mathcal{A}_{prev}(\alpha) \simeq (\mathcal{A}_{pos}(\alpha^R))^R$. Thus, the number of states coincides with the number of states of $\mathcal{A}_{pos}$. The interest in this construction relies in its connection to the recent *au point* DFA construction [4, 39] that we will mention in Section 5.

4 Extended Regular Expressions to NFAs

Although regular languages are trivially closed for boolean operations, the manipulation of intersection and complementation with regular expressions or nondeterministic finite automata is non-trivial and leads to an exponential blow up. However, there are several applications where extended regular expressions are used to represent information and it is important to study their conversion to automata. Caron *et al.* [14] extended the notion of partial derivatives and partial derivative automaton to regular expressions with intersection and complementation.

Broda et al. [11] extended the same notions to regular expressions with shuffle and studied the average number of states of the corresponding partial derivative automaton. The complexity of the shuffle (or interleaving) operation is well studied in the worst case. Mayer and Stockmeyer [33] showed that for REs with shuffle, of size n, an equivalent NFA needs at most 2^n states, and presented a family of REs with shuffle, of size $O(n)$, for which the corresponding NFAs have at least 2^n states. Gelade [20], and Gruber and Holzer [24] showed that there exists a double exponential trade-off in the translation from REs with shuffle to standard REs. Gelade also gave a tight double exponential upper bound for the translation of REs with shuffle to DFAs.

Broda et al. showed that the number of states of the partial derivative automata is in the worst case at most 2^m, where m is the number of letters in the expression, while asymptotically and on average it is no more than $(\frac{4}{3})^m$. Considering the grammar for regular expressions (1) with one more rule for shuffle $\alpha \sqcup\!\sqcup \alpha$, we can extend the definition of the support π in (19) by:

$$\pi(\alpha \sqcup\!\sqcup \beta) \quad = \quad \pi(\alpha) \sqcup\!\sqcup \pi(\beta) \cup \pi(\alpha) \sqcup\!\sqcup \{\beta\} \cup \{\alpha\} \sqcup\!\sqcup \pi(\beta), \tag{26}$$

where for $S, T \subseteq \mathsf{R}$, $S \sqcup\!\sqcup T = \{\ \alpha \sqcup\!\sqcup \beta \mid \alpha \in S, \beta \in T\ \}$, and $\{\varepsilon\} \sqcup\!\sqcup S = S \sqcup\!\sqcup \{\varepsilon\} = S$. With this expression it is easy to see that now $|\pi(\alpha)| \leq 2^{|\alpha_\Sigma|}$ and this bound is reachable for the family of regular expressions $\alpha_n = a_1 \sqcup\!\sqcup \cdots \sqcup\!\sqcup a_n$, where $n \geq 1$, $a_i \neq a_j$ for $1 \leq i \neq j \leq n$. Let $P_k(z)$ be the generating function for an upper bound for the number of elements in π. For expressions of size n, one has,

$$[z^n]P_k(z) \quad \sim \quad \frac{-(3+3k)^{\frac{1}{4}}\rho_k^{-n-\frac{1}{2}} + (3+4k)^{\frac{1}{4}}(\rho_k')^{-n-\frac{1}{2}}}{2\sqrt{\pi}}(n+1)^{-\frac{3}{2}},$$

where now $\rho_k = \frac{-1+2\sqrt{3+3k}}{11+12k}$ and $\rho_k' = \frac{-1+2\sqrt{3+4k}}{11+16k}$.

For a regular expression α of size n, let avP and avL be the average size of π and the average alphabetic size, respectively. Taking into account the worst case upper bound, we have compared the values of $\log_2 avP$ and avL, obtaining

$$\lim_{n,k\to\infty} \frac{\log_2 avP}{avL} \quad = \quad \log_2 \frac{4}{3} \sim 0.415.$$

Therefore, one has the following significant improvement, when compared with the worst case, for the average-case upper bound: for large values of k and n an upper bound for the average number of states of $\mathcal{A}_{pd}$ is $(\frac{4}{3} + o(1))^{|\alpha|_\Sigma}$.

Regular expressions with intersection have a worst-case behaviour similar to shuffle [20, 24]. The definition of the support π in this case is just $\pi(\alpha \cap \beta) = \pi(\alpha) \cap \pi(\beta)$ (with the $\cap$ for sets of REs defined as above for $\sqcup\!\sqcup$, only interchanging the operators). However, the analytic analysis of the correspondent generating function is much harder and an estimation of an upper bound of the average size is an on-going work by the authors of this paper. Despite that, experimental results suggest that the average number of states in $\mathcal{A}_{pd}$ automata is much smaller than $2^{|\alpha|_\Sigma}$.

We note that for both intersection and complementation is not clear how to extend the position based constructions.

5 Regular Expressions to DFAs

A word derivative w.r.t. a RE α, $w^{-1}\alpha$ is such that $\mathcal{L}(w^{-1}\alpha) = w^{-1}\mathcal{L}(\alpha)$. The set of word derivatives $D(\alpha)$ is finite modulo the ACI axioms. The Brzozowski derivative automaton can be defined by: $A_{\mathcal{B}}(\alpha) = (\mathcal{D}(\alpha), \Sigma, \delta, [\alpha], F)$, where $F = \{\ [d] \in \mathcal{D}(\alpha) \mid \varepsilon(d) = \varepsilon\ \}$, and $\delta([q], \sigma) = [\sigma^{-1}q]$, for all $[q] \in \mathcal{D}(\alpha), \sigma \in \Sigma$. McNaughton and Yamada [34] presented a DFA construction from an RE that coincides with the determinization of the $\mathcal{A}_{pos}$ automaton. Recently, Asperti [4] introduced a DFA construction from an RE that uses *pointed* REs α' where some letters are annotated with a point and $\mathcal{L}(\alpha')$ is the set words that start at some *pointed* letter. All these constructions lead to DFAs that have a size that is, in the worst case, exponential in the size of the initial RE. For all these constructions, no average-case complexity results are known, as far as the authors are aware of. Some experimental results suggest that *au point* is on average smaller then the other constructions. To understand why this happens and to find estimates of the average size of each construction is thus an open problem.

6 KATs to NTAs

Kleene algebra with tests (KAT) [28] is a decidable equational system combining Kleene and Boolean algebras, and is specially suited to capture and verify properties of simple imperative programs. The equational theory of KAT is PSPACE-complete. The decidability, conciseness and expressiveness of KAT motivated its recent automatisation within several theorem provers [40, 41, 3] and functional languages [1, 42]. Most of those implementations use (variants of) the coalgebraic automaton on guarded strings developed by Kozen [30]. In that approach, derivatives are considered over symbols of the from $\mathsf{v}p$, where p is an alphabetic *program* symbol and v a valuation of boolean variables (the *guard*, normally called atom). This induces an exponential blow-up on the number of states or transitions of the automata. This exponential growth was avoided in Silva [45], and in Broda et al. [8, 10], by using for KAT standard finite automata, where transitions are labeled both with program symbols and boolean tests (instead of atoms).

The abstract syntax of KAT expressions, over an alphabet $\mathsf{P} = \{p_1, \ldots, p_k\}$ of program symbols and $\mathsf{T} = \{t_0, \ldots, t_{l-1}\}$ of boolean variables (tests), can be given by the following unambiguous grammar, suitable for applying the symbolic method.

$$\begin{aligned} \mathsf{BExp} : b &\rightarrow 0 \mid 1 \mid t \mid \neg b \mid (b+b) \mid (b \cdot b) && (27)\\ \mathsf{AExp} : a &\rightarrow p \mid (a+a) \mid (a+b) \mid (b+a) \mid (a \cdot a) \mid (a \cdot b) \mid (b \cdot a) \mid a^\star && (28)\\ \mathsf{Exp} : e &\rightarrow b \mid a. && (29) \end{aligned}$$

Here BExp, AExp, and Exp represent sets of boolean expressions, KAT expressions with at least one program symbol $p \in \mathsf{P}$, and KAT expressions, respectively. For simplicity, the grammar excludes subexpressions of the form $b^\star$, as their semantics correspond to the set of all boolean assignments and thus are equivalent to 1. For the negation of test symbols we use $\bar{t}$ instead of $\neg t$. The set At, of *atoms* over T, is the set of all boolean assignments to all elements of T, $\mathsf{At} = \{\ x_0 \cdots x_{l-1} \mid x_i \in \{t_i, \bar{t}_i\},\ t_i \in \mathsf{T}\ \}$. Elements of At are denoted by v, and we write $\mathsf{v} \leq b$, if $\mathsf{v} \rightarrow b$ is a boolean tautology.

The set of *guarded strings* over P and T is $\mathsf{GS} = (\mathsf{At} \cdot \mathsf{P})^\star \cdot \mathsf{At}$. Regular sets of guarded strings form the standard language-theoretic model for KAT [29]. A *(nondeterministic) automaton with tests* (NTA) over the alphabets P and T is a tuple $\mathcal{A} = \langle Q, q_0, o, \delta \rangle$, where Q is a finite set of states, $q_0 \in Q$ is the initial state, $o : Q \rightarrow \mathsf{BExp}$ is the output function, and $\delta \subseteq Q \times (\mathsf{BExp} \times \mathsf{P}) \times Q$ is the transition relation. A guarded string $\mathsf{v}_0\sigma_1 \ldots \sigma_n\mathsf{v}_n$, with $n \geq 0$, is accepted by the automaton $\mathcal{A}$ if and only if there is a sequence of states $q_0, q_1, \ldots, q_n \in Q$, where q_0 is the initial state, and, for $i = 0, \ldots, n-1$, one has $\mathsf{v}_i \leq b_i$ for some $(q_i, (b_i, \sigma_{i+1}), q_{i+1}) \in \delta$, and $\mathsf{v}_n \leq o(q_n)$.

Silva [45] presented the Glushkov construction for KAT, and Broda et al. [8, 10] defined the partial derivative automaton for KAT. The asymptotic average size of both constructions were studied in [8]. It was shown that, contrary to other automata constructions for KAT expressions, they enjoy the same descriptional complexity behaviour as their counterparts for regular expressions.

Consider the generating function

$$R_m(z) = \frac{1 - z - \sqrt{\Delta_m(z)}}{4z}, \text{ where } \Delta_m(z) = 1 - 2z - (15 + 8m)z^2, \tag{30}$$

for the number of regular expressions generated by the grammar in (1), including $\emptyset$, which is almost identical to the one in Subsection 2.2. It is easy to see that $B_l(z) = R_l(z)$ and $E_{k,l}(z) = R_{k+l}(z)$, where l and k are respectively the sizes of P and T, and $B_l(z)$ and $E_{k,l}(z)$ respectively the generating functions for BExp and Exp. Using the technique presented in Section 2 applied to (30), the asymptotic estimates for the number of regular expressions of size m is

$$[z^n]R_m(z) \sim \frac{\sqrt{\rho_m}\,\sqrt[4]{2m+4}}{4\sqrt{\pi}}\,\rho_m^{-(m+1)}(m+1)^{-\frac{3}{2}}, \tag{31}$$

where $\rho_m = \frac{-1+2\sqrt{2m+4}}{15+8m}$ is the radius of convergence of $R_m(z)$. Let $P_{k,l}(z)$ denote the generating function for the number of program symbols in KAT expressions. Then, we have $P_{k,l} = \frac{k}{k+l}Let_{k+l}(z)$, with $Let_m(z)$ as in (8). Therefore, the probability, for a uniform distribution, that a symbol in a KAT expression of size n is a program symbol is

$$\frac{[z^n]P_{k,l}(z)}{n\,[z^n]E_{k,l}(z)} \sim \frac{\left(4(k+l)+8-\sqrt{2(k+l)+4}\right)k}{(15+8\,(k+l))\,(k+l+2)}\left(1+\frac{1}{n}\right)^{3/2} = \eta_{k,l,n}. \tag{32}$$

The average number of program symbols, as $k+l$ increases, tends to $\frac{1}{2(c+1)}$, where $c = \frac{l}{k}$. For instance, if $l = k$, $l = 2k$, and $l = \frac{1}{2}k$, this limit is, respectively, $\frac{1}{4}$, $\frac{1}{6}$, and $\frac{1}{3}$. Furthermore, for any ratio c, the asymptotic average number of states in Glushkov automata is less than half the size of the corresponding expressions.

Since for KAT the recursive definition of the support π just differs by adding $\pi(b) = \emptyset$, which does not affect the computations, one can apply the method used in Subsection 3.3 in order to get an upper bound for the state complexity of the partial derivative automaton. One obtains,

$$\frac{[z^n]I_{k,l}(z)}{[z^n]E_{k,l}(z)} \sim \lambda_{k,l}\,n, \tag{33}$$

where $\lambda_{k,l} = \frac{1+\rho_{k+l}}{16(1-\rho_{k+l})}\left(a_k(\rho_{k+l}) + b(\rho_{k+l})^2 - 2b(\rho_{k+l})\sqrt{a_k(\rho_{k+l})}\right)$, with $a_k(z) = 16z^4 - 24z^3 + (64k+1)z^2 + 6z + 1$, and $b(z) = -4z^2 + 3z + 1$. Therefore

$$\frac{[z^n]I_{k,l}(z)}{[z^n]P_{k,l}(z)} \sim \frac{\lambda_{k,l}}{\eta_{k,l,n}}. \tag{34}$$

One can see that, for a fixed value of l this ratio approaches $\frac{1}{2}$, as k grows. This means that the number of states in the equation automaton is asymptotically, and on average, half the number of states in the Glushkov automaton.

It is more difficult to obtain a sufficiently accurate upper bound for the average number of transitions in the Glushkov NTAs. In particular, several grammars for expressions with different properties, such as for KAT expressions that have no atom $\mathsf{v} \in \mathsf{At}$ in their associated language, have to be considered. In this case the computations no longer mirror the ones for regular expressions, but nevertheless the same result is reached: asymptotically, and on average, the number of transitions of the Glushkov automaton is linear in the size of the KAT expression. To estimate the average number of transitions in the $\mathcal{A}_{pd}$ for KAT expressions is an open problem.

6.1 SKA and SKAT

Synchronous Kleene algebra (SKA), introduced by Prisacariu [43], combines KA with a synchrony model of concurrency. Synchronous here means that two concurrent processes execute a single action simultaneously at each time instant of a unique global clock.

A SKA over a finite set $\mathsf{A_B}$ is given by a structure $(\mathcal{A}, +, \cdot, \times, ^*, 0, 1, \mathsf{A_B})$, where $\mathsf{A_B} \subseteq \mathcal{A}$, $(\mathcal{A}, +, \cdot, ^*, 0, 1)$ is a Kleene algebra, and $\times$ is a binary operator that is associative, commutative, distributive over $+$, with absorvent element 0 and identity 1. Furthermore, it satisfies $a \times a = a$, $\forall a \in \mathsf{A_B}$, as well as the following equation for synchrony: $(\alpha^\times \cdot \alpha) \times (\beta^\times \cdot \beta) = (\alpha^\times \times \beta^\times) \cdot (\alpha \times \beta)$, where $\alpha^\times$ and $\beta^\times$ are of the form $a_1 \times \cdots \times a_n$, for $a_i \in \mathsf{A_B}$.

Broda et al. [5] defined the partial derivative automaton $\mathcal{A}_{pd}$ for SKA. It was shown that the worst-case upper bound for the number of states of this automaton coincides with the one for the $\mathcal{A}_{pd}$ for regular expressions with shuffle. This implies the same upper bound for the average number of states of the $\mathcal{A}_{pd}$ for SKA. Prisacariu also generalised Kleene algebra with tests to the synchronous setting (SKAT). Broda et al. extended NTA's for SKAT, as well as the derivative based methods already developed for SKA. Also in this case, experimental results suggest that on average the size of $\mathcal{A}_{pd}$ for both SKA and SKAT are much smaller than the worst-case upper bound. Thus, a more fine-grained study of the average-case complexity of these automata in the analytic combinatorics framework is worthwhile.

7 Conclusions

We presented recent results on the average size of automata obtained from regular expressions and some extended expressions. The framework of analytic combinatorics was the main tool for estimating the asymptotic number of states and of transitions for automata, as a function of the expressions' size. In general it is necessary to obtain generating functions associated with the measures under consideration and then to be able to estimate the asymptotic behaviour of their coefficients.

Both tasks may turn out to be very hard, and small differences on recursive definitions can lead to functions with completely different analytic behaviours.

We finish by pointing out some future directions of work. Concerning ε-NFAs, there are other conversions from REs to automata with better worst-case complexity. In particular, Gruber and Gulan's construction is optimal w.r.t. the alphabetic size of regular expressions [23, 25]. This construction corresponds to applying the ε-follow construction to the star normal form of the initial expression. Thus, if one is able to estimate the asymptotic number of expressions in star normal form of a given size, one can proceed as in Section 3.1 and obtain the average size of this construction.

Concerning DFAs, the main ingredient is to tackle the subset construction, i.e. determinization, within the analytic framework. Any progress in this direction will allow to obtain estimates for the average-case complexity of the RE to DFA conversions.

Nicaud et al. [38] studied the average number of transitions of Glushkov automata under the non-uniform distribution, inspired by random binary search trees (BST-like). With this distribution, the average number of transitions of the Glushkov automaton is quadratic with respect to the size of the regular expression. We believe that the same result is valid for the partial derivative automaton. However, we think that one needs a better understanding of the relevance of the different distributions for REs before this approach is applied to all the previous described constructions.

References

[1] Ricardo Almeida. Decision Algorithms for Kleene Algebra with Tests and Hoare Logic. Master's thesis, Faculdade de Ciências da Universidade do Porto, 2012.

[2] Valentin M. Antimirov. Partial Derivatives of Regular Expressions and Finite Automaton Constructions. *Theoretical Computer Science*, 155(2):291–319, 1996.

[3] Alasdair Armstrong, Georg Struth, and Tjark Weber. Program Analysis and Verification Based on Kleene Algebra in Isabelle/HOL. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *4th Inter. Conference ITP 2013, Rennes, France. Proceedings*, volume 7998 of *Lecture Notes on Computer Science*, pages 197–212. Springer, 2013.

[4] Andrea Asperti, Claudio Sacerdoti Coen, and Enrico Tassi. Regular Expressions, au point. *CoRR*, abs/1010.2604, 2010.

[5] Sabine Broda, Sílvia Cavadas, Miguel Ferreira, and Nelma Moreira. Derivative Based Methods for Deciding SKA and SKAT. Submitted.

[6] Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. On the Average State Complexity of Partial Derivative Automata: an Analytic Combinatorics Approach. *International Journal of Foundations of Computer Science*, 22(7):1593–1606, 2011.

[7] Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. On the Average Size of Glushkov and Partial Derivative Automata. *International Journal of Foundations of Computer Science*, 23(5):969–984, 2012.

[8] Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. On the Average Size of Glushkov and Equation Automata for KAT Expressions. In *19th Inter. Symposium on Fundamentals of Computation Theory*, volume 8070 of *Lecture Notes on Computer Science*, pages 72–83. Springer, 2013.

[9] Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. A Hitchhiker's Guide to Descriptional Complexity through Analytic Combinatorics. *Theoretical Computer Science*, 528:85–100, 2014.

[10] Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. On the Equivalence of Automata for KAT-expressions. In Arnold Beckmann, Erzsébet Csuhaj-Varjú, and Klaus Meer, editors, *Language, Life, Limits - 10th Conference on Computability in Europe, CiE 2014, Budapest, Hungary, June 23-27, 2014. Proceedings*, volume 8493 of *Lecture Notes on Computer Science*, pages 73–83. Springer, 2014.

[11] Sabine Broda, António Machiavelo, Nelma Moreira, and Rogério Reis. Partial Derivative Automaton for Regular Expressions with Shuffle. In Jeffrey Shallit and Alexander Okhotin, editors, *Proceedings of the 17th Int. Workshop on Descriptional Complexity of Formal Systems (DCFS15)*. Springer, 2015.

[12] Anne Brüggemann-Klein. Regular Expressions into Finite Automata. *Theoretical Computer Science*, 48:197–213, 1993.

[13] John Brzozowski. Derivatives of Regular Expressions. *J. Association for Computer Machinery*, 11(4):481–494, 1964.

[14] Pascal Caron, Jean-Marc Champarnaud, and Ludovic Mignot. Partial Derivatives of an Extended Regular Expression. In Adrian Horia Dediu, Shunsuke Inenaga, and Carlos Martín-Vide, editors, *Language and Automata Theory and Applications — 5th International Conference, LATA 2011, Tarragona, Spain, May 26-31, 2011. Proceedings*, volume 6638 of *Lecture Notes in Computer Science*, pages 179–191. Springer, 2011.

[15] Jean-Marc Champarnaud, Faissal Ouardi, and Djelloul Ziadi. Follow Automaton versus Equation Automaton. In Lucian Ilie and Detlef Wotschke, editors, *DCFS*, volume Report No. 619, pages 145–153. Department of Computer Science, The University of Western Ontario, Canada, 2004.

[16] Jean-Marc Champarnaud and Djelloul Ziadi. Computing the Equation Automaton of a Regular Expression in Space and Time. In Amihood Amir and Gad M. Landau,

editors, *CPM*, volume 2089 of *Lecture Notes in Computer Science*, pages 157–168. Springer, 2001.

[17] Jean-Marc Champarnaud and Djelloul Ziadi. Canonical Derivatives, Partial Derivatives and Finite Automaton Constructions. *Theoretical Computer Science*, 289:137–163, 2002.

[18] Jean-Marc Champarnaud and Djelloull Ziadi. From Mirkin's Prebases to Antimirov's Word Partial Derivatives. *Fundamenta Informaticae*, 45(3):195–205, 2001.

[19] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[20] Wouter Gelade. Succinctness of Regular Expressions with Interleaving, Intersection and Counting. *Theoretical Computer Science*, 411(31-33):2987–2998, 2010.

[21] V. M. Glushkov. The Abstract Theory of Automata. *Russian Math. Surveys*, 16(5):1–53, 1961.

[22] Hugo Gouveia, Nelma Moreira, and Rogério Reis. Small NFAs from Regular Expressions: Some Experimental Results. In Fernando Ferreira, Hélia Guerra, Elvira Mayordomo, and João Rasga, editors, *Proceedings of 6th Conference on Computability in Europe (CIE 2010)*, pages 194–203, Ponta Delgada, Azores, Portugal, June/July 2010. CMATI.

[23] Hermann Gruber and Stefan Gulan. Simplifying Regular Expressions. In Adrian Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *4th International Conference on Language and Automata Theory and Applications, LATA 2010. Proceedings*, volume 6031 of *Lecture Notes on Computer Science*, pages 285–296, Trier, Germany, 05 2010. Springer.

[24] Hermann Gruber and Markus Holzer. Finite Automata, Digraph Connectivity, and Regular Expression Size. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *35th ICALP*, volume 5126 of *Lecture Notes on Computer Science*, pages 39–50. Springer, 2008.

[25] Hermann Gruber and Markus Holzer. From Finite Automata to Regular Expressions and Back-A Summary on Descriptional Complexity. In Zoltán Ésik and Zoltán Fülöp, editors, *Proceedings 14th International Conference on Automata and Formal Languages, AFL 2014, Szeged, Hungary, May 27-29, 2014.*, volume 151 of *EPTCS*, pages 25–48, 2014.

[26] Lucien Ilie and Sheng Yu. Follow Automata. *Information and Computation*, 186(1):140–162, 2003.

[27] Dexter Kozen. *Automata and Computability*. Springer, 1997.

[28] Dexter Kozen. Kleene Algebra with Tests. *Trans. on Prog. Lang. and Systems*, 19(3):427–443, 05 1997.

[29] Dexter Kozen. Automata on Guarded Strings and Applications. *Matématica Contemporânea*, 24:117–139, 2003.

[30] Dexter Kozen. On the Coalgebraic Theory of Kleene Algebra with Tests. Computing and Information Science Technical Reports `http://hdl.handle.net/1813/10173`, Cornell University, May 2008.

[31] Eva Maia, Nelma Moreira, and Rogério Reis. Partial Derivative and Position Bisimilarity Automata. In Markus Holzer and Martin Kutrib, editors, *Implementation and Application of Automata, 19th International Conference (CIAA 2014)*, volume 8587 of *Lecture Notes on Computer Science*, pages 264–277. Springer, 2014.

[32] Eva Maia, Nelma Moreira, and Rogério Reis. Prefix and Right-Partial Derivative Automata. In Mariya Soskova and Victor Mitrana, editors, *Computability in Europe (CiE 2015)*, number 9136 in Theoretical Computer Science and General Issues, pages 1–10. Springer, 2015.

[33] Alain J. Mayer and Larry J. Stockmeyer. Word Problems—This Time with Interleaving. *Information and Computation*, 115(2):293–311, 1994.

[34] R. McNaughton and H. Yamada. Regular Expressions and State Graphs for Automata. *IEEE Transactions on Electronic Computers*, 9:39–47, 1960.

[35] B. G. Mirkin. An Algorithm for Constructing a Base in a Language of Regular Expressions. *Engineering Cybernetics*, 5:51–57, 1966.

[36] Cyril Nicaud. *Étude du comportement en moyenne des automates finis et des langages rationnels*. PhD thesis, Université de Paris 7, 2000.

[37] Cyril Nicaud. On the Average Size of Glushkov's Automata. In Adrian Horia Dediu, Armand-Mihai Ionescu, and Carlos Martín-Vide, editors, *Proc. 3rd LATA*, volume 5457 of *Lecture Notes on Computer Science*, pages 626–637. Springer, 2009.

[38] Cyril Nicaud, Carine Pivoteau, and Benoît Razet. Average Analysis of Glushkov Automata under a BST-Like Model. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2010, December 15-18, 2010, Chennai, India*, pages 388–399, 2010.

[39] Tobias Nipkow and Dmitriy Traytel. Unified Decision Procedures for Regular Expression Equivalence. *Archive of Formal Proofs*, 2014, 2014.

[40] David Pereira. *Towards Certified Program Logics for the Verification of Imperative Programs*. PhD thesis, University of Porto, 2013.

[41] Damien Pous. Kleene Algebra with Tests and Coq Tools for while Programs. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving — 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 180–196. Springer, 2013.

[42] Damien Pous. Symbolic Algorithms for Language Equivalence and Kleene Algebra with Tests. In Sriram K. Rajamani and David Walker, editors, *42nd POPL 2015*, pages 357–368. ACM, 2015.

[43] Cristian Prisacariu. Synchronous Kleene algebra. *J. Log. Algebr. Program.*, 79(7):608–635, 2010.

[44] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009.

[45] Alexandra Silva. Position Automata for Kleene Algebra with Tests. *Sci. Ann. Comp. Sci.*, 22(2):367–394, 2012.

[46] Seppo Sippu and Eljas Soisalon-Soininen. *Parsing Theory*, volume II: LR(k) and LL(k) Parsing of *EATCS Monographs on Theoretical Computer Science*. Springer, 1990.

[47] K. Thompson. Regular Expression Search Algorithm. *Communications of the ACM*, 11(6):410–422, 1968.

[48] Hiroaki Yamamoto. A New Finite Automaton Construction for Regular Expressions. In Suna Bensch, Rudolf Freund, and Friedrich Otto, editors, *Sixth Workshop on Non-Classical Models for Automata and Applications - NCMA 2014, Kassel, Germany, July 28-29, 2014. Proceedings*, volume 304 of *books@ocg.at*, pages 249–264. Österreichische Computer Gesellschaft, 2014.

The Logic in Computer Science Column

by

Yuri Gurevich

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

SELECTED PAPERS FROM THE 1^{st} WORKSHOP "LOGIC, LANGUAGE, AND INFORMATION"

Guido Sciavicco
Dept. of Information and Communication Engineering
University of Murcia, Spain
guido@um.es
Alfredo Burrieza
Dept. of Phylosophy
university of Málaga, Spain
burrieza@uma.es

This issue's Logic Column in Computer Science features five selected papers from the 1^{st} Workshop "Logic, Language, and Information", held in Málaga (Spain) from the November 3^{rd} to November 5^{th}, 2014, and co-edited by Prof. Guido Sciavicco (University of Murcia) and Prof. Alfredo Burrieza (University of Málaga).

The workshop was held as the foundational act of the recently created Logic, Language and Information Research Unit (UILLI-AT), which features research groups from the University of Málaga and Sevilla, under the control of the institute Andalucia Tech. External researchers are also involved in the unit, and the basic areas of interests are logic, linguistics, computer science, and cognitive science, particularly focused on information management and representation. There were 16 contributions and 3 invited talks at the workshop.

Although the contributions to the workshop were focused on very different areas, the common denominator to all of them was logic. Among the areas that have been discussed during this event, we mention: formal analysis of concepts, information extraction models for metabolic networks, interval temporal logics, preference change logics in the context of social networks, automatic proof systems in databases, formal methods for analysis of spoken language, epistemic logics for collective awareness, bio-informatics analysis methods, transformation of programs, belief revision methods, abductive processes, and mereology in the context of temporal reasoning.

The selected papers, that underwent a full review process, are the following ones. *Distributed Explicit Knowledge and Collective Awareness*, by Alfredo Burrieza and Claudia Fernández: in this paper the authors present an approach to model the communication among a group of agents with limited knowledge resources; this leads to the introduction of the concept of collective awareness, which allows one to transform implicit distrbuited knowledge into explicit one. *A Logical Approach for Direct-Optimal Basis of Implications*, by Estrella Rodríguez-Lorenzo, Pablo Cordero, Manuel Enciso and Angel Mora: here the authors consider the problem of formal concept analysis, and in particular the problem of analyzing data by means of sets of implications; optimization is taken care of by means of Simplification Logic, presented by the authors, and shown to be equivalent to Armstrong axiomatization. *Abductive Reasoning in Dynamic Epistemic Logic - Generation and Selection of Hypothesis*, by Ismael Delgado-Arróniz: the paper aims to represent abductive reasoning in the context of epistemic logic; in particular, it is focused on highlighting the role of experience as a tool to select the best explaining hypothesis, and several methods are presented. *LCC-Program Transformers through Brzozowski's Equations*, by Enrique Sarrión-Morillo: an alternative to classic Kleene translation based on equational methods for program transformation is presented, aimed to dealing with the Logic of Change and Communication; this alternative method is studied in terms of complexity, formulas' length, and simplicity of implementation. And, finally, *Mereology and Temporal Structures*, by Pedro González Núñez: is this paper a semantic structure based on Kamp frames is presented as the theoretical basis to deal with mereological concepts such as *part of* in a spatio-temporal context; a first-order multi-modal language is described with a non-classical semantics to work with such structures.

Distributed explicit knowledge and collective awareness

Alfredo Burrieza Claudia Fernández-Fernández
Universidad de Málaga, Andalucía Tech
Departamento de Filosofía, España
burrieza@uma.es, clau5anj@gmail.com

Abstract

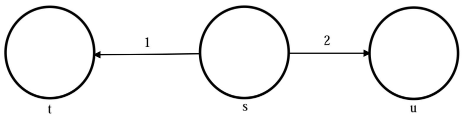
Our goal is to model communication in a group among agents with limited resources. Therefore we redefine the concept of distributed knowledge in order to distinguish between implicit and explicit knowledge. The most useful tool to do this is the concept of awareness, as a way of limiting and selecting what the agents really know. In this sense we propose a collective awareness that ensures full explicit communication and shows the dynamic aspects of the information exchange in a group. Depending on the definition of this collective awareness we are able to model the various ways in which the agents behave during communication.

1 Introduction

The standard definition of distributed knowledge (see [2]) intends to capture the knowledge flow between the agents in a group. The group is treated as another agent (the 'wise man') and acquires some knowledge that no individual agent on his own could posses. This new knowledge is derived from the information exchange between the agents in the group. Consider the standard epistemic language with the distributed knowledge operator D_G (where G is a set of agents). Let $\mathcal{M} = (S, R_1, \ldots, R_n, V)$ be any Kripke model (more details below) and $\models$ be the usual satisfaction relation, defined on the model. Then we have that for any $s \in S$:

$$\mathcal{M}, s \models D_G\varphi \quad \text{iff} \quad \mathcal{M}, t \models \varphi, \text{for all } t \text{ such that } (s,t) \in \textstyle\bigcap_{i \in G} R_i \qquad (D1)$$

This definition gives rise to strange cases in which we can model a group knowledge that has been established without a justified information exchange between the agents in the group, we could call these type of situations 'mysterious knowledge'. Imagine a simple case: Agent 1 knows that the movie X is not shown at at 17:00h, agent 2 knows it, too; but both (as a group), after communication, know that the movie is shown at 17:00h (p). How is this possible? Consider the following model $\mathcal{M}_{\text{movie}} = (S, R_1, R_2, V)$, where:



- $S = \{s, t, u\}$; $R_1 = \{(s, s), (s, t)\}$; $R_2 = \{(s, s), (s, u)\}$; $V(q) = \varnothing$ for all atom q of the language.

We have that $\mathcal{M}, s \models D_G p$, i.e., the group G (where $G = \{1, 2\}$) knows p at s, according to the definition above, in ($D1$) (since $\bigcap_{i \in G} R_i = \varnothing$). But p is no logical consequence of the combined knowledge of the agents at s, since this knowledge is consistent and $\neg p$ is part of it. This is a case of 'mysterious knowledge' where we cannot justify coherently how the group acquires their knowledge. In our example we use a frame with no special properties on the accessibility relations. In [5] the authors present another case of mysterious knowledge where the accessibility relations are equivalence relations.

To avoid this kind of situations, we can use a different distributed knowledge definition that considers the logical consequences of the knowledge of the group members, as in [3]. But, although this situation could be fixed, the information flow happens in an ideal context. The communication that is being modelled focuses on implicit knowledge and belief. It does not take into account agents with limited reasoning, real agents. Our goal is to approach the distributed knowledge of a group whose members have limited reasoning resources.

When a group of agents establishes communication they exchange different kinds of information: knowledge, beliefs, doubts, mistakes, etc. If we focus only on the knowledge, then it should be clear that this knowledge is always explicit. If the agents have limited reasoning resources then we need an appropriate tool that distinguishes between what is implicit or explicit. Fagin et al., in [2], refer to the awareness of the agent as a way of limiting their knowledge (see also [1]).

The information exchange between the agents in a group modifies each individual awareness. Therefore the explicit knowledge of the group, seen as the knowledge of the wise man, depends on the awareness of the group. We could then consider a 'collective awareness' that will be applied to the explicit knowledge of the group in the same way that the individual awareness is applied to the explicit knowledge of the agent. In other words, if we have $K_i^e\varphi \equiv K_i\varphi \wedge A_i\varphi$, where $K_i^e\varphi$ means 'agent i explicitly knows φ', $K_i\varphi$ means 'agent i implicitly knows φ' and $A_i\varphi$ means 'agent i is aware of φ'; we can establish in a natural way that for any group of agents G we have: $D_G^e\varphi \equiv D_G\varphi \wedge A_G\varphi$, where $D_G^e\varphi$ means 'φ is distributed explicit knowledge among the agents in G', $D_G\varphi$ means 'φ is distributed implicit knowledge among the agents in G' and $A_G\varphi$ means 'group G is aware of φ'.

Our aim is to present the minimum conditions under which we can define the collective awareness, that is, under which expressions such as 'group G is aware of

φ' make sense. We want to analyze the results of combining this notion with two different senses of D_G in order to describe the different concepts of explicit group knowledge: ($D1$) previously defined and ($D2$) presented later.

Group knowledge attempts to reflect how the knowledge is gained after an information exchange between the members of the group. This can be thought of as a new agent representing the group (the wise man) who possesses the information resulting from the exchange. Since the information being exchanged is explicit knowledge, the awareness of the wise man needs to be the result of the interaction of the individual awarenesses. In a natural way this knowledge exchange needs to have an impact on the collective awareness of the group.

On the formal account we can reduce the collective awareness to the intersection of all the information of the individual awarenesses (*pure awareness intersection*). Nevertheless, being less strict about this matter we can suppose that the collective awareness will contain *at least* this intersection (*awareness intersection*) (AI). This less strict version enables us to reflect the dynamic aspects of communication, the fact that after the information exchange the individual awarenesses acquire the new shared information. The way in which we represent this notion is by allowing the collective awareness of the wise man to include the additional information that results from communication and that does not belong to, or cannot be a result of, the awareness intersection.

On the other hand, the new information of the collective awareness, that does not belong to any individual, has its origin in the communication itself. Then we can state the *principle of limited collective awareness* (PLCA), according to which the content of the collective awareness can only be generated by the interaction of the information of the individual awarenesses.

In general, the agents can communicate everything they know or not, depending on the context. Thus, there will be some information that they will not communicate to the others, but which nevertheless belongs to their individual awareness. We can consider both these type of models and, furthermore, distinguish between those cases where all the information they communicate is knowledge (*full rational communication*) and where not necessarily all of it is knowledge (*partial rational communication*).

2 Epistemic logic with distributed knowledge and collective awareness

Consider a countable set of propositional letters $\mathcal{P}$ and a finite set of agents $\mathcal{A}g = \{1, \ldots, n\}$, the language $\mathcal{L}^{DA^c}$ of epistemic logic with distributed knowledge and collective awareness is given by the following definition:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \rightarrow \varphi \mid K_i\varphi \mid K_i^e\varphi \mid D_G\varphi \mid D_G^e\varphi \mid A_i\varphi \mid A_G\varphi$$

(where $p \in \mathcal{P}$, $i \in G \subseteq \mathcal{A}g$)

A frame is a tuple $\mathcal{F} = (S, R_1, \ldots, R_n, \mathcal{A}_1, \ldots, \mathcal{A}_n, \mathcal{A}_G)$, where:

1. S is a non-empty set of sates (also called 'worlds').
2. $R_i \subseteq S \times S$ for all $1 \leq i \leq n$. Each R_i is an accessibility relation for agent i.
3. $\mathcal{A}_i \colon S \longrightarrow 2^{\mathcal{L}^{DA^c}}$ for all $1 \leq i \leq n$.
4. $\mathcal{A}_G \colon S \longrightarrow 2^{\mathcal{L}^{DA^c}}$, where $\mathcal{A}_G$ satisfies: for any $s \in S$,

   (AI): $\bigcap_{i \in G} \mathcal{A}_i(s) \subseteq \mathcal{A}_G(s)$,

   (PLCA): $\mathcal{A}_G(s) \subseteq For(ATOM(\bigcup_{i \in G} \mathcal{A}_i(s)))$.

   where

   $For(ATOM(\bigcup_{i \in G} \mathcal{A}_i(s)))$ is the set of formulas generated by the atoms that appear in the formulas of $\bigcup_{i \in G} \mathcal{A}_i(s)$ (*the awareness set in s*).

Note that we did not specify any special properties for the individual awareness on item 3. It is wellknown [2] that $\mathcal{A}_i$ can have different properties (closed under subformulas, generated by a subset of primitive propositions, etc.). Analogous considerations can be expressed regarding collective awareness and depending on the properties of the individual awarenesses. Note also that the two general conditions of 'awareness intersection' and 'limited collective awareness', included on item 4, are the minimum intuitive requirements we impose on the collective awareness frames.

(AI) says that $\mathcal{A}_G(s)$ contains at least the intersection of individual awarenesses (before communication) and it can be expanded with new information (after communication). This new information represents the modifications of the individual awarenesses after communication. This mechanism works similarly to the D_G operator. This operator collects what the agents know after communication, but the model can only reflect, as a picture, what the agents know before communication. In this regard we are dealing with static models.

(PLCA) says that after communication, the collective awareness cannot have more information than the information contained by the set of formulas generated by the atoms of the awareness set. For instance, if no agent in the group has notice about the trigeminus, it is impossible that the information 'the trigeminus is a nerve' can appear in the collective awareness after communication.

A model is a tuple $\mathcal{M} = \{\mathcal{F}, V\}$, where $\mathcal{F}$ is a frame and V is a valuation function $V \colon \mathcal{P} \longrightarrow 2^S$ such that V associates every $p \in \mathcal{P}$ with a subset of S, intuitively the states in which p is true. In addition, a satisfaction relation $\models$ between models and formulas in $\mathcal{L}^{DA^c}$ can be defined. We write $\mathcal{M}, s \models \varphi$ to mean that the formula φ is true at (satisfied in) state s in $\mathcal{M}$ and it can be inductively defined as follows:

$\mathcal{M}, s \models p$	iff	$s \in V(p)$ (for each $p \in \mathcal{P}$)
$\mathcal{M}, s \models \neg\varphi$	iff	$\mathcal{M}, s \not\models \varphi$
$\mathcal{M}, s \models \varphi \wedge \psi$	iff	$\mathcal{M}, s \models \varphi$ and $\mathcal{M}, s \models \psi$
$\mathcal{M}, s \models \varphi \rightarrow \psi$	iff	$\mathcal{M}, s \not\models \varphi$ or $\mathcal{M}, s \models \psi$
$\mathcal{M}, s \models K_i\varphi$	iff	for all t such that $(s, t) \in R_i$: $\mathcal{M}, s \models \varphi$
$\mathcal{M}, s \models A_i\varphi$	iff	$\varphi \in \mathcal{A}_i(s)$
$\mathcal{M}, s \models K_i^e\varphi$	iff	$\mathcal{M}, s \models K_i\varphi$ and $\mathcal{M}, s \models A_i\varphi$
$\mathcal{M}, s \models A_G\varphi$	iff	$\varphi \in \mathcal{A}_G(s)$

We can extend the satisfaction relation with both the following alternatives for distributed knowledge, $(D1)$ introduced before and $(D2)$ below (see [3]):

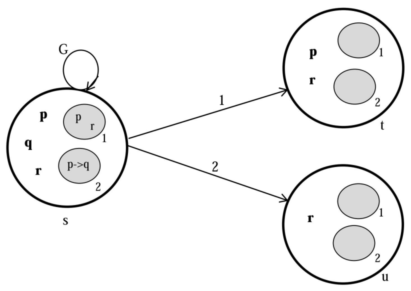
$\mathcal{M}, s \models D_G\varphi$	iff	$\mathcal{M}, t \models \varphi$, for all t such that $(s, t) \in \bigcap_{i \in G} R_i$	(D1)
$\mathcal{M}, s \models D_G\varphi$	iff	$\{\psi \in \mathcal{L}^{A^c} \mid \mathcal{M}, s \models K_i\psi$ for some $i \in G\} \Vdash \varphi$	(D2)

We also have:

$\mathcal{M}, s \models D_G^e\varphi$	iff	$\mathcal{M}, s \models D_G\varphi$ and $\mathcal{M}, s \models A_G\varphi$

Note that in $(D2)$ we use $\mathcal{L}^{A^c}$, i.e., the language resulting from $\mathcal{L}^{DA^c}$ by dropping D_G and D_G^e, avoiding this way circularity in the definition, as pointed out in [3]. On the other hand, the symbol $\Vdash$ is a relation of *logical consequence* between a set of formulas and one formula. In general, $\Phi \Vdash \varphi$ means that for every model $\mathcal{M}$ and every state s in $\mathcal{M}$, if all formulas in Φ are satisfied in s, φ is also satisfied in s. In addition, the notions of *satisfiability* and *validity* are defined as usual.

Example 1. *In the following model $\mathcal{M} = (S, R_1, R_2, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_G, V)$ we take s as the actual state and $G = \{1, 2\}$. We define the model only attending to atoms p, q, r:*



- $S = \{s, t, u\}$.
- $R_1 = \{(s, s), (s, t)\}$; $R_2 = \{(s, s), (s, u)\}$.
- $\mathcal{A}_1(s) = \{p, r\}$; $\mathcal{A}_2(s) = \{p \rightarrow q\}$; $\mathcal{A}_1(t) = \mathcal{A}_2(t) = \mathcal{A}_1(u) = \mathcal{A}_2(u) = \emptyset$.
- *$\mathcal{A}_G(s)$ is defined below in different ways;* $\mathcal{A}_G(t) = \mathcal{A}_G(u) = \emptyset$.

- $V(p) = \{s, t\}$; $V(q) = \{s\}$; $V(r) = S$.

Before communication we have:

- $\mathcal{M}, s \models K_1 p \qquad \mathcal{M}, s \models K_1 r \qquad \mathcal{M}, s \models K_2(p \to q)$

The agents can interchange information. And what happens after communication? We can contemplate several possibilities. In all of them we have the same result using (D1) or (D2):

1. • $\mathcal{A}_G(s) = \bigcap_{i \in G} \mathcal{A}_i(s) = \emptyset$ *(there is no communication at all)*
   The distributed implicit knowledge is infinite ($\mathcal{M}, s \models D_G p, \mathcal{M}, s \models D_G r, \ldots$*), and the distributed explicit knowledge does not increase (it was empty and remains empty).*
2. • $\mathcal{A}_G(s) = \{r\}$
   1 does not speak about all his knowledge. 2 does not speak at all. We will focus on the distributed explicit knowledge. As a consequence: $\mathcal{M}, s \models D^e_G r$ *(only!).*
3. • $\mathcal{A}_G(s) = \{p, q, r, p \to q\}$
   1 speaks about all his knowledge. 2 speaks about all his knowledge. In particular they can conclude q, since $\mathcal{M}, s \models D_G q$ *and, as* $q \in \mathcal{A}(s)$*, we obtain* $\mathcal{M}, s \models D^e_G q$*.*
4. • $\mathcal{A}_G(s) = \{p, p \to q\}$
   1 speaks about part of his knowledge. 2 speaks about all his knowledge. Although they communicate p and $p \to q$ *they cannot conclude q despite all (they may lack Modus Ponens). Indeed, though* $\mathcal{M}, s \models D_G q$ *we have* $\mathcal{M}, s \not\models D^e_G q$*, since* $q \notin \mathcal{A}(s)$*.*

3 Models with rational information flow

We say that there is ‘rational information flow’ in a group whenever the collective awareness of a group acquires knowledge from the individual agents after communication. We can define two versions of rational information flow: (*i*) all the acquired information needs to be knowledge (strong version), or (*ii*) at least part of the acquired information is knowledge (weak version). We can also specify two ways in which the information flows: either all explicit knowledge is acquired or only part of it.

We are interested in collecting classes of structures with rational communication flow and in defining the concept of explicit distributed knowledge in relation to this property. Note that the minimum conditions (AI) and (PLCA) do not commit themselves to neither of these versions. There is also no guarantee that those intersections cannot be empty. However, if there is a real knowledge exchange between the agents in the group those sets can never be empty. The rational communication flow models are of interest because they ensure fruitful knowledge exchange, where the agents have really learned new information.

In what follows we will use the following notation: We will call $KS_G(\mathcal{M}, s)$ and $KS^e_G(\mathcal{M}, s)$ respectively *implicit knowledge set* and *explicit knowledge set* of a group of agents G in a state s of a model $\mathcal{M}$, defined as follows:

$$KS_G(\mathcal{M}, s) = \{\psi \in \mathcal{L}^{DA^c} \mid \mathcal{M}, s \models K_i\psi \text{ for some } i \in G\}$$

$$KS^e_G(\mathcal{M}, s) = \{\psi \in \mathcal{L}^{DA^c} \mid \mathcal{M}, s \models K^e_i\psi \text{ for some } i \in G\}$$

Consider the following four possibilities for A_G that reflect different forms of communication:

$$\mathcal{A}_G(s) = \bigcap_{i \in G} \mathcal{A}_i(s) \tag{A1}$$

$$\mathcal{A}_G(s) \subseteq \{\psi \in \mathcal{L}^{DA^c} \mid KS^e_G(\mathcal{M}, s) \Vdash \psi\} \tag{A2}$$

$$\{\psi \in \mathcal{L}^{DA^c} \mid KS^e_G(\mathcal{M}, s) \Vdash \psi\} \subseteq \mathcal{A}_G(s) \tag{A3}$$

$$\{\psi \in \mathcal{L}^{DA^c} \mid KS^e_G(\mathcal{M}, s) \Vdash \psi\} \cap \mathcal{A}_G(s) \neq \emptyset \tag{A4}$$

Combining these notions of $\mathcal{A}_G$ with $(D1)$ and $(D2)$ we are able to model many ways of knowledge transfer between the agents.

If we assume $(A1)$, then two different things may happen: either there is no communication at all, or everything the agents communicate is already known by them. If we assume $(A2)$, $(A3)$ or $(A4)$ there can be information that the group explicitly knows without the need that any of their members do. The group acquires this knowledge after deriving it from the knowledge of their members. On the other hand, regarding the specific case of $(A2)$, the collective awareness is only 'rational'; that is, it only contains the logical consequences of the explicit knowledge of their members.

In the case of $(A3)$ and $(A4)$ the collective awareness has a 'rational core', $\{\psi \in \mathcal{L}^{DA^c} \mid KS^e_G(\mathcal{M}, s) \Vdash \psi\}$, standing for the information that can be derived from the explicit knowledge set. Regarding $(A3)$ the agents communicate all their knowledge. But the collective awareness is not necessarily reduced to its rational core. This strikes us more intuitive since the awareness can contain inconsistent information. Assuming $(A4)$, there can be members of the group that do not communicate all their knowledge. This has a direct impact on the explicit knowledge of the group which does not contain everything its members really know.

4 Classes of models and full explicit communication

The *Principle of Full Communication* establishes that whenever φ is considered group knowledge, it should be possible for the members of the group to establish φ through communication. It is argued by Van der Hoek et al., in [5], that group knowledge should comply with this principle. They formulate it as follows:

$$\mathcal{M}, s \models D_G\varphi \text{ implies } KS_G(\mathcal{M}, s) \Vdash \varphi$$

The authors use the language $\mathcal{L}^D$ (resulting from $\mathcal{L}^{DA^c}$ by dropping the explicit epistemic and awareness operators). A dissertation about the class of models that comply with this principle using (D1) can be found in [4]. Since we want to deal with real agents whose reasoning resources are limited, the knowledge that is being established through communication needs to be explicit. Hence we can establish the principle of *full explicit communication*:

$$\mathcal{M}, s \models D^e_G\varphi \text{ implies } KS^e_G(\mathcal{M}, s) \Vdash \varphi$$

Full explicit communication can be studied combining the definitions of distributed implicit knowledge, $(D1)$ or $(D2)$, and the conditions on the collective awareness, $(A1)$-$(A4)$ above. The result of this combination is given by the following propositions:

Proposition 1. *In the following classes of models distributed explicit knowledge does not comply with the principle of full explicit communication:*

1. *The class of models that satisfies* $(D1)$ *and either* $(A1)$ *or* $(A3)$ *or* $(A4)$.
2. *The class of models that satisfies* $(D2)$ *and either* $(A1)$ *or* $(A3)$ *or* $(A4)$.

Proof. We will prove in item 2 the case $(D2)$ and $(A4)$. Let $G = \{1, 2\}$ and consider the model $(S, R_1, R_2, \mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_G, V)$, where:
$S = \{s\}$; $R_1 = \{(s, s)\}$; $R_2 = \varnothing$; $\mathcal{A}_1(s) = \{p, q\}$; $\mathcal{A}_2(s) = \varnothing$; $V(p) = \{s\}$; $V(\varphi) = \varnothing$ for all atom φ in $\mathcal{P}$ distinct of p. Assume also that $\mathcal{A}_G(s) = \{p, q\}$ which satisfies (A4), because $KS^e_G(\mathcal{M}, s) = \{p\}$. Now, we have that $\mathcal{M}, s \models D_Gq$ (since $\mathcal{M}, s \models K_2q$) and as $q \in \mathcal{A}_G(s)$, we obtain $\mathcal{M}, s \models D^e_Gq$, but $\{p\} = KS^e_G(\mathcal{M}, s) \nVdash q$. □

The following result is immediate:

Proposition 2. *In the following classes of models distributed knowledge complies with the principle of full explicit communication:*

1. *The class of models that satisfies* $(D1)$ *and* $(A2)$.
2. *The class of models that satisfies* $(D2)$ *and* $(A2)$.

5 Conclusions and future work

We have seen that collective awareness is an adequate concept for modeling communication with information exchange in a group of agents. This notion reflects, in a static way, the dynamics of communication allowing changes and integrating new information. But there are still many unexplored areas in this field, such as: (*i*) Exploring more classes of models that comply with the principle of full explicit communication and defining formal systems to deal with this concept syntactically. (*ii*) Redefining explicit distributed knowledge specifying the type of information that collective awareness can contain. (*iii*) Analyzing the concepts of distributed explicit knowledge and collective awareness from the perspective of Dynamic Epistemic Logic (DEL) and Public Announcement Logic (PAL).

Acknowledgements. This paper was partially supported by the Spanish Research Project TIN2012-39353-C04-01.

References

[1] R. Fagin, J.Y. Halpern. Belief, Awareness, and Limited Reasoning. *Artificial Intelligence*, 34: 39–76, 1988.

[2] R. Fagin, J.Y. Halpern, Y. Moses, M.Y. Vardi. *Reasoning About Knowledge.* The MIT Press, 1995.

[3] J. Gerbrandy. Distributed Knowledge. *Twendial'98: Formal Semantics and Pragmatics of Dialogue. (Joris Hulstijn and Anton Nijholt, eds.)*, TWLT 13: 111–124, 1998.

[4] F. Roelofsen. Distributed knowledge. *Journal of Applied Non-Classical Logics*, 17:2, 255–273, 2007.

[5] W. van der Hoek, B. van Linder, John-Jules Meyer. Group knowledge is not always distributed (neither is it always implicit). *Mathematical Social Sciences*, 38: 215–240, 1999.

A LOGICAL APPROACH FOR DIRECT-OPTIMAL BASIS OF IMPLICATIONS

Estrella Rodríguez-Lorenzo, Pablo Cordero,
Manuel Enciso, Angel Mora
Universidad de Málaga, Andalucía Tech- Spain
estrellarodlor,amora@ctima.uma.es, pcordero,enciso@uma.es

Abstract

In Formal Concept Analysis, knowledge extracted from a data set is represented in two alternative ways: concept lattices and sets of implications. The sets of implications are optimized under different criteria linked to several properties. In this paper the optimization task is strongly based on the Simplification Logic. Specifically, we present a review of how minimal sets of implications (basis) with different properties can be calculated with a logical style. Therefore, different techniques to manipulate them are outlined. Our logic-based approach property fits with the logic programming paradigm and, thus, a Prolog implementation to calculate direct basis from a set of implications is also sketched .

1 Introduction and background

Formal Concept Analysis (FCA) is an useful tool for mining information from a dataset. FCA has been used in different areas: Artificial Intelligence, Databases, Software Engineering, Data Mining, and recently in the Semantic Web.

In this section, we summarize the main concepts regarding FCA. For a more detailed explanation, we refer the reader to [7]. Data are represented through binary tables, named formal contexts $\mathbb{K} := (G, M, I)$, in which a set of objects G and a set of attributes M are related via the binary relation I. From $\mathbb{K}$, two mappings are defined:

- $(\ )' \colon 2^G \to 2^M$ where $A' = \{m \in M \mid g\ I\ m$ for all $g \in A\}$ for all $A \subseteq G$.
- $(\ )' \colon 2^M \to 2^G$ where $B' = \{g \in G \mid g\ I\ m$ for all $m \in B\}$ for all $B \subseteq M$.

In FCA several automated method have been introduced to extract knowledge from formal concepts. This knowledge is extracted in the shape of *concepts* and they can be represented in the so called *Concept Lattice*. A concept is a pair $\langle A, B\rangle \in 2^G \times 2^M$ such that $A' = B$ and $B' = A$ (i.e. a set of objects that are precisely characterized by a set of attributes) and an order relation is established providing a hierarchy in the concept set.

In this paper, we deal with an alternative way to represent this knowledge, that is the set of *implications*. An attribute implication is an expression $A \to B$ where A and B are sets of attributes. A formal context satisfies $A \to B$ if every object that has all the attributes in A also has all the attributes in B. In other terms, $A \to B$ holds (is valid) in $\mathbb{K}$ whenever $A' \subseteq B'$.

The set of all valid implications in a context, called *full implicational system*, gathers the same knowledge as its corresponding concept lattice. However, the first -alternative- approach provides an interesting advantage: since it satisfies the Armstrong's axioms [**?**], some subsets can be considered as representatives of the full implicational systems. Thus, an *implicational system* (briefly IS) for $\mathbb{K}$ is a set Σ of implications satisfing that the valid implications on $\mathbb{K}$ are those that can be derived from Σ using Armstrong's axioms. That is, the implicational system Σ represents all the knowledge obtained from $\mathbb{K}$ in a shorten way. Implicational System knowledge representation is strongly related with two issues:

1. Do Armstrong's axioms can be used efficiently?

2. Since several implicational systems can represent the same knowledge, there is an optimal one?

The first question use to be addressed with indirect methods based on the semantics, avoiding a direct syntactic manipulation provided by Armstrong's inference system. An alternative way that remains faithful to the logic point of view stems from the Simplification Logic. In Section 2 we present this logic that allows an efficient reasoning with implications.

The second question is addressed by characterizing those implicational systems fulfilling some minimality criteria. Such Implicational Systems are usually called *basis*. Among the different basis definitions, the Duquenne-Guigues basis [8], also called Stem basis, has been widely accepted in FCA. This basis is minimal in the number of implications. Nevertheless, as it is shown in [6], Duquenne-Guigues bases tend to have redundant attributes and therefore, an equivalent one having the same cardinality but less attributes can be provided. In that paper, we also propose a method to obtain a basis with minimal size in the left-hand side of the implications.

Other well-known property used to define another kind of bases is *directness*, i.e., a single traversal of the implicational system is enough to compute the closure of an given set of attributes. A basis fulfilling this property is named *direct basis*. This property is usually accompanied by some minimality criteria. We are particularly interested in those ones with minimum size (number of attributes). In [2, 3, 11] several methods to calculate the direct-optimal basis are introduced, where minimality and directness have been joined in the same notion of basis.

Here, our main issue is how to calculate such a direct-optimal basis, providing a tool for a very efficient computation of attribute closures. Our method to calculate the direct-optimal basis [11] is based on Simplification Logic [5], $\mathbf{SL}_{\text{FD}}$, a sound and complete inference system for Implicational Systems. Our logic is strongly based on the Simplification Rule, which describes the redundancy removal of attributes. This method based on $\mathbf{SL}_{\text{FD}}$ is more efficient than previous methods appeared in the literature. To prove this fact, we have developed an illustrative empirical test using Prolog.

2 Simplification Logic and closures

Armstrong's Axioms [1] is the former system introduced to manage implications in a logical style. In this section, we briefly present Simplification Logic ($\mathbf{SL}_{\text{FD}}$ for short), which is an equivalent logic that arises from the idea of simplifying the set of implications by efficiently removing redundant attributes [10]. [1]

The language: Given a non-empty finite alphabet S (namely attributes set), the language of $\mathbf{SL}_{\text{FD}}$ is $\mathcal{L}_S = \{A \to B \mid A, B \subseteq S\}$.

In order to distinguish between language and metalanguage, inside implications, AB means $A \cup B$ and A-B denotes the set difference $A \smallsetminus B$.

Semantics: A context $\mathbb{K}$ is said to be a model for $A \to B \in \mathcal{L}_S$, denoted by $\mathbb{K} \models A \to B$, if this implication holds in the context. For an IS Σ, $\mathbb{K} \models \Sigma$ means $\mathbb{K} \models A \to B$ for all $A \to B \in \Sigma$. If Σ_1 and Σ_2 are implicational systems, $\Sigma_1 \equiv \Sigma_2$ denotes the equivalence of the two sets of implications (i.e. $\mathbb{K} \models \Sigma_1$ iff $\mathbb{K} \models \Sigma_2$ for all context $\mathbb{K}$).

Syntactic derivations: Reflexivity as axiom scheme and the following inference rules named fragmentation, composition and simplification are considered in $\mathbf{SL}_{\text{FD}}$.

$$\texttt{[Ref]}\ \frac{}{A \to A} \qquad \texttt{[Frag]}\ \frac{A \to BC}{A \to B} \qquad \texttt{[Comp]}\ \frac{A \to B,\ C \to D}{AC \to BD} \qquad \texttt{[Simp]}\ \frac{A \to B,\ C \to D}{A(C\text{-}B) \to D}$$

[1] See [9] for a more detailed presentation of the Simplification Logic and its advantadges.

Given a set of implications Σ and an implication $A \to B$, $\Sigma \vdash A \to B$ denotes that $A \to B$ can be derived from Σ by using the axiomatic system in a standard way. If any implication valid in a formal context $\mathbb{K}$ can be derived from Σ and vice versa, then Σ is called an implicational system (IS) for $\mathbb{K}$.

The main advantage of $\mathbf{SL}_{\text{FD}}$ is that its inferences rules induce equivalence relations among sets of implications. Moreover, these equivalencies are enough to compute all the derivations (see [9] for further details and proofs).

Theorem 2.1 ([9])**.** *In* $\mathbf{SL}_{\text{FD}}$ *logic, the following equivalences hold:*

1. *Fragmentation Equivalency* **[FrEq]***:* $\{A \to B\} \equiv \{A \to B\text{-}A\}$
2. *Composition Equivalency* **[CoEq]***:* $\{A \to B, A \to C\} \equiv \{A \to BC\}$
3. *Simplification Equivalency* **[SiEq]***: If* $A \cap B = \emptyset$ *and* $A \subseteq C$ *then*

$$\{A \to B, C \to D\} \equiv \{A \to B, C\text{-}B \to D\text{-}B\}$$

Note that these equivalencies (read from left to right) remove redundant information, approaching our main spirit when creating $\mathbf{SL}_{\text{FD}}$.

Definition Let $\Sigma \subseteq \mathcal{L}_S$ be an IS and $X \subseteq S$. The closure of X wrt Σ is the largest subset of S, denoted X^+_Σ, such that $\Sigma \vdash X \to X^+_\Sigma$.

3 Direct-Optimal basis

A mainstream topic in FCA is the study of different properties to be fulfilled by implicational systems. As we have mentioned in the introduction, our goal is the minimization of the computation of attribute closure computations. In [3], Bertet and Monjardet present a survey concerning implicational systems and basis. They show the equality among five basis presented in different works. They also study the properties they satisfy, including directness and minimality. The conclude that all the presented bases are equivalent to the so called direct-optimal basis.

The formal definition of these properties (minimality, optimality and directness) is the following:

Definition An IS Σ is said to be:

- *minimal* if $\Sigma \smallsetminus \{A \to B\} \not\equiv \Sigma$ for all $A \to B \in \Sigma$,
- *minimum* if $\Sigma' \equiv \Sigma$ implies $|\Sigma| \leq |\Sigma'|$, for all IS Σ',
- *optimal* if $\Sigma' \equiv \Sigma$ implies $||\Sigma|| \leq ||\Sigma'||$, for all IS Σ',

where $|\Sigma|$ is the cardinal of Σ and $||\Sigma||$ denotes its size, i.e. $||\Sigma|| = \sum_{A\to B\in\Sigma}(|A| + |B|)$.

An IS is said to be a basis if it is minimal. We are looking for bases satisfying this property because the less cardinal of the IS, the better the performance of closure computation. Moreover, to reduce the cost of the computation of closures, we demand for another criterion: directness. It ensures that the closure computation only needs one traversal of the IS.

Although closure is a linear task, the search for fast and easy closure methods is a hot topic because several problems are addressed by exhaustively computing closures. Thus, many of the classical algorithms in FCA are solved by intensively computing the closure of a set of attributes. A significant reduction in the performance of closure methods is relevant when a huge -sometimes exponential-number of closures are executed to solve the original problem.

For this reason, we have paid attention to the notion of direct-optimal basis [2, 3], introduced as follows:

Definition Let S be a set of attributes, an IS Σ is said to be *direct* if, for all $X \subseteq S$:

$$X^+_\Sigma = X \cup \{b \in B \mid A \subseteq X \text{ and } A \to B \in \Sigma\}$$

Moreover, Σ is said to be *direct-optimal* if it is direct and, for any direct IS Σ', $\Sigma' \equiv \Sigma$ implies $||\Sigma|| \leq ||\Sigma'||$.

In other words, Σ is said to be direct-optimal if it is direct and it is optimal among all the equivalent direct ISs. In [3], the existence and the unicity for a direct-optimal basis equivalent to a given one was proved.

In the following section, we are introducing a method to calculate the direct-optimal basis for any IS proposed in [11] and its improved version proposed in [12].

4 Computing direct-optimal basis

This section deals with the integration of the techniques proposed by Bertet et al. [2–4] and the Simplification Logic proposed by Cordero et al. [5]. First, we developed a function to get the direct-optimal basis whose first step is the narrowing of the implications (see [11]). To this end, in this paper we use *reduced* ISs.

Definition An IS Σ is *reduced* if $B \neq \emptyset$ and $A \cap B = \emptyset$ for all $A \to B \in \Sigma$.

Obviously, an arbitrary IS Σ can be turned into a reduced equivalent one Σ_r by applying **[FrEq]**, and by removing implications of the form $A \to \emptyset$. The method proposed here to get a direct optimal basis begins with this transformation, preserving reduceness in further steps.

In [3,4] the authors apply two completely separated stages, first is focussed in directness and, later, an optimization stage is carried out by removing redundant implications. In our method, we have introduced a new inference rule covering in juts one step both properties: directness and minimality. The kernel of the new method is the so named Strong Simplification:

$$\texttt{[sSimp]} \quad \text{If } B \cap C \neq \emptyset \text{ and } D \not\subseteq A \cup B, \frac{A \to B, C \to D}{A(C\text{-}B) \to D\text{-}(AB)} \tag{1}$$

The exhaustively application of this rule to a reduced IS renders an equivalent direct and reduced implicational system, direct-reduced IS in the following. As we proved in [12], the implicational system Σ_{dr} generated from an IS Σ is defined as the smallest one containing Σ which is closed for `[sSimp]`.

As a final step, the three first equivalencies from Theorem 2.1 are used to remove redundant information preserving directness. The implicational system generated in this way by applying these equivalences is named *simplified* IS.

To conclude, the method turns the direct-reduced implicational system obtained in previous stages into an equivalent simplified-direct-reduced one Σ_{sdr} [12]. Indeed, Σ_{sdr} is exactly the direct-optimal basis.

Although the direct-optimal basis is unique, the cost of its computation varies depending on the proposed method. So, to efficiently solve the original problem demanding closure computation, a reduction in the computation of the basis is demanded. The exponential cost of this process is due to the generation of the direct implicational system. In [12], we reduce the input of this stage. The reduction step described above is substituted by simplification, providing a greater reduction in the redundancy. Now, simplification is achieved by applying all the four equivalences in Theorem 2.1 to remove all the redundant attributes in the implications.

Finally, we include here a Prolog implementation[2] based on $\mathbf{SL}_{\text{FD}}$ to calculate the direct-optimal basis from a IS. Due to the fact that our methods are based on logic, Prolog prototypes can be developed in a more direct way.

The input of the Prolog program is a set of attributes S and a set of implications Σ over the attributes in S. The output is the direct optimal basis equivalent to this set of implications. The main predicate of the method developed is `directoptimalSL`. There are three main operations in the method: the first one, `applySL` predicate, executes the four equivalences of $\mathbf{SL}_{\text{FD}}$ the second one,

[2]Available at `http://www.lcc.uma.es/~enciso/do2Simp.zip`

`sSimp`, is a predicate which applies exhaustively the [sSimp] rule (1) to any pair of implications to obtain a direct IS. The last one `applySimplification` renders the simplified-direct-reduced basis: Σ_{sir}. When an implication is added in one step of this execution, the flag `fixpoint` takes the value `false` in order to repeat the method again until the fixpoint is reached. An sketch of this process is showed here.

```
directoptimalSL(Input,Output):-
     ...
     fixPoint_Non,
     applySL,
     applysSimp,
     applySimplification,!.
applySL:-
    siEq, rSiEq, CoEq,FrEq,
    applySL.
applySL.
applysSimp:-
   read2implications(implication(A,B),
                    implication(C,D)),
    sSimp(implication(A,B),implication(C,D),
    fail.
applysSimp.
applySimplification:-
    siEq, CoEq,FrEq,
    applySimplification.
applySimplification.
sSimp(implication(A,B),implication(C,D),
  implication(ACminusB,DminusAB)):-
     union(A,C,AC), difference(AC,B,ACminusB),
     union(A,B,AB), difference(D,AB,DminusAB),
     fixpoint(false),!.
```

Example In this example, we will compute the direct basis of the following set of implications stored in a file called `ganter.txt`:

```
implication([a],[b,c]).
implication([d],[b]).
implication([c],[b]).
implication([a,b,c,d],[e,g]).
implication([a,b,c,e],[d,g]).
```

We call the Prolog predicate:

```
directoptimalSL('ganter.txt','Outganter.txt').

-> Equivalences:     CoEq + SiEq
       implication([c],[b]) +
implication([a,b,c,e],[d,g]) |---
        implication([a,c,e],[d,g])  added
....
***    A DIRECT IS
implication([c], [b]).
implication([d], [b]).
implication([a, b, c, e], [d, g]).
implication([a, b, c, d], [e, g]).
implication([a], [b, c]).
implication([a, c, e], [d, g]).
implication([a, c, d], [e, g]).
implication([a, e], [d, g]).
implication([a, d], [e, g]).

*** BEGIN   Simplification  **
->Equivalences:  SiEq
   implication([c],[b]) +
   implication([a,b,c,e],[d,g]) |---
   implication([a,b,c,e],[d,g])  removed
   implication([a,c,e],[d,g]) yet exist
...
** OUTPUT: DIRECT OPTIMAL BASIS
implication([c], [b]).
implication([d], [b]).
implication([a], [b, c]).
implication([a, e], [d, g]).
implication([a, d], [e, g]).
```

5 Conclusions

In this work, we have outlined two methods to calculate the direct-optimal basis. We apply such methods to reduce the cost in closure computations. In FCA, several methods exhaustively calculate closures of attribute sets and this kind of basis allows their computation in just one traverse of the Implicational System. Prolog has been used as an useful tool to quickly develop prototypes of these methods. Due to space limitations a comparison is omitted and it could be the goal of an extended work. The development of an integrated tool with all the algorithms to manipulate implications in the direct-optimal issue is a future work.

Acknowledgment

This work has been partially supported by grants TIN11-28084 and TIN2014-59471-P of the Science and Innovation Ministry of Spain.

References

[1] W W. Armstrong, *Dependency structures of data base relationships*, Proc. IFIP Congress. North Holland, Amsterdam: 580–583, 1974.

[2] K. Bertet, M. Nebut, *Efficient algorithms on the Moore family associated to an implicational system*, DMTCS, 6(2): 315–338, 2004.

[3] K. Bertet, B. Monjardet, *The multiple facets of the canonical direct unit implicational basis*, Theor. Comput. Sci., 411(22-24): 2155–2166, 2010.

[4] K. Bertet, *Some Algorithmical Aspects Using the Canonical Direct Implicationnal Basis*, CLA:101–114, 2006.

[5] P Cordero, A. Mora, M. Enciso, I.PéÂŐrez de Guzmán, *SLFD Logic: Elimination of Data Redundancy in Knowledge Representation*, LNCS, 2527: 141–150, 2002.

[6] P. Cordero, M. Enciso, A. Mora, M. Ojeda-Aciego, *Computing Left-Minimal Direct Basis of implications*. CLA: 293–298, 2013.

[7] B. Ganter, *Two basic algorithms in concept analysis*, Technische Hochschule, Darmstadt, 1984.

[8] J.L. Guigues and V. Duquenne, *Familles minimales d'implications informatives résultant d'un tableau de données binaires*, Math. Sci. Humaines: 95, 5–18, 1986.

[9] A. Mora, M. Enciso, P. Cordero, and I. Fortes, *Closure via functional dependence simplification*, I. J.of Computer Mathematics, 89(4): 510–526, 2012.

[10] A. Mora, M. Enciso, P. Cordero, and I. Pérez de Guzmán, *An Efficient Preprocessing Transformation for Functional Dependencies Sets Based on the Substitution Paradigm*, LNCS, 3040: 136–146, 2004.

[11] E. Rodríguez-Lorenzo, K. Bertet, P.Cordero, M.Enciso, and A. Mora, *The Direct-optimal basis via reductions*, CLA, 145–156, 2014.

[12] E. Rodríguez-Lorenzo, K. Bertet, P.Cordero, M.Enciso, A. Mora, and M. Ojeda-Aciego, *From Implicational Systems to Direct-Optimal bases: A Logic-based Approach*, Applied Mathematics & Information Sciences, 2L: 305–317, 2015.

Abductive Reasoning in Dynamic Epistemic Logic - Generation and Selection of Hypothesis

Ismael D. Arroniz
Universidad de Sevilla, Andaluca Tech
ismdelarr@gmail.com

Abstract

We propose to represent abductive reasoning in a dynamic epistemic logic framework. The framework emphasizes the role of experience, defined as the result of a dynamic process over an agent^{TM}s information, in the generation of hypotheses and the selecting the best one. We collect several insights of different contexts in logic, such as IBE or AKM model, and we introduce two extra criteria. Besides the one based on experience, a traditional criterion using the idea of minimality, a pragmatic one is also explored. We also introduce a method to combine different orders.

1 Introduction

Abduction is one of the most important non-monotonic reasoning processes. Traditionally described as the reasoning that goes from facts to their causes, in the process of looking for explanations. Originally studied by Charles S. Peirce we can present abduction with his words:

The surprising fact, C, is observed.
But if A were true, C would be a matter of course.
Hence, there is reason to suspect that A is true.

Abductive reasoning has been useful in fields such as philosophy of science and cognitive science. Abductive reasoning is also useful in Artificial Intelligence, where it has been applied to diagnosis and natural language understanding tasks [11, 1, 8, 7].

Recent advances using Dynamic Epistemic Logic as framework[13, 10, 12] allow us to study explicitly the actions involved in the abductive process, understand as a process that involves an agent™s information. In this paper, we try to add some components to dynamic epistemic logic in order to describe abductive reasoning step by step. We focus on the semantics for space limit reasons. This work is organisation as follows: We present a semantic model that allows us to study the generation and selection of hypotheses based on experience in Section 2. Then in Section 3 we present our basic definitions for the study of abductive reasoning under this new framework. Section 4 presents a method to select explanations using several criteria. Section 5 recalls other proposals that present abductive reasoning like a belief change process. We finish in Section 6 with a summary of our proposal and further lines of inquiry for future work.

2 Semantic model

Epistemic logic and its possible worlds semantics are a powerful framework that allows us to represent an agent™s information not only about propositional facts, but also to the agent™s information. Other proposals also point to the importance of experience in abductive reasoning [6][11]. In epistemic terms, we understand the experience as the result of a dynamic process of an agent™s information. In this process, the agent will change not only their knowledge but also their beliefs. We try to formalize this notion in the epistemic framework. Moreover, in order to approximate us to a more realistic scenario we use a non-omniscient agent. Logical omniscience, useful in some applications, is an unrealistic idealization in some others. Most of the proposals to solve this problem focus on weakening the properties of the agent™s information (usually by distinguishing between implicit and explicit information). We use a framework based on [14] for representing implicit and explicit beliefs that combines a framework for representing implicit and explicit information with plausibility models for representing beliefs. We add some specific components for the purpose of our research.

Definition 2.1 (Best explanation model)**.** *A best explanation model* BE *is a possible words model $M = \langle W, \leq, V, A, S, \leq, C \rangle$ where $M = \langle W, \leq, V \rangle$ is a plausibility model presented in [2] and where:*

- *$A : W \rightarrow \wp(\mathcal{L})$ is the acknowledgement set function, indicating the formulas the agent has acknowledged as true at each possible world.*
- *$S \subseteq \mathcal{L}$ is a finite set of formula of the language. Any element will be considerated an explanation (hypothesis).*
- *$\leq \subseteq (S \times S)$ is a locally well-preorder [1] priority relation over S*

[1]For more details consult [2]

$\leq$ relation allow us to talk about the priority that an agent gives to each explanation. If $t \leq u$, u is at least as priorly as t.

- *$C : \mathcal{L} \longrightarrow \mathbb{N}$ is the cost function, assigned a natural number for any formula of the language. We all know that not every explanation is verifiable with the same cost, whether economic or simplicity reasons. In some cases, it^{TM}s simpler, faster or cheaper to discard some possibilities that are not priorities for what the agent knows or believes but for their easy verification.*

In this framework, we can define the notions of implicit and explicit knowledge and beliefs. The agent knows φ implicitly if and only if φ is true in all the epistemically indistinguishable worlds. The agent knows φ explicitly if, in addition, she acknowledges it as true in all these worlds. The agent believes φ implicitly if and only if φ is true in the most plausible worlds and the agent believes φ explicitly if, in addition, she acknowledges it as true in these *best* worlds.

Implicit knowledge:	$K_{Im}\varphi := [\sim]\varphi$
Explicit knowledge:	$K_{Ex}\varphi := [\sim](\varphi \wedge A\varphi)$
Implicit belief:	$B_{Im}\varphi := \langle\leq\rangle[\leq]\varphi$
Explicit belief:	$B_{Ex}\varphi := \langle\leq\rangle[\leq](\varphi \wedge A\varphi)$

3 Abductive problem and abductive solution

When an agent observes a surprising fact, there is no element of uncertainty about it. We use a public announcement definition [3] modified for our non-omniscient logic to represent this action on the model.

Definition 3.1 (Observation)**.** *Given a* BE *model $M = \langle W, \leq, V, A, S, \preceq, C\rangle$ and a formula χ of propositional language $\chi \in \mathcal{L}_p$*[2]*, the observation operation $\chi!$ produces a model $M_{\chi!} = \langle W', \leq', V', A', S, \preceq, C\rangle$ where:*

$W' := \{w \in W \mid (M, w) \Vdash \chi\}$
$\leq' := \leq \cap (W' \times W')$ and
For all $w \in W'$, $V'(w) := V(w)$ and $A'(w) := A(w) \cup \{\chi\}$

We eliminate all the worlds, where χ is false. Moreover, χ is added to A, a function of acknowledgment. The agent^{TM}s knowledge about χ becomes explicit. In epistemic logic terms, we can say that an abductive problem is generated

[2]Formulas of propositional langugage $\mathcal{L}_p$ are given by:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \psi$$

when there exists a formula that was not explicitly known prior to the agent™s observation.

Definition 3.2 (Abductive problem)**.**

$$\chi \textit{ is an abductive problem iff } \chi \in \mathcal{L}_p \textit{ and } (M_{\chi!}, w) \Vdash K_{Ex}\chi \textit{ and } (M, w) \nVdash K_{Ex}\chi$$

Knowing that $(M_{\chi!}, w)$ represents the sub-model obtained when χ is observed, we define an abductive problem as a fact that is not known explicitly in the first instance but it is before the observation and can be expressed in the propositional language.[3]

After the observation, the agent tries to find what they could have been able to infer from the observation that raised the problem. We define this process as an action called Generation in our semantic model. We propose that an abductive solution is a formula that, together with the background theory (including knowledge and beliefs), entails the surprising observation.

Definition 3.3 (Generation)**.** *Given a model $M = \langle W, \leq, V, A, S, \preceq, C\rangle$ y χ, and an abductive problem in (M, w), the generation operation $\chi?$ produces a model $M_{\chi?} = \langle W, \leq, V, A, S', \preceq', C\rangle$ where:*

$$S' = \{\texttt{DNF}(\psi) \mid \psi \to \chi \in A(w)\} \text{ for all } w \in W$$

S is the explanation set where we generate all the possible candidates of explanations; the antecedents of conditionals that have χ as consequent in the acknowledgement function A We use DNF[4] as a method to standardise and identify any formula for its syntactic form. All candidates generated are sorted based on the background theory and, therefore, on what we call experience.

$\preceq'$ is defined as follows:

$$\psi_1 \preceq' \psi_2 \text{ iff } \begin{cases} (M, w) \Vdash K_{Ex}(\psi_2 \to \chi), \text{or} \\ (M, w) \nVdash K_{Ex}(\psi_2 \to \chi) \text{ and exists} \\ w_1 \in MM(\psi_1 \to \chi), w_2 \in \mathrm{MM}(\psi_2 \to \chi) \text{ such that } w_1 \leq w_2 \end{cases}$$

Where M is the set of world where the implication is acknowledge

$$M(\beta_1 \to \beta_2) = \{w \in W \mid \beta_1 \to \beta_2 \in A(w)\}$$

[3]With this restriction, we avoid technical problems generated by more expressive languages.

[4]$\texttt{DNF}(\psi)$ is a disjunctive normal form that validates the following relation:

$$\vdash \psi \leftrightarrow \texttt{DNF}(\psi)$$

and MM is the maximum set of worlds in M

$$\text{MM}(\beta_1 \rightarrow \beta_2) = \{w \in M(\beta_1 \rightarrow \beta_2) \mid \text{ for all } u \in M(\beta_1 \rightarrow \beta_2), u \leq w\}$$

In words, if an hypothesis is part of a implication that an agent knows explicitly, she puts this hypothesis at the top in the priority order. If not, an hypothesis will be more priorly that another if the most plausible world where the agent is acknowledged of the implication that contain the hypothesis as antecedent is more plausible that the maximum plausible world where the agent is acknowledged of the implication that contain the second hypothesis as antecedent.

4 Selecting the best explanation

Some authors argue that abduction is just the Generation step, and together with Selection is part of a more complex process called Inference to the Best Explanation (IBE)[9]. We think the name is not important because everything is part of an explanatory inference, and avoiding this debate, we make significant advances in the study of the selection stage. Traditionally, the criteria to select the best explanation was syntactic, referring to the complexity of the formula. Now we try to formalize a more pragmatic approach. In addition to the experiential criteria, we consider it appropriate to add more ways to sort explanations. According to this, we introduce two criteria more than the experiential approach with the requirement of a method to combine different criteria. We combine them using social-choice techniques[5], and we apply a hierarchy to these criteria. At the top, we consider the experience as the best way to prioritize explanations. We use $\preceq'$ relation detailed in Definition 3.3. At an inferior level, we consider a logic order $\prec_{log}$ that follows a syntactic criterion.

$$\psi_1 \prec_{log} \psi_2$$

if any of the following cases is true:

- ψ_2 is a formula α, being α an atomic formula *(a)*
- ψ_2 is a formula β and ψ_1 is not a formula α, being β the negation of an atomic formula *($\neg a$)*
- ψ_2 is a formula γ and ψ_1 is not a formula α nor β, being γ the disjunction of literals ($\neg A \vee B$)
- ψ_2 is a formula δ and ψ_1 is not a formula α nor β neither γ, being δ the conjunction of literals ($\neg A \wedge B$)

- ψ_2 is a formula ϵ and ψ_1 is not a formula ϵ, being ϵ the disjunction of conjunctions $(A \wedge \neg B) \vee C$

Because all possible explanation generated in Definition 3.3 is in its disjunctive normal form (DNF) we know their syntactic structure is α, β, γ, δ or ϵ kind. Finally, we consider the context criteria $\preceq_C$ that tells us about the contextual differences between explanations. We assign a numerical value to any hypothesis that defines the difficulty of the verification. Using the arithmetic symbol $\geqslant$ we state an order based on $\preceq_C$ the component of the best explanation model BE

$$\psi_1 \preceq_C \psi_2 \text{ syss } C(\psi_1) \geqslant C(\psi_2)$$

In some cases, we need to prioritize explanations based on their practicality. Any of these orders act only when the superior order is not definitive enough to order explanations, resolving the draw. In order to represent this notion in a semantic model, we describe an action that combines three orders with a final priority relation $\preceq_f$:

Definition 4.1 (Selection)**.** *Given a model $M = \langle W, \leq, V, A, S, \preceq, C\rangle$ the operation selection $\Rsh$ produces a model $M_{\Rsh} = \langle W, \leq, V, A, S, \preceq_f, C\rangle$ where:*
$\psi_1 \preceq_f \psi_2$ iff any of the following cases is true:

- $\psi_1 \prec \psi_2$
- $\psi_1 \precsim \psi_2$ *and* $\psi_1 \prec_{log} \psi_2$
- $\psi_1 \precsim \psi_2$ *and* $\psi_1 \precsim_{log} \psi_2$ *y* $\psi_1 \preceq_C \psi_2$

It is desirable that only one explanation is at maximum. Cases where the final order result with two different explanations at the top may also occur. In this situation, there is need to find some other criterion that distinguishes them. Our method is perfectly applicable to different approaches in the hierarchy of the various orders.

5 Belief revision

At the belief revision stage, we focus on the agent$^{\text{TM}}$s information changes. Once an agent establishes an explanation as a definitive best explanation, the agent modifies their beliefs and, therefore, the information they hold about the situation. As noted above, many studies have linked belief revision with abduction [4]. bductive reasoning does not guarantee that the hypothesis is correct. For that reason it cannot eliminate such possible worlds where the best explanation is not true, it just gives them a higher order of plausibility. In our model, we represent it with an action of belief revision, based on other works [2].

Definition 5.1 (Belief Revision)**.** *Let $M = \langle W, \leq, V, A, S, \preceq, C \rangle$ be a model and a let χ be a formula, the belief change $\Uparrow \chi$ operation produces a model $M_{\Uparrow\chi} = \langle W, \leq', V, A, S, \preceq, C \rangle$ differing from M in:*
$w \leq' u$ iff any of the following cases is true:

- $(M, u) \Vdash \chi \wedge A\chi$ *and* $w \leq u$
- $(M, w) \Vdash \neg(\chi \wedge A\chi)$ *and* $w \leq u$
- $(M, w) \Vdash \neg(\chi \wedge A\chi)$ *and* $(M, u) \Vdash \chi \wedge A\chi$ *and* $w \sim u$

A world u will be at least as plausible as a world w, if and only if they already are of that order and u satisfies χ, or they already are of that order and w satisfies $\neg\chi$ or they are comparable, w satisfies $\neg\chi$ and u satisfies χ. This operation preserves the properties of the plausibility relation and hence preserves plausibility models.

6 Summary and further work

From some existing tools of dynamic epistemic logic, we have argued that experience plays a key role in abductive reasoning. We have extended the language of a non-omniscient epistemic logic with some components that collect some intuitions of deliberation about what explanations come to a surprising fact and which one is the priority. In addition, our proposal emphasizes more than other perspectives, the role of experience, understood as the prior information the agent holds. We have also added a contextual component to the abduction. To order the explanations, we have taken into account the practicality and efficiency of verification.

As possible extensions of our proposal, we consider the fact that abductive problems so far have been limited to phenomena and not epistemic problems. It would be interesting to study how some basic intuitions about abduction would be applicable to problems of the type $K\varphi \wedge \neg\varphi$. *I know φ, however $\neg\varphi$ is the case.* Another interesting research line is to consider a multi-agent system, where concepts such as common knowledge or distributed knowledge, as well as public and private announcements, are taken into account. This would be interesting to study under an experience approach.

References

[1] ALISEDA, A. Abduction as Epistemic Change: A Peircean Model in Artificial Intelligence, *P. A. Flach y A. C. Kakas, eds., Abduction and Induction, Kluwer, Dordrecht, 2000*, pages 45-58

[2] A. BALTAG and S. SMETS. A qualitative theory of dynamic interactive belief revision. *In G. Bonanno, W. van der Hoek, and M. Wooldridge, editors, Logic and the Foundations of Game and Decision Theory (LOFT7), volume 3 of Texts in Logic and Games,* pages 13 60.

[3] A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements, common knowledge and private suspicions. *Technical Report SEN-R9922, CWI, Amsterdam, 1999.*

[4] BOUTILLIER, C. BECHER, V. Abduction as belief revision, *Artificial Intelligence 77, 1, 1995* pages 43-94

[5] CHEVALEYRE, Y. ENDRISS,U. LANG,J., MAUDET,N. A Short Introduction to Computational Social Choice, *SOFSEM 2007: Theory and Practice of Computer Science, Springer*, pages 51-69

[6] HOFFMANN, M. Hay una lgica de la abduccin?, *Analoga Filosfica, M©xico 12/1 (1998)*, pages 41-56

[7] KAKAS, A.C.; KOWALSKI, R.A.; TONI, F. (1993). Abductive Logic Programming. *Journal of Logic and Computation Vol. II,6. 1992* pages 719“770

[8] KUIPERS, T. Abduction aiming at empirical progress of even truth aproximation leading to a challenge for computational modelling, *Foundations of Science, Vol. IV, 1999.* pages 307-323

[9] LIPTON, P. Inference to the best explanation *W.H. Newton-Smith (eds.) A Companion to the Philosophy of Science (Blackwell, 2000)* pages 184-193.

[10] NEPOMUCENO, ., SOLER-TOSCANO, F., VELZQUEZ-QUESADA, F. An epistemic and dynamic approach to abductive reasoning: selecting the best explanation *Logic Journal of the IGPL, Vol. XXI,6, 2013* pages 943-961

[11] SOLER TOSCANO, F. Razonamiento abductivo en lgica clsica, Cuadernos de Lgica, epistemologa y lenguaje, *College Publications, 2012*

[12] SOLER-TOSCANO, F., VELZQUEZ-QUESADA, F. Generation and Selection of Abductive Explanations for Non-Omniscient Agents, *Journal of Logic, Language and Information, Vol. XXIII, 2, 2014*, pages 141-168

[13] VELZQUEZ-QUESADA, F., SOLER-TOSCANO, F., NEPOMUCENO, . An epistemic and dynamic approach to abductive reasoning: Abductive problem and abductive solution, *Journal of Applied Logic, Vol. XI, 4, 2013*, pages 505“522

[14] VELZQUEZ-QUESADA, F. Dynamic Epistemic Logic for Implicit and Explicit Beliefs, *Journal of Logic, Language and Information vol. XXIII,2* pages 107-140, 2014

LCC-PROGRAM TRANSFORMERS THROUGH BRZOZOWSKI'S EQUATIONS

Enrique Sarrión-Morillo. Universidad de Sevilla, Andalucía Tech.
Dpto. Filosofía y Lógica y Filosofía de la Ciencia. esarrion@us.es

Abstract

The original work about Logic of Communication and Change uses Kleene's translation from finite automata to regular expressions in program transformers. It appears in the axioms set that accomplish to reduce LCC to PDL. This work presents an elegant matrix treatment of Brzozowski's equational method for program transformers. The two alternatives generate equivalent formulas although the obtained ones through Brzozowski's method are usually much smaller; moreover this method possess computational advantages because its complexity in typical cases (but not the worst) is polinomial, whereas in the original LCC paper is always exponential.

Keywords: Brzozowski's equational method, propositional dynamic logic, logic of communication and change, program transformer.

1 Introduction

The *Logic of Communication and Change* (LCC) is a powerful logic system [10] consisting of a Propositional Dynamic Logic [6] (PDL) interpreted epistemically and the action models machinery [3, 2] for representing the knowledge about actions, allows to model diverse epistemic actions and also factual changes.

Such as other logical frameworks, LCC formulas are interpreted over epistemic models: an *epistemic model* M is a triple $(W, \langle R_a \rangle_{a \in \mathsf{Ag}}, V)$ where $W \neq \emptyset$ is a set of worlds, $R_a \subseteq (W \times W)$ is an epistemic relation for each agent $a \in \mathsf{Ag}$ and $V : \mathsf{Var} \to \wp(W)$ is an atomic evaluation[1].

Meanwhile, the action models (relational structures too), are used for representing the knowledge about actions in the system: if $\mathcal{L}$ be a language built upon Var and Ag that can be interpreted over epistemic models, then an $\mathcal{L}$ *action model*[2]

[1]From now on, Ag is a finite set of agents and Var is a set of propositional variables.
[2]The language $\mathcal{L}$ is just a parameter.

U is a tuple $(\mathsf{E}, \langle \mathsf{R}_a \rangle_{a \in \mathsf{Ag}}, \mathsf{pre}, \mathsf{sub})$ where $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}\}$ is a *finite* set of actions, $\mathsf{R}_a \subseteq (\mathsf{E} \times \mathsf{E})$ is a relation for each $a \in \mathsf{Ag}$, $\mathsf{pre} : \mathsf{E} \to \mathcal{L}$ is a precondition map assigning a formula $\mathsf{pre}(\mathsf{e}) \in \mathcal{L}$ to each action $\mathsf{e} \in \mathsf{E}$, and $\mathsf{sub} : (\mathsf{E} \times \mathsf{Var}) \to \mathcal{L}$ is a postcondition map assigning a formula $\mathsf{sub}(\mathsf{e}, p) \in \mathcal{L}$ to each atom $p \in \mathsf{Var}$ at each action $\mathsf{e} \in \mathsf{E}$. With respect to the postcondition map, it is required that $\mathsf{sub}(\mathsf{e}, p) \neq p$ only for a finite number of atoms p. From now on, all action models are assumed to be $\mathcal{L}_{\mathsf{LCC}}$ action models.

In order to obtain the $\mathcal{L}_{\mathsf{LCC}}$ language, formulas and programs, respectively, are defined simultaneously with the notion of an $\mathcal{L}_{\mathsf{LCC}}$ action model (i.e. an action model using $\mathcal{L}_{\mathsf{LCC}}$ for its precondition and postcondition functions):

$$\begin{array}{lcl} \varphi & ::= & \top \mid p \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid [\pi]\varphi \mid [\mathsf{U}, \mathsf{e}]\varphi \\ \pi & ::= & a \mid ?\varphi \mid \pi_1 ; \pi_2 \mid \pi_1 \cup \pi_2 \mid \pi^* \end{array}$$

where $p \in \mathsf{Var}$, $a \in \mathsf{Ag}$, U is an $\mathcal{L}_{\mathsf{LCC}}$ action model and e an action in this model.

We establish semantics of $\mathcal{L}_{\mathsf{LCC}}$ throught $||\cdot||^M$ function, that collects the worlds of a given epistemic model M in which a given $\mathcal{L}_{\mathsf{LCC}}$ formula holds or the the pairs of worlds related by a given $\mathcal{L}_{\mathsf{LCC}}$ program: let $M = (W, \langle R_a \rangle_{a \in \mathsf{Ag}}, V)$ be an epistemic model and $\mathsf{U} = (\mathsf{E}, \langle \mathsf{R}_a \rangle_{a \in \mathsf{Ag}}, \mathsf{pre}, \mathsf{sub})$ an action model. The function $||\cdot||^M$, returning both those worlds in W in which an $\mathcal{L}_{\mathsf{LCC}}$ formula holds and those pairs in $W \times W$ in which an $\mathcal{L}_{\mathsf{LCC}}$ program holds, is given by

$$\begin{array}{rlrl} ||\top||^M := & W & ||a||^M := & R_a \\ ||p||^M := & V(p) & ||?\varphi||^M := & \mathsf{Id}_{||\varphi||^M} \\ ||\neg\varphi||^M := & W \setminus ||\varphi||^M & ||\pi_1;\pi_2||^M := & ||\pi_1||^M \circ ||\pi_2||^M \\ ||\varphi_1 \wedge \varphi_2||^M := & ||\varphi_1||^M \cap ||\varphi_2||^M & ||\pi_1 \cup \pi_2||^M := & ||\pi_1||^M \cup ||\pi_2||^M \\ ||[\pi]\varphi||^M := & \{w \in W \mid \forall v((w,v) \in ||\pi||^M \Rightarrow v \in ||\varphi||^M)\} & ||\pi^*||^M := & (||\pi||^M)^* \\ ||[\mathsf{U},\mathsf{e}]\varphi||^M := & \{w \in W \mid w \in ||\mathsf{pre}(\mathsf{e})||^M \Rightarrow (w, \mathsf{e}) \in ||\varphi||^{M \otimes \mathsf{U}}\} & & \end{array}$$

where $\circ$ and * are the composition and the reflexive transitive closure operator, respectively. Notice two special cases for *test*: $||?\bot||^M = \varnothing$ and $||?\top||^M = \mathsf{Id}_W$.

A key feature of this logic is that it characterises the effect of an action model's execution via *reduction axioms*: valid formulas through which it is possible to rewrite a formula with update modalities as an equivalent one without them, thus reducing LCC to PDL and hence providing a compositional analysis for a wide range of informational events. In order to obtain an correct and complete axiom system, is necessary to introduce the program transformer functions: let $\mathsf{U} = (\mathsf{E}, \langle \mathsf{R}_a \rangle_{a \in \mathsf{Ag}}, \mathsf{pre}, \mathsf{sub})$ be an action model with $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}\}$. The *program transformer* T^{U}_{ij} ($i, j \in \{0, \ldots, n-1\}$) on the set of LCC programs is defined as:

$$\blacktriangleright\ T^{\mathsf{U}}_{ij}(a) := \begin{cases} ?\mathsf{pre}(e_i); a & \text{if } \mathsf{e}_i \mathsf{R}_a \mathsf{e}_j \\ ?\bot & \text{otherwise} \end{cases} \qquad \blacktriangleright\ T^{\mathsf{U}}_{ij}(?\varphi) := \begin{cases} ?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U}, \mathsf{e}_i]\varphi) & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases}$$

$$\blacktriangleright\ T^{\mathsf{U}}_{ij}(\pi_1;\pi_2) := \bigcup_{k=0}^{n-1} (T^{\mathsf{U}}_{ik}(\pi_1); T^{\mathsf{U}}_{kj}(\pi_2)) \qquad \blacktriangleright\ T^{\mathsf{U}}_{ij}(\pi_1 \cup \pi_2) := T^{\mathsf{U}}_{ij}(\pi_1) \cup T^{\mathsf{U}}_{ij}(\pi_2)$$

$$\blacktriangleright\ T^{\mathsf{U}}_{ij}(\pi^*) := K^{\mathsf{U}}_{ijn}(\pi)$$

with K^{U}_{ijn} inductively defined as indicated below:

$$\blacktriangleright\ K^{\mathsf{U}}_{ij0}(\pi) := \begin{cases} ?\top \cup T^{\mathsf{U}}_{ij}(\pi) & \text{if } i = j \\ T^{\mathsf{U}}_{ij}(\pi) & \text{otherwise} \end{cases}$$

$$\blacktriangleright\ K^{\mathsf{U}}_{ij(k+1)}(\pi) = \begin{cases} (K^{\mathsf{U}}_{kkk}(\pi))^* & \text{if } i = k = j \\ (K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi) & \text{if } i = k \neq j \\ K^{\mathsf{U}}_{ikk}(\pi); (K^{\mathsf{U}}_{kkk}(\pi))^* & \text{if } i \neq k = j \\ K^{\mathsf{U}}_{ijk}(\pi) \cup (K^{\mathsf{U}}_{ikk}(\pi); (K^{\mathsf{U}}_{kkk}(\pi))^*; K^{\mathsf{U}}_{kjk}(\pi)) & \text{if } i \neq k \neq j \end{cases}$$

The axiom system for LCC, combines the known axiom system of its PDL fragment ([6]) with *recursion* axioms for its action model fragment:

(*taut*) propositional tautologies	(*top*) $[\mathsf{U},\mathsf{e}]\top \leftrightarrow \top$
(*K*) $[\pi](\varphi_1 \to \varphi_2) \to ([\pi]\varphi_1 \to [\pi]\varphi_2)$	(*atm*) $[\mathsf{U},\mathsf{e}]p \leftrightarrow (\mathsf{pre}(\mathsf{e}) \to \mathsf{sub}(\mathsf{e},p))$
(*test*) $[?\varphi_1]\varphi_2 \leftrightarrow (\varphi_1 \to \varphi_2)$	(*neg*) $[\mathsf{U},\mathsf{e}]\neg\varphi \leftrightarrow (\mathsf{pre}(\mathsf{e}) \to \neg[\mathsf{U},\mathsf{e}]\varphi)$
(*seq*) $[\pi_1;\pi_2]\varphi \leftrightarrow [\pi_1][\pi_2]\varphi$	(*conj*) $[\mathsf{U},\mathsf{e}](\varphi_1 \wedge \varphi_2) \leftrightarrow ([\mathsf{U},\mathsf{e}]\varphi_1 \wedge [\mathsf{U},\mathsf{e}]\varphi_2)$
(*choice*) $[\pi_1 \cup \pi_2]\varphi \leftrightarrow [\pi_1]\varphi \wedge [\pi_2]\varphi$	(*prog*) $[\mathsf{U},\mathsf{e}_i][\pi]\varphi \leftrightarrow \bigwedge_{j=0}^{n-1}[T^{\mathsf{U}}_{ij}(\pi)][\mathsf{U},\mathsf{e}_j]\varphi$
(*mix*) $[\pi^*]\varphi \leftrightarrow \varphi \wedge [\pi][\pi^*]\varphi$	(N_{U}) From $\vdash \varphi$ infer $\vdash [\mathsf{U},\mathsf{e}]\varphi$
(*ind*) $\varphi \wedge [\pi^*](\varphi \to [\pi]\varphi) \to [\pi^*]\varphi$	(N_π) From $\vdash \varphi$ infer $\vdash [\pi]\varphi$
(*MP*) From $\vdash \varphi_1$ and $\vdash \varphi_1 \to \varphi_2$ infer $\vdash \varphi_2$	

The crucial reduction axiom is the one characterising the effect of an action model over epistemic PDL programs (*prog*). It is based on the correspondence between action models and finite automata observed in [9]; its main component, the program transformer function T^{U}_{ij}, follows Kleene's translation from finite automata to regular expressions [7]. The present work proposes an alternative definition that uses a matrix treatment of Brzozowski's equational method for obtaining an expression representing the language accepted by a given finite automaton [4, 5]. This alternative definition posses several advantages: first, it have a lower complexity in typical cases (i.e., when the relation differs sufficiently of the Cartesian product), thus allowing more efficient implementations of any LCC-based method; second, the formulas which are obtained are usually much smaller (in the original LCC paper, the size of the transformed formulas of type π^* is always exponential); and third, the matrix treatment presented here is more synthetic, simple and elegant, thus allowing a simpler implementation.

This paper is organized as follows: after this Introduction, Section 2 explains how we can obtain the corresponding expression to each program transformer's type, particularly for Kleene closure through Brzozowski's equational method. Then Section 3 introduces this paper's matrix proposal and discusses the complexity of it. Finally, Section 4 indicates briefly some conclusions.

2 Program transformers through Brzozowski's equations

The new definition of program transformer differs mainly, but not only, on the case for the Kleene closure operator. For every program π a matrix $\mu^{\mathsf{U}}(\pi)$, whose cells are LCC programs, is defined. In this matrix, $\mu^{\mathsf{U}}(\pi)[i, j]$ (the cell in the i^{th} row and j^{th} column) corresponds to the transformation (i.e. the path in M) of π from e_i to e_j (i.e. the path in $M \otimes \mathsf{U}$). The matrix $\mu^{\mathsf{U}}(\pi)$ can be interpreted as the adjacency matrix of a labelled directed graph whose nodes are the actions in E and each edge from e_i to e_j is labelled with the transformation of π from e_i to e_j.

Before presenting the formal definitions, we introduce our method in an informal way. In the following paragraphs we introduce examples of action models and label the edges both with an LCC program and its transformation. Labels have two parts separated by a vertical bar, the left part is the LCC program and the right part is its transformation. For example, the label $a \mid ?\mathsf{pre}(\mathsf{e}_0); a$ from e_0 to e_1 in the agents' diagram indicates that $\mu^{\mathsf{U}}(a)[0, 1] = ?\mathsf{pre}(\mathsf{e}_0); a$.

Agents Suppose that U contains an edge from e_i to e_j labelled with agent a. Let w be an state in an epistemic model M. What do we have to test in order to ensure that, after executing $(\mathsf{U}, \mathsf{e}_i)$ over (M, w), an a-path from (w, e_i) to some state (w', e_j) will persist $M \otimes \mathsf{U}$? First, we need to test in M that e_i is executable in w, an then that an a-path exists from w to w'. Trivially, if there is no a-path from e_i to e_j in U, then the transformation of a is $?\bot$, that is, $\mu^{\mathsf{U}}(a)[i, j] = ?\bot$.

$$\begin{tikzcd}\mathsf{e}_0 \arrow[r, "a \mid ?\mathsf{pre}(\mathsf{e}_0); a"] & \mathsf{e}_1\end{tikzcd}$$

Test The transformation of a test from some e_i to itself is just a test. But the test has two parts, because in order to test $?\varphi$ in (w, e_i) we should first test that $\mathsf{pre}(\mathsf{e}_i)$ is true in w. Then, as the execution of action e_i may change the valuation function, what we test in w is not just φ but rather $[\mathsf{U}, \mathsf{e}_i]\varphi$. The transformation of a test from some state to a different one is always $?\bot$.

$$\begin{tikzcd}\mathsf{e}_i \arrow[loop above, "?\varphi \mid ?(\mathsf{pre}(\mathsf{e}_i) \land [\mathsf{U}, \mathsf{e}_i]\varphi)"]\end{tikzcd}$$

Non-deterministic choice The transformation of the choice $\pi_1 \cup \pi_2$ is the choice of the transformations of both programs π_1 and π_2. But, as the choice of some program π with $?\bot$ is equivalent to π, we can simplify some cases. In the diagram, the transformations of some π_1 and π_2 are labelled with dashed lines. When such a line between two nodes does not exist, the transformation should be $?\bot$. Labels of the form S_i^{jk} represent LCC programs corresponding to the transformation of π_i from e_j to e_k. The transformations of $\pi_1 \cup \pi_2$ are shown with continuous lines.

$$\begin{tikzcd}[column sep=huge]
\mathsf{e}_2 & \mathsf{e}_0 \arrow[l, "\pi_1 \cup \pi_2 \mid S_1^{02}"'] \arrow[l, dashed, bend right, "\pi_1 \mid S_1^{02}"'] \arrow[r, "\pi_1 \cup \pi_2 \mid S_1^{01} \cup S_2^{01}"] \arrow[r, dashed, bend left, "\pi_1 \mid S_1^{01}"] \arrow[r, dashed, bend right, "\pi_2 \mid S_2^{01}"'] & \mathsf{e}_1
\end{tikzcd}$$

Sequential composition State e_k is reachable from e_i through the concatenation of π_1 and π_2 iff there is some intermediate state e_j that is reachable from e_i through a π_1-path and there is a π_2-path from e_j to e_k. But there can be different intermediate states with these properties (e_1 and e_2 in the example below). So the transformation of the concatenation $\pi_1;\pi_2$ is the choice of all the different possible concatenations (see below).

$$\begin{tikzcd}[column sep=huge]
 & \mathsf{e}_1 \arrow[dr, dashed, "\pi_2 \mid S_2^{13}"] & \\
\mathsf{e}_0 \arrow[ur, dashed, "\pi_1 \mid S_1^{01}"] \arrow[dr, dashed, "\pi_1 \mid S_1^{02}"'] \arrow[rr, "\pi_1;\pi_2 \mid (S_1^{01};S_2^{13}) \cup (S_1^{02};S_2^{23})"] \arrow[rr, dashed, bend left=10] & & \mathsf{e}_3 \\
 & \mathsf{e}_2 \arrow[ur, dashed, "\pi_2 \mid S_2^{23}"'] &
\end{tikzcd}$$

Up to now, we proceed in a very similar way to that of original program transformer definition, just simplifying some trivial cases like $\pi \cup ?\bot$, which is reduced to π. The main novelty of our transformation is with Kleene closure. We use a method proposed by Brzozowski [4], presented here in a matrix format.

Kleene closure The following graph will be used to illustrate the creation of the transformations of π^* given those of π. This is where our program transformers are substantially different from those in [10].

$$\begin{tikzcd}[column sep=huge]
\mathsf{e}_0 \arrow[r, bend left, "\pi \mid S^{01}"] & \mathsf{e}_1 \arrow[l, bend left, "\pi \mid S^{10}"] \arrow[loop below, "\pi \mid S^{11}"] & \mathsf{e}_2 \arrow[l, "\pi \mid S^{21}"'] \arrow[loop below, "\pi \mid S^{22}"]
\end{tikzcd}$$

In the above graph, labels S^{ij} represent the transformations of π from e_i to e_j (when there is no arrow between two —equal or different— nodes, it is assumed that the corresponding program is $?\bot$). In order to find the labels X^{ij} for the transformations of π^*, we follow an equational method first proposed by Brzozowski [4]. Observe, for example, how a π^*-path from e_1 to e_0 might start with S^{10} (an instance of π from e_1 to e_0) and then continue with X^{00} (an instance of π^* from e_0 to e_0), but it might also start with S^{11} (an instance of π from e_1 to e_1) and then continue with X^{10} (an instance of π^* from e_1 to e_0). In this case, these are the only two possibilities, and they can be represented by the following equation:

$$X^{10} = (S^{10};X^{00}) \cup (S^{11};X^{10}) \qquad (1)$$

The equations for X^{00} and X^{20} can be obtained in a similar way:

$$X^{00} = ?\mathsf{pre}(\mathsf{e}_0) \cup (S^{01}; X^{10}) \quad (2)$$

$$X^{20} = (S^{22}; X^{20}) \cup (S^{21}; X^{10}) \quad (3)$$

This yields an equation system of LCC programs with X^{00}, X^{10} and X^{20} as its only variables. Observe how, in (2), $?\mathsf{pre}(\mathsf{e}_0)$ indicates that a possible π^*-path from e_0 to e_0 is to do nothing, but the transformation should check whether e_0 is executable at the target state; hence the test $?\mathsf{pre}(\mathsf{e}_0)$.

To solve the above system we proceed by substitution using properties of Kleene algebra [8], such as associative and distributive properties of the operators. First, we can use (2) to replace X^{00} in (1):

$$\begin{aligned} X^{10} &= (S^{10}; (?\mathsf{pre}(\mathsf{e}_0) \cup (S^{01}; X^{10}))) \cup (S^{11}; X^{10}) = (S^{10}; ?\mathsf{pre}(\mathsf{e}_0)) \cup \\ &\cup (S^{10}; S^{01}; X^{10}) \cup (S^{11}; X^{10}) = (S^{10}; ?\mathsf{pre}(\mathsf{e}_0)) \cup (((S^{10}; S^{01}) \cup S^{11}); X^{10}) = \quad (4) \\ &= ((S^{10}; S^{01}) \cup S^{11})^*; S^{10}; ?\mathsf{pre}(\mathsf{e}_0) \end{aligned}$$

The last equality uses Arden's Theorem [1]: $X = B \cup (A; X)$ implies $X = A^*; B$.

Now, we use (4) to substitute X^{10} in (2):

$$X^{00} = ?\mathsf{pre}(\mathsf{e}_0) \cup (S^{01}; ((S^{10}; S^{01}) \cup S^{11})^*; S^{10}; ?\mathsf{pre}(\mathsf{e}_0)) \quad (5)$$

Finally, we substitute X^{10} in (3) and apply Arden's Theorem to solve X^{20}:

$$\begin{aligned} X^{20} &= (S^{22}; X^{20}) \cup (S^{21}; ((S^{10}; S^{01}) \cup S^{11})^*; S^{10}; ?\mathsf{pre}(\mathsf{e}_0)) = \quad (6) \\ &= (S^{22})^*; S^{21}; ((S^{10}; S^{01}) \cup S^{11})^*; S^{10}; ?\mathsf{pre}(\mathsf{e}_0) \end{aligned}$$

After solving these equations, each X^{00}, X^{10} and X^{20} represents a transformed π^*-path from e_0, e_1 and e_2 to e_0, respectively.

By using a matrix calculus similar to that in chapter 3 of [5] we calculate all X^{ij} in parallel and thus avoid repeating the process for each destination node. This method is more synthetic, clear, elegant and allows a simpler implementation that the original LCC paper. The following section will present the formal definition of the matrix calculus; now, we just introduce the basis of the matrix calculus. The equations used above can be represented in the following matrix:

	e_0	e_1	e_2	e_0	e_1	e_2
e_0	$?\bot$	S^{01}	$?\bot$	$?\mathsf{pre}(\mathsf{e}_0)$	$?\bot$	$?\bot$
e_1	S^{10}	S^{11}	$?\bot$	$?\bot$	$?\mathsf{pre}(\mathsf{e}_1)$	$?\bot$
e_2	$?\bot$	S^{21}	S^{22}	$?\bot$	$?\bot$	$?\mathsf{pre}(\mathsf{e}_2)$

The left part contains the π-paths from one node (row) to another one (column). It is an accessibility matrix for the π-graph above. Call $\mu^{\mathsf{U}}(\pi)[i, j]$ the cell corresponding to row e_i and column e_j in this left part and $A^{\mathsf{U}}[i, j]$ the cell with the same position at the right part. Observe that $A^{\mathsf{U}}[i, j] = ?\mathsf{pre}(\mathsf{e}_i)$ if $i = j$ and $?\bot$ otherwise. We may check that the equations for X^{ij} that we created above looking at the π-graph can be created now by:

$$X^{ij} = (\mu^{\mathsf{U}}(\pi)[i, 0]; X^{0j}) \cup (\mu^{\mathsf{U}}(\pi)[i, 1]; X^{1j}) \cup (\mu^{\mathsf{U}}(\pi)[i, 2]; X^{2j}) \cup A^{\mathsf{U}}[i, j] \quad (7)$$

The greatest advantage of working with matrices is that we can transform several equations at the same time by working in a row. Look at the following matrix, which is the result of applying Arden's Theorem to the e_1 row.

	e_0	e_1	e_2	e_0	e_1	e_2
e_0	$?\bot$	S^{01}	$?\bot$	$?\mathsf{pre}(\mathsf{e}_0)$	$?\bot$	$?\bot$
e_1	$(S^{11})^*;S^{10}$	$?\bot$	$?\bot$	$?\bot$	$(S^{11})^*;?\mathsf{pre}(\mathsf{e}_1)$	$?\bot$
e_2	$?\bot$	S^{21}	S^{22}	$?\bot$	$?\bot$	$?\mathsf{pre}(\mathsf{e}_2)$

The transformation consists on replacing the position $[\mathsf{e}_1, \mathsf{e}_1]$ in the left part with $?\bot$ and concatenate $(S^{11})^*$ with the others cells in the row.

Not only Arden's Theorem, but also the substitution operation can be done in parallel using matrix operations. Look at the following matrix:

	e_0	e_1	e_2	e_0	e_1	e_2
e_0	$?\bot$	S^{01}	$?\bot$	$?\mathsf{pre}(\mathsf{e}_0)$	$?\bot$	$?\bot$
e_1	$(S^{11})^*;S^{10}$	$?\bot$	$?\bot$	$?\bot$	$(S^{11})^*;?\mathsf{pre}(\mathsf{e}_1)$	$?\bot$
e_2	$(S^{21};(S^{11})^*;S^{10})$	$?\bot$	S^{22}	$?\bot$	$(S^{21};(S^{11})^*;?\mathsf{pre}(\mathsf{e}_1))$	$?\mathsf{pre}(\mathsf{e}_2)$

The above matrix has been obtained from the previous one by applying one substitution that has converted the left $[\mathsf{e}_2, \mathsf{e}_1]$ position into $?\bot$ and each (left/right) position $[\mathsf{e}_2, \mathsf{e}_i]$ (different to the left $[\mathsf{e}_2, \mathsf{e}_1]$ position) contains now a program with the form $(B; C) \cup D$, where B is always the previous program in the left $[\mathsf{e}_2, \mathsf{e}_1]$ position (always S^{21}, in this case), C is the program in the (resp. left/right) $[\mathsf{e}_1, \mathsf{e}_i]$ position (that is, the program in the same column and the above row) and D is the previous program in the position being modified.

3 A matrix calculus for program transformation

In this section we introduce the formal definitions of our matrix calculus.

Definition 3.1 (Program transformation matrix)**.** Let $\mathsf{U} = (\mathsf{E}, \mathsf{R}, \mathsf{pre}, \mathsf{sub})$ be an action model with $\mathsf{E} = \{\mathsf{e}_0, \ldots, \mathsf{e}_{n-1}\}$. The function $\mu^{\mathsf{U}} : \Pi \to \mathcal{M}_{n\times n}$, with Π the set of LCC programs and $\mathcal{M}_{n\times n}$ the class of n-square matrices, takes an LCC program π and returns a n-square matrix $\mu^{\mathsf{U}}(\pi)$ in which each cell $\mu^{\mathsf{U}}(\pi)[i, j]$ is an LCC program representing the transformation of π from e_i to e_j in the sense of the program transformers $T^{\mathsf{U}}_{ij}(\pi)$ of [10]. The recursive definition of $\mu^{\mathsf{U}}(\pi)$ is:

- *Agents*:
$$\mu^{\mathsf{U}}(a)[i, j] := \begin{cases} ?\mathsf{pre}(\mathsf{e}_i); a & \text{if } \mathsf{e}_i \mathsf{R}_a \mathsf{e}_j \\ ?\bot & \text{otherwise} \end{cases} \tag{8}$$
- *Test*:
$$\mu^{\mathsf{U}}(?\varphi)[i, j] := \begin{cases} ?(\mathsf{pre}(\mathsf{e}_i) \wedge [\mathsf{U}, \mathsf{e}_i]\varphi) & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases} \tag{9}$$
- *Non-deterministic choice*:
$$\mu^{\mathsf{U}}(\pi_1 \cup \pi_2)[i, j] := \bigoplus \left\{\mu^{\mathsf{U}}(\pi_1)[i, j],\ \mu^{\mathsf{U}}(\pi_2)[i, j]\right\} \tag{10}$$
where $\bigoplus\Gamma$ is the non-deterministic choice of the programs in Γ set after removing occurrences of $?\bot$, that is (being "$\bigcup$" the generalised non-deterministic choice ("$\cup$") of a program non-empty set),

$$\bigoplus\Gamma := \begin{cases} \bigcup(\Gamma \setminus \{?\bot\}) & \text{if } \varnothing \neq \Gamma \neq \{?\bot\} \\ ?\bot & \text{otherwise} \end{cases} \tag{11}$$

- *Sequential composition*:

$$\mu^{\mathsf{U}}(\pi_1;\pi_2)[i,j] := \bigoplus\left\{\mu^{\mathsf{U}}(\pi_1)[i,k] \odot \mu^{\mathsf{U}}(\pi_2)[k,j] \;\mid\; 0 \leq k \leq n-1\right\} \tag{12}$$

  where $\sigma \odot \rho$ is the sequential composition of σ and ρ after removing superfluous occurrences of $?\bot$ and $?\top$, that is,

$$\sigma \odot \rho := \begin{cases} \sigma;\rho & \text{if } \sigma \neq ?\bot \neq \rho \text{ and } \sigma \neq ?\top \neq \rho \\ \sigma & \text{if } \sigma \neq ?\top = \rho \\ \rho & \text{if } \sigma = ?\top \\ ?\bot & \text{otherwise} \end{cases} \tag{13}$$

- *Kleene closure*:

$$\mu^{\mathsf{U}}(\pi^*) := S_0^{\mathsf{U}}\left(\mu^{\mathsf{U}}(\pi) \mid A^{\mathsf{U}}\right) \tag{14}$$

  where $\mu^{\mathsf{U}}(\pi) \mid A^{\mathsf{U}}$ is the $n \times 2n$ matrix obtained by augmenting $\mu^{\mathsf{U}}(\pi)$ with A^{U}, an $n \times n$ matrix defined as

$$A^{\mathsf{U}}[i,j] := \begin{cases} ?\mathsf{pre}(\mathsf{e}_i) & \text{if } i = j \\ ?\bot & \text{otherwise} \end{cases} \tag{15}$$

  The function S_k^{U} (with $0 \leq k \leq n$), defined as

$$S_k^{\mathsf{U}}(M \mid A) := \begin{cases} A & \text{if } k = n \\ S_{k+1}^{\mathsf{U}}(Subs_k(Ard_k(M \mid A))) & \text{otherwise} \end{cases} \tag{16}$$

  receives an argument $M \mid A$ and performs an iterative process applying Arden's Theorem to row k (via function $Ard_k : \mathcal{M}_{n\times 2n} \to \mathcal{M}_{n\times 2n}$) and substituting rows different from k (via function $Subs_k : \mathcal{M}_{n\times 2n} \to \mathcal{M}_{n\times 2n}$) until a $k = n$, then returning the right part of the augmented matrix. The two auxiliary functions, Ard_k and $Subs_k$, are given by

$$Ard_k(N)[i,j] \;:=\; \begin{cases} N[i,j] & \text{if } i \neq k \\ ?\bot & \text{if } i = k = j \\ N[i,j] & \text{if } i = k \neq j \text{ and } N[k,k] = ?\bot \\ N[k,k]^* \odot N[i,j] & \text{otherwise} \end{cases} \tag{17}$$

$$Subs_k(N)[i,j] \;:=\; \begin{cases} N[i,j] & \text{if } i = k \\ ?\bot & \text{if } i \neq k = j \\ \bigoplus\{N[i,k] \odot N[k,j], N[i,j]\} & \text{otherwise} \end{cases} \tag{18}$$

Now it is possible to substitute the previous version of the crucial reduction axiom by the following: $[\mathsf{U},\mathsf{e}_i][\pi]\varphi \;\leftrightarrow\; \bigwedge_{\substack{0\leq j\leq n-1 \\ \mu^{\mathsf{U}}(\pi)[i,j]\neq ?\bot}} [\mu^{\mathsf{U}}(\pi)[i,j]][\mathsf{U},\mathsf{e}_j]\varphi$

3.1 Complexity

The program transformers in [10] require exponential time due to the use of Kleene's method [7]; moreover, the size of the transformed formulas of type π^* is also exponential because of the definition of $K^{\cup}_{ijn}$. The advantage of our transformers is that they can be executed in polynomial time in cases different to the worst one[3]; moreover, in typical cases our method generates much smaller expressions.

Let $M = (W, \langle R_a \rangle_{a \in \mathsf{Ag}}, V)$ be an epistemic model with $card(W) = n$. If M is a complete model, then the number of operators in $\mu^{\cup}(\pi^*)[n-1, 0]$ is in the order of 2^{2n} (i.e. our transformers produce an exponential output), which implies that the required time is also exponential. If M is a *chain model*[4], then the number of operators in $\mu^{\cup}(\pi^*)[n-1, 0]$ is in the order of $2n^2$ (i.e. in this case the length of the output is polynomial), thus the required time is also polinomial.

An analysis of the operations involved in computing $\mu^{\cup}(\pi)$ for the different kinds of programs π show that, while Kleene's method forces the program transformers to use an exponential number of operations, our proposal uses only a polynomial number of matrix operations: if π is an agent, a test or a non-deterministic choice, then the whole matrix $\mu^{\cup}(\pi)$ can be computed in $O(n^2)$; but, if π is a sequential composition or a Kleene closure, then $\mu^{\cup}(\pi)$ is computed in $O(n^3)$.

Just, the last is the crucial case; therefore it is analyzed with more detail. With $\mu^{\cup}(\pi)$ given, computing $\mu^{\cup}(\pi^*)$ requires first to build $(\mu^{\cup}(\pi) \mid A^{\cup})$ and then n iterations of $S^{\cup}_k$ (see (14)). Note that the size of $(\mu^{\cup}(\pi) \mid A^{\cup})$ is $n \times 2n$ and to build each cell requires a constant number of operations. So building the initial matrix is in $O(n^2)$. Now, each one of the n calls to $S^{\cup}_k$ is in $O(n^2)$, as Ard_k (see (17)) only changes the cells in row k, $Subs_k$ only the cells in the other rows, and each cell can be modified in constant time. So the n calls to $S^{\cup}_k$ are computed in $O(n^3)$. If the matrices for subprograms are not given and g is the number of subprograms in π, building $\mu^{\cup}(\pi)$ from scratch requires a number of matrix operations in $O(g \cdot n^3)$.

3.2 Possible improvements

The operators "$\oplus$" and "$\odot$" used in the previous definition are versions, respectively, of non-deterministic choice and sequential composition that remove unnecessary occurrences of $?\bot$ and $?\top$; thus returning programs that are (potentially) syntactically shorter but nevertheless semantically equivalent to their PDL counterparts "$\cup$" and "$;$". The $\odot$ operation's definition can be modified to obtain even simpler expressions. For example, $\sigma \odot \rho$ might be defined as σ if $\sigma \neq ?\top = \rho$ and as ρ if $\sigma = ?\top$. Moreover, the algorithm implementing Ard_k and $Subs_k$ functions can be improved by disregarding the $N[i, j]$ elements with $j < k$ or $j > n+k$ (being $N[i, j]$ a $n \times 2n$ matrix), since those are necessarily equal to $?\bot$. These changes,

[3]The worst cases are *complete* models, i.e. fully connected models.

[4]A chain model is a model with a linear order relation.

despite not lowering the translation's complexity order, would nevertheless make it more efficient.

4 Conclusions

This work presents an alternative definition of the program transformers, which are used in the crucial axiom of the axiom system for LCC. This system allows us to reduce LCC to PDL. The proposal uses a matrix treatment of Brzozowski's equational method in order to obtain a regular expression representing the language accepted by a finite automaton. While Brzozowski's method and that used in the original LCC paper [10] are equivalent, the first is computationally more efficient in typical cases and generates much smaller expressions. Moreover, the matrix treatment presented here is more synthetic, clear and elegant, thus allowing a simpler implementation.

References

[1] D. N. Arden. Delayed-logic and finite-state machines. In *SWCT (FOCS)*, pages 133–151. IEEE Computer Society, 1961.

[2] A. Baltag and L. S. Moss. Logics for epistemic programs. *Synthese*, 139(2):165–224, 2004.

[3] A. Baltag, L. S. Moss, and S. Solecki. The logic of public announcements and common knowledge and private suspicions. In I. Gilboa, editor, *TARK*, pages 43–56, San Francisco, CA, USA, 1998. Morgan Kaufmann.

[4] J. A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964.

[5] J. H. Conway. *Regular Algebra and Finite Machines*. Chapman and Hall, 1971.

[6] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, Cambridge, MA, 2000.

[7] S. Kleene. Representation of events in nerve nets and finite automata. In C. E. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, NJ, 1956.

[8] D. Kozen. On kleene algebras and closed semirings. In B. Rovan, editor, *MFCS*, volume 452 of *Lecture Notes in Computer Science*, pages 26–47. Springer, 1990.

[9] J. van Benthem and B. Kooi. Reduction axioms for epistemic actions. In R. Schmidt, I. Pratt-Hartmann, M. Reynolds, and H. Wansing, editors, *Advances in Modal Logic (Number UMCS-04-09-01 in Technical Report Series)*, pages 197–211. Department of Computer Science, University of Manchester, 2004.

[10] J. van Benthem, J. van Eijck, and B. Kooi. Logics of communication and change. *Information and Computation*, 204(11):1620–1662, 2006.

MEREOLOGY AND TEMPORAL STRUCTURES

Pedro González Núñez
Universidad de Sevilla, Andalucía Tech
pegnunez@gmail.com

Abstract

We will propose a semantics for first order multimodal logics combining necessity and temporal operators, which intends to reflect some philosophical insights brought up when discussing certain topics in ontology. Namely, our proposal will allow us to construe a primarily semantic notion of part, and specifically of temporal and spatial part, which will in turn enable an easy and intuitive way of modeling some metaphysical claims, and also set a solid framework to develop multimodal logics including temporal and non-temporal operators.

1 Introduction

How to best conceive change and the notion of the flow of time are questions that have an illustrious lineage in the history of philosophy. In contemporary literature there are quite a handful of intertwined topics that are deployed when discussing conceptions of time in relation with identity and constitution. We will present a model theory [1] that is tailored to let us easily model some of these topics, which include, but are not limited to, the use of the notion of "temporal part" to explain maintenance of identity through change, and some approach to the idea that there is a notion of "loose" or "vague" identity that is used in ordinary speech that should be differentiated from the proper notion of ("strict") identity".

To achieve this, we are going to make some strong assumptions, necessary to increase the manageability of the philosophical notions we will try to formalize. First, we shall adopt the full law of Leibniz –i.e. both the principle of identity of the indiscernibles and the principle of indiscernibility of the identicals– as a given. This amounts to assuming that our language will be expressive enough for the purposes of distinguishing the objects of our theories. Therefore it can be

[1]The basic structure for the treatment of time is heavily based on the so-called Kamp frames, and in a variation on bundled tree structures found in [7]. More systematic work on the relation between Kamp frames and bundled trees can be found in [12].

understood as a way to limit the scope of our task rather than as an unjustified metaphysical claim.

Secondly, as an application of the aforementioned principles we shall understand that, for purpose of our models, it suffices for an object a to be a part of another object b at some point of time and region of space, that they are undistinguishable in that point and region. Although if understood as a philosophical claim it would require an independent argument to be justified, we can take this to be, like the one above, a methodological constraint we impose to ourselves.

Thirdly, we are going to adopt the following semantical stance: the truth of the attribution of some predicate P to some term t at some state w is to be understood in a different way than the standard treatment. Instead, we shall define the denotation of a predicate symbol as an element of some non-empty set, we shall understand a "property" as the relativisation of the denotation of a predicate-symbol (say, P) to a world (say, w), that is, as the pair $\langle ||P||, w \rangle$. Then, we shall understand the denotation of individual terms as sets of "properties" in the mentioned sense, and say that Pt is true in a world if the pair $\langle ||P||, w \rangle$ is an element of the denotation of t. Thus, with these guiding notions of basing parthood in sameness of "properties", and construing individuals as sets of "properties" [2] we will be able to draw semantical notions of parthood in terms of set-theoretical inclusion.

2 Temporal structures: language and semantics

Definition 1 (Language)**.** The formulas of our basic language are defined by the following grammar:

$\phi ::= Pt \mid \neg\phi \mid \phi \wedge \phi \mid \langle F \rangle\phi \mid \langle P \rangle\phi \mid \langle S \rangle\phi \mid \forall x\phi \mid t_1 = t_2$

where t is either an individual constant or variable.

As can be seen, we are limiting ourselves to monadic predicate-letters.

The symbols of the form $\langle * \rangle$ are to be understood as "possibility" operators. $\langle P \rangle$ is to be read as "at some point in the past"; $\langle F \rangle$ is to be read as "at some point of the future"; $\langle S \rangle$ will be given two alternate lectures depending on the kind of discourse we are representing: we can either speak of alternative or counterfactual histories, in which case it will be understood as "at this time in some other history", or we can speak of states of affairs in different regions of space, in which case it will be read as "in some spatial region". We will write $[*]$ for the dual of any modal operator $\langle * \rangle$. We shall also use the usual truth-functional conectives definable by means of negation and conjunction, and the existential quantifier defined by means of the universal quantifier and negation.

[2]Let us maintain the quotation marks as a reminder that here property has a meaning quite different than in usual formal semantics.

Definition 2 (Parallel histories frame)**.** A parallel history frame is a 5-tuple $F = \langle W, \prec, \approx, U, D \rangle$, where

- W is a non-empty set
- $\prec$ is a binary relation on W such that for all $x, y, z, \in W$
  - $\forall xyz(x \prec y \wedge y \prec z \rightarrow x \prec z)$,
  - $\forall xy \neg(x \prec y \wedge y \prec x)$,
  - $\forall xyz(x \prec y \wedge x \prec z \rightarrow (y \prec z \vee y = z \vee z \prec y))$,
  - $\forall xyz(y \prec x \wedge z \prec x \rightarrow (y \prec z \vee y = z \vee z \prec y))$,
- $\approx$ is an equivalence relation such that for all $x, y, z, w, \in W$
  - $\forall xy(x \approx y \rightarrow \neg x \prec y)$,
  - $\forall xyz(x \prec y \wedge x \approx z \rightarrow \exists w(w \approx y \wedge z \prec w))$,
  - $\forall xy(\neg(x \prec y \vee y \prec x \vee x = y) \rightarrow \exists z((x \prec z \vee z \prec x \vee x = z) \wedge z \approx y))$
- U is a non-empty set such that $W \bigcap U = \emptyset$
- D is a non-empty set such that $D \subseteq \wp(U \times W)$

Additionally, we shall say that $w \sim v$, or informally, that w and v belong to the same history, whenever $w \prec v$, $v \prec w$ or $v = w$

Relation $\prec$ is constructed as an earlier-later relation as usual in Kamp frames [3]. The relation $\approx$, however is a variation on the one used in Kamp frames. The motivation for the variation is achieving a notion of necessity which check for all histories in the model rather than only those with a common past up to the point of evaluation, which is the norm in renderings of "historical" necessity.

Definition 3 (Parallel histories model)**.** A parallel histories model is a tuple $M = \langle W, \prec, \approx, U, D, I \rangle$, where $W, U, D, \prec$ and $\approx$ are as above and I is an interpretation function that assigns elements of D to individual constants, and elements of U to predicate symbols. We shall also have an assignment a which assigns elements of D to the individual variables of our language. Note that we do not impose any restrictions on the atoms that are true in worlds related by $\approx$.

Definition 4 (Semantic rules)**.** The rules that govern the semantic interpretation of our language, denoted by $|| * ||$, are as follows:

- For any constant symbol c, $||c||_{M,w,a} = I(c)$

[3]We follow [6], page 664 in our understanding of Kamp structures.

- For any predicate symbol P, $||P||_{M,w,a} = I(P)$
- For any individual variable x, $||x||_{M,w,a} = a(x)$
- For any term t and predicate-symbol P, $||Pt||_{M,w,a} = 1$ iff $\langle ||P||_{M,w,a}, w\rangle \in ||t||_{M,w,a}$
- For any terms, t_1 and t_2, $||t_1 = t_2||_{M,w,a} = 1$ iff $||t_1||_{M,w,a} = ||t_2||_{M,w,a}$

  For any well-formed formulae φ and ψ:
- $M, w, a \models \neg\varphi$ syss $M, w, a \not\models \neg\varphi$
- $M, w, a \models \varphi \wedge \psi$ syss $M, w, a \models \varphi$ and $M, w, a \models \psi$
- For any variable x $M, w, a \models \forall x\varphi$ iff every assignment a' differing from a in at most the value of $a'(x)$ is such that $M, w, a' \models \varphi$
- $M, w, a \models \langle F\rangle\varphi$ iff exists v such that $w \prec v$ and $M, v, a \models \varphi$
- $M, w, a \models \langle P\rangle\varphi$ iff exists v such that $v \prec w$ and $M, v, a \models \varphi$
- $M, w, a \models \langle S\rangle\varphi$ iff exists v such that $v \approx w$ and $M, v, a \models \varphi$

These semantic rules let us wiev, as stated in the introduction, each individual as the set of "properties" which hold true of it. That is why we have limited ourselves to unary predicates, since we cannot set such an straightforward way of construing the interpretation of individual terms as being the set of interpretations of both the monadic and poliadic relations which are true of them, for the latter would involve the interpretation of some other individual terms. Trying to find a relatively homogeneus treatment for poliadic relations is an objective set for future investigations.

With this we have a semantics that enforces both Leibniz principles precisely due to how we have set the semantics for terms, according to our described aim. Beyond that, the changes in atomic sentences interpretation have no further implications in terms of validities. The most salient feature of the temporal structure is that every "history" is enforced to have the same topology, so to speech, so that for every moment of every history, there is an equivalence class of moments that are "simultaneous" to it. That enables another interpretation for the modality $\langle S\rangle$, as said above: instead of having alternative histories, we can view the structure as modeling simultaneous histories, i.e. histories of different regions of space. This is crucial, given that once having the notion of temporal part we can, with some modifications, either use it to represent counterfactual discourse concerning objects and their temporal parts, or discourse about the interaction of objects and both their temporal and spatial parts within the same history. Let's then define such notions.

3 Mereology: notions of parthood and some applications

Below we shall introduce new primitive relational symbols for parthood relationships. In order to facilitate the formulation of the semantic rules, we shall introduce an auxiliary definition.

Definition 5 (Reduction of an individual to a world)**.** Given an individual $d \in D$ and a world $v \in W$ we shall define the reduction of d to v as

$$red(d, w_1) = \{\langle u, w_i \rangle \in d \mid w_i = w_1\}$$

With this set-theoretical device, our mereological basic relations can be defined as follows.

Definition 6 (Spatial parthood relation)**.** Let us add to our basic language a binary relation symbol with interfixed notation $\sqsubseteq^S$ to denote the notion '...'is an (improper) spatial part of...". The semantic rule for this symbol is as follows:

$M, w, a \models t_1 \sqsubseteq^S t_2$ iff:

$$\cup\{red(||t_1||_{M,w,a}, v)|v \approx w\} \subseteq \cup\{red(||t_2||_{M,w,a}, v)|v \approx w\}, \text{ and } red(||t_1||_{M,w,a}, w) \neq \emptyset$$

This says that some object a is a spatial part of b at some point of space and time w if, provided that a has some property at w, then the reduction of a to the spatio-temporal locations simultaneous with w is a subset of the respective reduction of b. That is, the relation holds whenever the two individuals are indistiguishable from each other at any state simultanous with the point of evaluation.

Definition 7 (Temporal parthood in counterfactual contexts)**.** Let us add to our basic language a binary relation symbol with interfixed notation $\sqsubseteq^{T1}$ to denote the notion "... is an (improper) temporal part of...". The semantic rule for this symbol is as follows:

$M, w, a \models t_1 \sqsubseteq^{T1} t_2$ iff:

$$\cup\{red(||t_1||_{M,w,a}, v)|v \sim w\} \subseteq \cup\{red(||t_2||_{M,w,a}, v)|v \sim w\}, \text{ and } red(||t_1||_{M,w,a}, w) \neq \emptyset$$

This says that some object a is a temporal part (in a couterfactual context) of b at some point of space and time w if, provided that a has some property at w, then the reduction of a to the states in the history to which w belongs is a subset of the respective reduction of b. That is, the relation holds whenever the two individuals are indistiguishable from each other at any state in the same history as the point of evaluation.

Now, it is noteworthy that while the notion of spatial part as defined is intuitive when the frame is interpreted as an spatio-temporal frame, that intuitive interpretation is lost when we switch to the counterfactual histories interpretation of the semantics. In a similar way, while the notion of some object being a temporal part of another in a given history, whereas in some other history it might not be, is perfectly intelligible, the notion of being a temporal part in a region of space while possibly not being so in another defeats the notion of temporal part under that interpretation: a is a temporal part of b at time t if and only if a is indistinguishable with b at that time, everywhere. While the relation of spatial parthood is variable from time to time, the relation of temporal parthood is constant from time to time and from place to place. Thus, an alternative should be construed to represent the notion of temporal parthood when our language and semantics are given the spatio-temporal interpretation.

Definition 8 (Temporal parthood (in spatio-temporal contexts))**.** We yet again extend our language with a second temporal parthood relation, expressed with the symbol $\sqsubseteq^{T2}$, interpreted as follows:
$M, w, a \models t_1 \sqsubseteq^{T2} t_2$ iff:

$$\{\cup\{red(||t_1||_{M,w,a}, v) \mid v \approx w'\} \mid \cup\{red(||t_1||_{M,w,a}, v)\} \neq \emptyset, w' \sim w\} \subseteq$$

$$\subseteq \{\cup\{red(||t_2||_{M,w,a}, v) \mid v \approx w'\} \mid \cup\{red(||t_1||_{M,w,a}, v)\} \neq \emptyset, w' \sim w\},$$

$$\text{and } red(||t_1||_{M,w,a}, w) \neq \emptyset$$

.

This definition is a bit more complex, but not too much. It says that an object a will in this sense be a temporal part of another object b iff, by considering the unions of reductions of a respect to worlds simultanous to some world w', and collecting the set of all such unions such that they are non-empty and w' is in the history the point of evaluation belongs to, and doing the same with respect to b, it happens that the set of such unions of reductions of a is included in the set of said unions of reductions of b. This means that if a exists at some time, then a is indistingueshable from b in any region and time in which some property hods true of a.

It can be shown that these relations are transitive and reflexive, while for the spatial and temporal in confterfactual context parthood relations their modal nature makes antisymmetry fail. (I will subsequently omit superindexes of our parthood symbols when the matter discussed is equally appliable to the three of them, or the reference to one or another is clear from the context). While $a \sqsubseteq^i b \wedge b \sqsubseteq^i b \rightarrow a = b$ when $i \in \{S, T1\}$ is not valid in every parallel histories frame, a modal version of it is so, different for each of the two problematic parthood relationships:

$$[P](a \sqsubseteq^S b \land b \sqsubseteq^S a) \land [F](a \sqsubseteq^S b \land b \sqsubseteq^S a) \rightarrow a = b$$

$$[S](a \sqsubseteq^{T1} b \land b \sqsubseteq^{T1} a) \rightarrow a = b$$

In general, it should be observed that adaptation of mereological axions for S- and T1-parthood should replace occurances of identity with local equality or indiscernibility, which can be defined as reciprocal parthood ($a \sqsubseteq b \land b \sqsubseteq a$). This can be understood by examinating the usual definion of "proper parthood" which we should symbolise $\sqsubset$. If we adopt the usual definition $x \sqsubset y =_{def} x \sqsubseteq y \land \neg(x = y)$, then we are putting forward a needlessly weak condition: In order to, say, recognise that a is a proper spatial part of b at some time, we must demand that they be distinct objects in our model, but also that they are distinguishable at the time of evaluation, that is at some simultaneous state to the point of evaluation, and mutatis mutandis with the $\sqsubset^{T1}$ relation. To achieve this relative inditinguishability, we should require that $x \sqsubset y =_{def} x \sqsubseteq y \land \neg(y \sqsubseteq x)$.

Now, this can allow us to fomally model puzzles such as that of the boat of Theseus, or similar ones, as a case of certain individuals being part of others at different times. This construction allow us to formally approach to notions of "loose identity": we can loosely say that two things are the same if they have the same set of parts, as in the puzzle referred above, if the two are parts of the same entity (e.g. me as a baby and me as an undergraduate can be construed as two different individuals, being what identifies them the fact that the former was indistinguishable from me as a whole when I was a baby and the later was indistinguishable from me some years ago, therefore, they are "the same" in a non-strict sense). Similar relations can be construed from qualitative similarities: loose identity based on similarity would be strict identity of some subsets of the individuals. Of course a study of all the different uses or "A and B are identical/the same" is not the object of this work, but may this serve as a lead to possible analyses of such phenomenon.

4 Possible formal developments

As it stands, we have a modal system that basically combines S5 for the $[S]/\langle S\rangle$ operators with the temporal system of choice. Searching for complete classes of models for a given axiomatisation of the temporal aspect and the different relations of parthood –and it is noteworthy that some stipulations on the parthood relations may impose restrictions not only in the composition of the domain of the appropriate models, but also on the onderlying structure $\langle W, R\rangle$. Another very compelling development possibility is refining the structure of the spatial aspect of our models.

Another line of investigation that can be started from this framework is the embedding of counterpart-like semantics in our frames, by interpreting our modal statements not as being about individuals which happen to be identical to those referenced in the scope of the modality, bus as being about individuals which stand in some "counterparthood" relation with them, relation that in principle need not be an equivalence.

References

[1] Benthem, Johan van (2010) – Modal logic for open minds. Stanford, California, CSLI Publications.

[2] Blackburn, P., de Rijke, M., Venema, Y. (2005) – Modal logic, Cambridge university Press.

[3] Burgess, John P. (1979) – "Logic and Time" in The Journal of Symbolic Logic, Vol. 44, No. 4, pp. 566–582

[4] Fitting, M., Mendelsohn, R. L.(1990) – First-order modal logic. Dordrecht, Kluwer Academic.

[5] Hawley, K., – "Temporal Parts", The Stanford Encyclopedia of Philosophy (Winter 2010 Edition), Edward N. Zalta (ed.), `http://plato.stanford.edu/archives/win2010/entries/temporal-parts/`

[6] Hodkinson, I., Reynolds, M. (2007) – Temporal logic. In Handbook of modal logic (Vol. 11). Amsterdam; Boston: Elseiver.

[7] Nute, D. (1991) – "Historical Necessity and Conditionals" Noûs, Vol. 25, No. 2, pp. 161–175

[8] Oderberg, D. S. (2004) – "Temporal Parts and the Possibility of Change", Philosophy and Phe-nomenological Research Vol. LXIX, No. 3

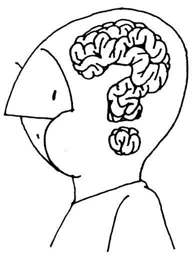
[9] Steen, M. (2008) – "Chisholm's changing conception of ordinary objects" , Grazer Philosophische Studien 76, pp. 1–56.

[10] , "Mereology", The Stanford Encyclopedia of Philosophy (Fall 2014 Edition), Ed-ward N. Zalta (ed.), forthcoming `http://plato.stanford.edu/archives/fall2014/entries/mereology/`

[11] Yourgrau, P. (1985) – "On the Logic of Indeterministic Time", The Journal of Philosophy, Vol. 82, No. 10, Eighty-Second Annual Meeting AmericanPhilosophical Association, Eastern Division, pp. 548–559

[12] Zanardo, A. (1996) - "Branching-Time Logic with Quantification over Branches: The Point of View of Modal Logic", The Journal of Symbolic Logic, Vol. 61, No. 1, pp. 1–39

News and Conference Reports



Report from the Japanese Chapter

R. Uehara (JAIST)

EATCS-JP/LA Workshop on TCS and Presentation Awards

The twelfth *EATCS/LA Workshop on Theoretical Computer Science* was held at Research Institute of Mathematical Sciences, Kyoto University, January 28 to January 30, 2014. (The program also can be found at `http://www2.infonets.hiroshima-u.ac.jp/lasymp/2014W/program/`.)

By attendees' voting, **Prof. Hiroshi Imai** (University of Tokyo) and **Ms. Ai Ishida** (Tokyo Institute of Technology) were selected at the the twelfth EATCS/LA Presentation Award:

> *Solving a Max Cut Benchmark by an Optimization Solver* by Takuto Ikuta, Hiroshi Imai, Yosuke Yano (The University of Tokyo)
>
> *Non-interactive Zero-Knowledge Proof Systems for Disavowal and Its Applications* by Ai Ishida (Tokyo Institute of Technology), Keita Emura (National Institute of Information and Communications Technology), Goichiro Hanaoka, Yusuke Sakai (National Institute of Advanced Industrial Science and Technology), Keisuke Tanaka (Tokyo Institute of Technology)

The award will be given them at the Summer LA Symposium held in July 2015.

We also established another presentation award, named "EATCS/LA Student Presentation Award" to encourage students. **Mr. Yuto Nakashima** (Kyushu Univeristy) who presented the following paper, was selected at the fourth EATCS/LA Student Presentation Award:

> *Lyndon ≤ LZ77 Conjecture* by Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda (Kyushu University)

The award (with some gift for playing in his laboratory) has been already given him at the last day, January 30, 2015.

Congratulations!

This workshop is jointly organized with *LA symposium*, Japanese association of theoretical computer scientists. Its purpose is to give a place for discussing topics on all aspects of theoretical computer science. (In fact, I've heard some different opinions that "L" stands for logic and/or language, and "A" stands for algorithm and/or automaton.) That is, this workshop is an unrefereed familiar meeting. All submissions are accepted for the presentation. There should be no problem of presenting these papers in refereed conferences and/or journals. We hold it twice a year (January/February, and July/August). If you have a chance, I recommend you to attend it. You can find the program of the last workshop in Appendix of this report.

Forthcoming Events in Japan

I am very happy to announce that (again) the first ICALP outside Europe will be held in Kyoto, Japan. That will be colocated with LICS 2015. You can find more information on the conferences at
`http://www.kurims.kyoto-u.ac.jp/icalp-lics2015/`.

ICALP 2015

The 42nd International Colloquium on Automata, Languages and Programming (ICALP 2015) will be held in Kyoto, Japan, during the week 6-10 July, 2015. The conference will be held at Grand Prince Hotel Kyoto
(`http://www.princehotels.com/en/kyoto/`), and workshops will be held at Kyoto University (`http://www.kyoto-u.ac.jp/en`).

LICS 2015

The 30th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2015) will be held in Kyoto, Japan, July 6–10, 2015. The details can be found at `http://lics.rwth-aachen.de/lics15/`.

I am also happy to announce that ISAAC will be held in Nagoya, Japan:

ISAAC 2015

The 26th International Symposium on Algorithms and Computation (ISAAC 2015) will be held in Nagoya, Japan, December 9–11, 2015. The important dates are as follows (To make sure for these important dates, please check the web site at `http://www.al.cm.is.nagoya-u.ac.jp/isaac2015/`):

Submission Deadline: June 19, 2015.

Notification of Acceptance: By August 31, 2015.

Camera Ready Copy: September 21, 2015.

Appendix:

Program of EATCS-JP/LA workshop on TCS (January 28th to 30th, 2015)

In the following program, "*" indicates that the talk is given in Student Session (shorter than the ordinary talk).

New Algorithms for Order Preserving Pattern Matching*
Takahiro Aoki, Yoshiaki Matsuoka, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda (Kyushu University)
Enumeration of α-Gapped Repeat on Overlap-Free Strings*

Yuta Fujishige, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda (Kyushu University)

Enumeration of α-Gapped Repeats in a Word*

Yuka Tanimura, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda (Kyushu University)

Non-interactive Zero-Knowledge Proof Systems for Disavowal and Its Applications

Ai Ishida (Tokyo Institute of Technology), Keita Emura (National Institute of Information and Communications Technology), Goichiro Hanaoka, Yusuke Sakai (National Institute of Advanced Industrial Science and Technology), Keisuke Tanaka (Tokyo Institute of Technology)

Streaming Algorithms for Sampling and Their Applications

Ryosuke Nakata (Tokyo Institute of Technology), Maxim Jourenko (Aachen University, Tokyo Institute of Technology), Keisuke Tanaka (Tokyo Institute of Technology/JST CREST)

On Arithmetic Garbled Circuits

Tomoyuki Komatsu (Tokyo Institute of Technology), Keisuke Tanaka (Tokyo Institute of Technology/JST CREST)

8k-Degree Grid Graph Representation of Tabular Diagrams

Takeo Yaku (Nihon University)

Online Computation of Fixed Gapped Palindrome*

Michitaro Nakamura, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda (Kyushu University)

Computing a Longest Common Flexible Pattern Including a Constrained Flexible Pattern*

Keita Kuboi, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda (Kyushu University)

All Five-Variable Logic Functions Can Be Computed by Three-Input Majority Gates with Depth Four*

Masao Moriya, Kazuyoshi Takagi, Naofumi Takagi (Kyoto University)

Approximating the Connected 2-Edge Dominating Set Problem

Tomoaki Shimoda, Toshihiro Fujito (Toyohashi University of Technology)

On Computability and Constructive Provability for Existence Theorems

Makoto Fujiwara (Tohoku University)

Simple #SAT Algorithms for Bounded Width Circuits and Bounded Depth Formulas

Hiroki Morizumi (Shimane University)

Dynamic Compressed Index

Takaaki Nishimoto (Kyushu University), Tomohiro I (Technische Universitaet Dortmund, Germany), Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda (Kyushu University)

The Class of the Computational Complexity of the Coin-Exchange Problem of Frobenius

Shunichi Matsubara (Aoyama Gakuin University)

Some Properties of Hippocratic Randomness

Hayato Takahashi (Gifu University)

On the Spanning Tree Congestion of Small Diameter Graphs*

Kohei Kubo, Yukiko Yamauchi, Shuji Kijima, Masafumi Yamashita (Kyushu University)

A Distributed Locomotion Algorithm for 3-Dimensional Metamorphic Robotic System[1]*

Fengqi Chen, Yukiko Yamauchi, Shuji Kijima, Masafumi Yamashita (Kyushu University)

The Team Assembling Problem for Heterogeneous Mobile Robots*

Zhiqiang Liu, Yukiko Yamauchi, Shuji Kijima, Masafumi Yamashita (Kyushu University)

Tangle and Ideal

Koichi Yamazaki (Gunma University)

Analyses of Space Complexity of Tree Evaluation Problems

Kazuo Iwama (Kyoto University), Atsuki Nagao (Kyoto University/JSPS CD2 Research Fellow)

Analytic Continuation in iRRAM: Implementations Inspired by Real Complexity Theory

Akitoshi Kawamura (University of Tokyo), Florian Steinberg, Holger Thies (Technische Universitaet Darmstadt)

Space Complexity of Self-Stabilizing Leader Election in Population Protocol on Hypernetworks*

Xiaoguang Xu, Yukiko Yamauchi, Shuji Kijima, Masafumi Yamashita (Kyushu University)

Lyndon $\leq$ LZ77 Conjecture*

Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, Masayuki Takeda (Kyushu University)

Randomized Approximation of the Frequency of Items in a Stream Using a Small Space*

Heejae Yim, Yukiko Yamauchi, Shuji Kijima, Masafumi Yamashita (Kyushu University)

Extracting LCS from Seaweed Diagrams

Yoshifumi Sakai (Tohoku University)

Polynomial-Time Learning of Formal Graph Systems with Bounded Tree-Width

Takayoshi Shoudai (Kyushu International University), Tomoyuki Uchida (Hiroshima City University)

Effective Method to Compute Frequencies of Order-Preserving n-Gram by Suffix Counting Representation

Yusuke Sato, Kazuyuki Narisawa, Ayumi Shinohara (Tohoku University)

Solving a Max Cut Benchmark by an Optimization Solver

Takuto Ikuta, Hiroshi Imai, Yosuke Yano (The University of Tokyo)

The Number of Matrix Multiplications for the Evaluation of Matrix Polynomial $I + A + A^2 + \cdots + A^{N-1}$*

Kotaro Matsumoto, Naofumi Takagi, Kazuyoshi Takagi (Kyoto University)

Abstracting Weighted Path Orders for Proving Termination of Term Rewriting Systems*

Takanori Omae (Nagoya University), Keiichirou Kusakari (Gifu University), Akihisa Yamada (National Institute of Advanced Industrial Science and Technology), Toshiki Sakabe (Nagoya University)

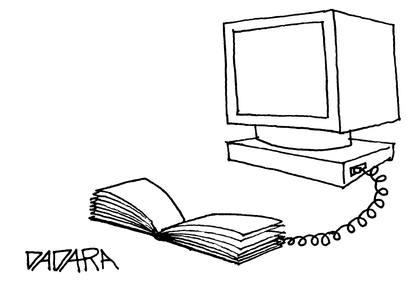
A Fast Filtration for Order-Preserving Matching

Youhei Ueki, Kazuyuki Narisawa, Ayumi Shinohara (Tohoku University)

The Japanese Chapter

Chair:	Osamu Watanabe
Vice Chair:	Ryuhei Uehara
Secretary:	Takehiro Ito
email:	eatcs-jp@is.titech.ac.jp
URL:	http://www.jaist.ac.jp/~uehara/EATCS-J/

Reflections on Influential Scientists and Ideas



Reflections on Influential Scientists and Ideas
A new section of the Bulletin of the EATCS

Luca Aceto
Reykjavik University

The piece by David Avis (School of Informatics, Kyoto University) that you are about to read offers a look at George Dantzig's best known contribution, the *simplex method*, and at its connections with theoretical computer science. In this article, David Avis also provides some food for thought for our research community and argues for a collaboration of the TCS and optimization communities to settle the question of whether there is a polynomial time pivot selection rule for the time-honoured simplex method.

Last year marked the centenary of George Bernard Dantzig's birth, and Kazuo Iwama originally commissioned this reflection piece to David Avis to celebrate that anniversary. Upon receiving David's article, Kazuo and I were struck by the thought that readers of the Bulletin might enjoy reading short articles devoted to anniversaries of influential scientists and to ideas related to theoretical computer science at large. Such reflection pieces could provide a new look at the legacy of pioneers and at some of the pearls of our subject, as well as possibly highlight some of the challenges that still need to be met. We feel that they would be useful to young members of our community, students and experienced researchers alike.

A list of upcoming anniversaries includes the 200th anniversary of the birth of George Boole and the 100th anniversary of the birth of Richard Hamming in 2015, as well as the centenary of Claude Shannon in 2016. Please get in touch with Kazuo if you are interested in contributing a reflection piece on any of those figures, or on a scientist or a "pearl of theoretical computer science" of your choice.

For the moment, Kazuo and I hope that you will enjoy David Avis' piece that gives this new section of the Bulletin an excellent start.

George Dantzig: father of the simplex method

David Avis
Kyoto University
avis@cs.mcgill.ca

100 years have passed since George Bernard Dantzig was born, about 70 years since he started developing the simplex method, and 10 since he died. His main legacy is the simplex method, which *Computer Science and Engineering* included as one of the top 10 algorithms (sic) of the 20th century. In 2006 Martin Grötschel said: "the development of linear programming is - in my opinion - the most important contribution of the mathematics of the 20th century to the solution of practical problems arising in industry and commerce." Yet linear programming, and especially the simplex method, has had a tenuous relation to theoretical computer science (TCS) over the years.

Let us begin with the fact that it is the *simplex method* not the *simplex algorithm*. It is a method because it describes a class of algorithms. An algorithm in the class is initialized at any vertex of a convex polyhedron and will follow edges on the boundary of the polyhedron until reaching a vertex that maximizes a given linear function[1]. These algorithms are specified by a *pivot rule* that is normally deterministic and defines a unique path along which the objective function increases monotonically until an optimum vertex is reached. It may not be the case that the algorithm makes progress along an edge at every step: it may make repeated pivots at a given vertex before making progress, a phenomenon known as *stalling*. Dantzig's original (and still widely used) pivot rule has two unfortunate properties. Firstly, it may stall indefinitely, going into an infinite loop. Secondly, it may follow a path on the polyhedron of exponential length, as shown by Klee and Minty in 1972. The first problem can be efficiently solved by the lexicographic ratio test, a method that simulates simplicity and renders all pivot rules finite, or by perturbation. The second problem has never been solved. Following Klee and Minty, a series of papers gave exponential lower bounds for the then known pivot rules, using variants of Klee-Minty cubes. These types of examples can be defeated by history based rules, such as those introduced by Zadeh in 1980. These

[1]For simplicity we omit the cases where the polyhedron is unbounded or empty.

rules resisted analysis for more than 30 years, despite Zadeh's offer of a $1000 prize.

Even with these serious limitations, the simplex method dominates optimization, especially integer programming, where it is routinely used to optimally solve large examples of NP-hard integer programming problems. For example the Concorde program of Applegate, Bixby, Cook and Chvátal has found the optimum solution of traveling salesman problems (TSPs) with as many as 85,900 cities. It is based on the branch-and-cut method that generates enormous numbers of extremely large linear programs. The 85,900 city TSP involved the solution of about one million sparse LPs each with roughly 100,000 constraints and 170,000 variables. Each new LP is created by adding a number of cutting planes to an LP with a fractional optimum solution. Using Dantzig's dual simplex method and the original optimum solution, this new LP can be optimized with ease in practice, but of course there is no theoretical foundation for this. Incidentally this method was pioneered by Dantzig, along with Fulkerson and Johnson, in their groundbreaking 1954 paper where they optimally solved a 49 city TSP by hand[2]!

Other methods for linear programming exist of course. Starting with the ellipsoid method of Khachian in 1979 the competing class of interior point algorithms has been extensively developed. To a TCS eye these are eminently preferable, being provably polynomial. Although the ellipsoid method is a non-starter for solving any problem encountered in practice, later algorithms are competitive with the simplex method. However this should not ease TCS discomfort with the simplex method. Imagine things were reversed: Dantzig had discovered an interior method in 1947 and Khachian had proposed the simplex method in 1979. What chance would Khachian have had of getting a non-finite, non-algorithm running in exponential time for a problem in P accepted to FOCS/STOC/SODA? Would it even have been programmed? When it comes to handling cutting planes the simplex method wins hands down against interior point methods. And it is integer programming that provides the huge bulk of LPs that need to be solved in practice. The simplex method also delivers an optimal basis and a proof of optimality via the dual multipliers that can be independently verified. But most uncomfortably of all: it really does work in practice and seemingly flies in the face of a lot of what TCS teaches students about P, NP-hardness, and exponential algorithms.

Given its importance why is it that Dantzig's simplex method was largely ignored in the TCS community until relatively recently? Most of us who studied at Stanford's Department of Operations Research, that Dantzig pioneered, largely funded, and presided over, were not enamored by the rather opaque description

[2]Despite Michigan State computer scientist Randy Olson's recent claims to the contrary: "With 50 landmarks to put in order, we would have to exhaustively evaluate 3 x 10^{64} possible routes to find the shortest one." http://www.randalolson.com/2015/03/08/computing-the-optimal-road-trip-across-the-u-s/)

in his opus *Linear Programming and Extensions*. This may have delayed the widespread understanding of the simplex method but was completely rectified by Chvátal's lucid description in the first few chapters of his now classic *Linear Programming*. I always found it very impressive that my PhD supervisor, a 27 year old nontenured assistant professor in the department at the time, dared to rewrite Dantzig's description in plain English[3]. Anyway he did, so there is no excuse for anyone not to understand it.

Many TCS books on algorithms ignore linear programming altogether. Where it is discussed it is often treated almost as a footnote. The encyclopedic Cormen, Leiserson, Rivest and Stein's *Algorithms(3rd edition)* describes it in Chapter 29 (of 35), Selected Topics. The excellent and highly readable Kleinberg and Tardos' *Algorithm Design* treats it in Chapter 11 (of 13), a chapter on approximation algorithms! This despite the fact that both texts contain many examples of the simplex method in disguised form in earlier chapters: Dijkstra's algorithm, Bellman-Ford, network flows, matchings etc.

Early TCS interest in linear programming was shown by computational geometers. However algorithms that are exponential in the dimension and linear in the number of constraints are of little interest when the number of variables is counted in the hundreds of thousands. A major theoretical result appeared in 1991 with the joint discovery by Kalai and Matousek, Sharir & Welzl of a subexponential pivot selection method based on selecting random facets at the current vertex. However this result has never been derandomized and, being recursive in the dimension, is not practical for the kinds of LPs encountered in practice. Another line of research involved the probabilistic analysis of the simplex method, commencing in the late 1970s with Borgwardt(Lanchester prize) and continuing into this century with the smoothed analysis of Spielman and Teng(Gödel prize). This is deep work indeed and gives considerable insight into the success of the simplex method for certain types of problems. However large scale combinatorial problems hardly seem to fit these models. More recently, in 2011, Friedmann, Hansen & Zwick gave subexponential lower bounds for the random facet rule and also for the more intuitive random edge rule, receiving STOC's best paper award. Also in 2011, Friedmann gave a subexponential lower bound for Zadeh's history based rule, solving that 30 year open problem, and picked up a cheque for $1000 from the man in person at an IPAM meeting in Los Angeles. Similar results followed. Each simplex pivot generates a lot of information about the polyhedron, and for a polynomial time pivot rule this entire history could be recorded. So history based rules offer good candidates for polynomial upper bounds. Much work needs to be done here. Surely the close collaboration of TCS and the optimization commu-

[3]Earlier Dantzig had asked Chvátal how old he was. When he heard, Dantzig replied: "Then I pity you because you are going to have to live through all the shit that is coming up."

nity would be able to settle this question: is there or is there not a polynomial time pivot selection rule for the simplex method? Of course I think all of us, including George, hope for a positive answer that is both strongly polynomial time and a winner in practice!

Kyoto University
June 1, 2015

Contributions by EATCS Award Recipients

Sampling from Discrete Distributions and Computing Fréchet Distances

Abstract of Doctoral Thesis [8]

Karl Bringmann

1 Sampling from Discrete Distributions

Sampling from a probability distribution is a fundamental problem that lies at the heart of randomized computation, and has never been as important as today, as most sciences perform computer simulations of models involving randomness. We approach this problem area from an algorithm theory perspective. The central problem in the first part of this dissertation is *proportional sampling*, defined as follows. We are given non-negative numbers $p_1, \dots, p_n$ that define a probability distribution on $\{1, \dots, n\}$ by picking i with probability proportional to p_i, i.e., the probability of sampling i is $\frac{p_i}{\sum_j p_j}$. The task is to build a data structure that supports sampling from this distribution as a query. The classic solution to this problem is the alias method by Walker from '74 [38], which uses $O(1)$ query time and $O(n)$ preprocessing time, i.e., the time for building the data structure is $O(n)$. It is easy to see that both time bounds of Walker's method are optimal. We extend this classic data structure in various directions as follows.

Succinct Sampling While the time bounds are well understood, space usage of discrete sampling algorithms has received little attention. To bound its space usage, we show that Walker's alias method can be implemented on the Word RAM model of computation (where each cell stores $w = \Omega(\log n)$ bits) with a space usage of $n(w + 2\lg n + O(1))$ bits [12]. Using the terminology of succinct data structures, this solution has a redundancy of $2n \lg n + O(n)$ bits, i.e., it uses $2n \lg n + O(n)$ bits in addition to the information theoretic minimum required for storing the input. We examine whether this space usage can be improved in two common models for data structures from the field of succinct data structures: In the systematic model, in which the input is read-only, we present a novel data structure using $r + O(w)$ redundant bits, $O(n/r)$ expected query time, and $O(n)$ preprocessing time for any r. This is an improvement in redundancy by a factor of $\Omega(\log n)$ over the alias method for $r = n$, even though the alias method is not systematic. Moreover, we complement this data structure with a lower bound showing that this trade-off is tight for systematic data structures. In the non-systematic model, in which the input numbers may be represented in more clever ways than just storing them one-by-one, we demonstrate a very surprising separation from the systematic case: With only 1 redundant bit, it is possible to support optimal $O(1)$ expected query time and $O(n)$ preprocessing time! On the one hand, these results improve upon the space requirement of the classic solution for a fundamental sampling problem, and on the other hand, they provide the strongest known separation between the systematic and non-systematic model for any data structure problem. Finally, we also believe that these upper bounds are practically efficient and simpler than Walker's alias method.

Restricted Inputs Since the preprocessing and query time bounds of Walker's alias method are optimal in the worst case, we examine the situation where we have additional knowledge about the input distribution [14]. For example, assume that we have the guarantee that the input is sorted. We show that, in this case, the preprocessing time can be reduced to $O(\log n)$ while still achieving expected query time $O(1)$. Moreover, one can further reduce the preprocessing time at the price of increasing the query time, specifically, any expected query time $O(t)$ can be achieved with $O(\log_t n)$ preprocessing time. In particular, we can achieve preprocessing and expected query time $O(\log n / \log \log n)$. We also show tight lower bounds for this trade-off curve at all of its points.

Subset Sampling Let us consider a different sampling problem [14]: In *subset sampling* we are given $p_1, \dots, p_n$ and consider n independent events, where event i occurs with probability p_i. The task is to sample the set of occurring events. This problem can be seen as a generalization of proportional sampling, since we show that any data structure for subset sampling can be transformed into a data structure for proportional sampling with the same asymptotic running times. As for proportional sampling, we consider sorted and unsorted input sequences and in both cases present data structures with optimal preprocessing-query time trade-offs. The situation is more complex than for proportional sampling, since the running times now also depend on the expected size μ of the sampled subset. For instance, we design a data structure for subset sampling on sorted inputs with preprocessing and expected query time $O(1 + \mu + \frac{\log n}{\log(\log(n)/\mu)})$, which corresponds to one point on an optimal trade-off curve.

Special distributions Particularly fast sampling methods are known for special distributions such as Bernoulli, geometric, or binomial random variables. For instance, a geometric random variable Geo(p) can be sampled using the simple formula $\lceil \frac{\log R}{\log(1-p)} \rceil$, where R is a uniformly random real in $(0, 1)$. On a Real RAM, this formula can be evaluated in constant time. However, on real-life computers this formula is typically evaluated with the usual floating point precision, so that it is not exact. Hence, we study whether special distributions can be sampled exactly and efficiently on a bounded-precision machine such as the Word RAM. We prove that a geometric random variable Geo(p) can be sampled in expected time $O(1 + \log(1/p)/w)$ on the Word RAM [10]. This is optimal, as it matches the expected number of output words. To this end, we have to avoid the simple formula above, as it is a long-standing open problem to compute logarithms in linear time. We also present optimal sampling algorithms on the Word RAM for Bernoulli and binomial random variables as well as ErdÅŚsâĂŞRÃľnyi random graphs.

Applications The insights on the above fundamental problems also prove beneficial for sampling more complex random structures motivated by physics. Consider the following simple exemplary process. The Internal Diffusion Limited Aggregation (IDLA [33]) process places particles on the initially empty integer grid $\mathbb{Z}^2$. In every step, a new particle is born at the origin and performs a random walk until it hits an empty grid cell and occupies it. This process models certain chemical and physical phenomena such as corrosion and the melting of a solid around a heat source. The emerging shape is roughly a ball. Proving this rigorously turned out to be a challenging mathematical problem which has only recently been resolved [32]. From a computational perspective, the trivial simulation algorithm takes time $O(n^2)$ to generate an IDLA shape with n particles. We prove that $O(n \log^2 n)$ time and $O(\sqrt{n} \log n)$ space are sufficient for exactly sampling from the IDLA distribution [15], which allows for experiments on a much larger scale.

2 Computing Fréchet Distances

The second part of this dissertation belongs to the area of computational geometry and deals with algorithms for the Fréchet distance, which is a popular measure of similarity of curves. Intuitively, the (continuous) Fréchet distance of two curves P, Q is the minimal length of a leash required to connect a dog to its owner, as they walk along P and Q, respectively, without backtracking.

Alt and Godau introduced the Fréchet distance to computational geometry in 1991 [4, 27]. For polygonal curves P and Q with n and m vertices, respectively, $n \geq m$, they presented an $O(nm \log(nm))$ algorithm. Since Alt and Godau's seminal paper, Fréchet distance has become a rich field of research, with many variants, extensions, and generalizations (see, e.g., [3, 17, 20, 24]). Being a natural measure for curve similarity, Fréchet distance has found applications in various areas such as signature verification (see, e.g., [34]), map-matching tracking data (see, e.g., [7]), and moving objects analysis (see, e.g., [18]).

A particular variant that we also discuss in this dissertation is the *discrete* Fréchet distance. Here, intuitively the dog and its owner are replaced by two frogs, and in each time step each frog can jump to the next vertex along its curve or stay at its current vertex. Defined in [25], the original algorithm for the discrete Fréchet distance has running time $O(nm)$.

Quadratic time complexity? Recently, improved algorithms have been found for the classic variants. Agarwal et al. [2] showed how to compute the discrete Fréchet distance in (mildly) subquadratic time $O(nm\frac{\log\log n}{\log n})$. Buchin et al. [19] designed algorithms for the continuous Fréchet distance with running time $O(n^2\sqrt{\log n}(\log\log n)^{3/2})$ on the Real RAM and $O(n^2(\log\log n)^2)$ on the Word RAM. However, the problem remained open whether there is a *strongly subquadratic*[1] algorithm for the Fréchet distance, i.e., an algorithm with running time $O(n^{2-\delta})$ for any $\delta > 0$. The only known lower bound shows that the Fréchet distance takes time $\Omega(n\log n)$ (in the algebraic decision tree model) [16]. The typical way of proving (conditional) quadratic lower bounds for geometric problems is via 3SUM [26]. In fact, Helmut Alt conjectured that the Fréchet distance is 3SUM-hard, but this conjecture remains open. Instead of relating the Fréchet distance to 3SUM, we consider the Strong Exponential Time Hypothesis.

Strong Exponential Time Hypothesis (SETH) The hypothesis SETH, introduced by Impagliazzo, Paturi, and Zane [30, 31], provides a way of proving conditional lower bounds. SETH asserts that satisfiability has no algorithms that are much faster than exhaustive search.

Hypothesis SETH: *For no $\varepsilon > 0$,* k-SAT *can be solved in time $O(2^{(1-\varepsilon)N})$ for all $k \geq 3$.*

Note that exhaustive search takes time $O^*(2^N)$ and the best-known algorithms for k-SAT have a running time of the form $O(2^{(1-c/k)N})$ for some constant $c > 0$ [36]. Thus, SETH is a reasonable hypothesis and, due to lack of progress in the last decades, can be considered unlikely to fail. It has been observed before this work that SETH can be used to prove lower bounds for polynomial time problems such

[1]We use the term *strongly subquadratic* to differentiate between this running time and the *(mildly) subquadratic* $O(n^2\log\log n/\log n)$ algorithm from [2].

as k-Dominating Set and others [35], the diameter of sparse graphs [37], and dynamic connectivity problems [1].

Main lower bound Our main result of the second part of this dissertation gives strong evidence that the Fréchet distance may have no strongly subquadratic algorithms by relating it to the Strong Exponential Time Hypothesis. *We prove that there is no $O(n^{2-\delta})$ algorithm for the (continuous or discrete) Fréchet distance for any $\delta > 0$, unless* SETH *fails [9].* Since SETH is a reasonable hypothesis, by this result one can consider it unlikely that the Fréchet distance has strongly subquadratic algorithms. In particular, any strongly subquadratic algorithm for the Fréchet distance would not only give improved algorithms for k-SAT that are much faster than exhaustive search, but also for various other problems such as Hitting Set, Set Splitting, and NAE-SAT via the reductions in [22]. Alternatively, in the spirit of [35], one can view the above theorem as a possible attack on k-SAT, as algorithms for the Fréchet distance now could provide a route to faster k-SAT algorithms. In any case, anyone trying to find strongly subquadratic algorithms for the Fréchet distance should be aware that this is as hard as finding improved k-SAT algorithms, which might be impossible.

We remark that all our lower bounds (unless stated otherwise) hold in the Euclidean plane, and thus also in $\mathbb{R}^d$ for any $d \geq 2$.

Extensions We extend our main lower bound in two important directions: We show approximation hardness and we show tight lower bounds if one curve has much fewer vertices than the other, $m \ll n$. In order to state our result, we first formalize that a statement holds for "$m \approx n^\gamma$ for any γ". We say that a statement holds *for any polynomial restriction of $n^{\gamma_0} \leq m \leq n^{\gamma_1}$* if it holds restricted to instances with $n^{\gamma-\delta} \leq m \leq n^{\gamma+\delta}$ for any constants $\delta > 0$ and $\gamma_0 + \delta \leq \gamma \leq \gamma_1 - \delta$. *We prove that there is no* 1.001*-approximation with running time $O((nm)^{1-\delta})$ for the (continuous or discrete) Fréchet distance for any $\delta > 0$, unless* SETH *fails. This holds for any polynomial restriction of* $1 \leq m \leq n$ *[9].* In recent work together with Wolfgang Mulzer focussing on the discrete Fréchet distance [13], we improve this result further by excluding 1.399-approximation algorithms even for one-dimensional curves, assuming SETH. Moreover, we present an α-approximation in time $O(n^2/\alpha + n \log n)$ for any $\alpha \geq 1$.

Realistic input curves In attempts to break the apparent quadratic time barrier at least for realistic inputs, various restricted classes of curves have been considered, such as backbone curves [6], κ-bounded and κ-straight curves [5], and ϕ-low density curves [24]. The most popular model of realistic inputs are *c-packed curves*. A curve π is c-packed if for any point $z \in \mathbb{R}^d$ and any radius

$r > 0$ the total length of π inside the ball $B(z, r)$ is at most cr, where $B(z, r)$ is the ball of radius r around z. This model is well motivated from a practical point of view. The model has been used for several generalizations of the Fréchet distance [21, 23, 28, 29]. Driemel et al. [24] introduced c-packed curves and presented a $(1 + \varepsilon)$-approximation for the continuous Fréchet distance in time $O(cn/\varepsilon + cn \log n)$, which works in any $\mathbb{R}^d$, $d \geq 2$.

While this algorithm takes near-linear time for small c and $1/\varepsilon$, is is not clear whether its dependence on c and $1/\varepsilon$ is optimal for c and $1/\varepsilon$ that grow with n. We give strong evidence that the algorithm of Driemel et al. has optimal dependence on c for any constant $0 < \varepsilon \leq 0.001$. *We prove that there is no* 1.001*-approximation with running time* $O((cn)^{1-\delta})$ *for the (continuous or discrete) Fréchet distance on c-packed curves for any* $\delta > 0$*, unless* SETH *fails. This holds for any polynomial restriction of* $1 \leq c \leq n$ *[9].* Since we prove this claim for any polynomial restriction $c \approx n^\gamma$, this result also excludes 1.001-approximations with running time, say, $O(c^2 + n)$.

Regarding the dependence on ε, in any dimension $d \geq 5$ we can prove a conditional lower bound that matches the dependency on ε of the algorithm by Driemel et al. up to a polynomial. *We prove that in* $\mathbb{R}^d$*,* $d \geq 5$*, there is no* $(1 + \varepsilon)$*-approximation for the (continuous or discrete) Fréchet distance on c-packed curves running in time* $O(\min\{cn/\sqrt{\varepsilon}, n^2\}^{1-\delta})$ *for any* $\delta > 0$*, unless* SETH *fails. This holds for sufficiently small* $\varepsilon > 0$ *and any polynomial restriction of* $1 \leq c \leq n$ *and* $\varepsilon \leq 1$ *[9].*

This, however, still leaves open a gap between the best-known upper and (conditional) lower bounds. We resolve this issue positively by giving a faster algorithm with time complexity $O(cn \log^2(1/\varepsilon)/\sqrt{\varepsilon} + cn \log n)$ [11]. This dependence on c, n and ε is optimal in high dimensions apart from lower order factors, unless SETH fails. In fact, the new algorithm was obtained by examining and exploiting properties that prevent a stronger lower bound, thus demonstrating that SETH-based lower bounds may also inspire algorithmic improvements. We leave open the challenging problem of determining the optimal running time in dimensions $d = 2, 3, 4$.

References

[1] A. Abboud and V. Vassilevska Williams. Popular conjectures imply strong lower bounds for dynamic problems. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, 2014. To appear.

[2] P. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir. Computing the dis-

crete Fréchet distance in subquadratic time. In *Proc. 24th ACM-SIAM Symposium on Discrete Algorithms (SODA'13)*, pages 156–167, 2013.

[3] H. Alt and M. Buchin. Can we compute the similarity between surfaces? *Discrete & Computational Geometry*, 43(1):78–99, 2010.

[4] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5(1-2):78–99, 1995.

[5] H. Alt, C. Knauer, and C. Wenk. Comparison of distance measures for planar curves. *Algorithmica*, 38(1):45–58, 2004.

[6] B. Aronov, S. Har-Peled, C. Knauer, Y. Wang, and C. Wenk. Fréchet distance for curves, revisited. In *Proc. 14th Annual European Symposium on Algorithms (ESA'06)*, volume 4168 of *LNCS*, pages 52–63. 2006.

[7] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st International Conference on Very Large Data Bases (VLDB'05)*, pages 853–864, 2005.

[8] K. Bringmann. Sampling from discrete distributions and computing Fréchet distances, 2014. Dissertation. `http://scidok.sulb.uni-saarland.de/volltexte/2015/5988/`.

[9] K. Bringmann. Why walking the dog takes time: Frechet distance has no strongly subquadratic algorithms unless SETH fails. In *Proc. 55th Annual IEEE Symposium on Foundations of Computer Science (FOCS'14)*, 2014. To appear.

[10] K. Bringmann and T. Friedrich. Exact and efficient generation of geometric random variates and random graphs. In *Proc. 40th International Colloquium on Automata, Languages, and Programming (ICALP'13)*, volume 7965 of *LNCS*, pages 267–278, 2013.

[11] K. Bringmann and M. Künnemann. Improved approximation for Fréchet distance on c-packed curves matching conditional lower bounds, 2014. Submitted. Preprint at arXiv 1408.1340.

[12] K. Bringmann and K. G. Larsen. Succinct sampling from discrete distributions. In *Proc. 45th Annual ACM Symposium on Symposium on Theory of Computing (STOC'13)*, pages 775–782, New York, NY, USA, 2013. ACM.

[13] K. Bringmann and W. Mulzer. Approximability of the discrete fréchet distance. In *Proc. 31st International Symposium on Computational Geometry (SoCG'15)*, 2015. To appear.

[14] K. Bringmann and K. Panagiotou. Efficient sampling methods for discrete distributions. In *Proc. 39th International Colloquium on Automata, Languages, and Programming (ICALP'12)*, volume 7391 of *LNCS*, pages 133–144, 2012.

[15] K. Bringmann, F. Kuhn, K. Panagiotou, U. Peter, and H. Thomas. Internal DLA: Efficient simulation of a physical growth model. In *Proc. 41th International Colloquium on Automata, Languages, and Programming (ICALP'14)*, volume 8572 of *LNCS*, pages 247–258, 2014.

[16] K. Buchin, M. Buchin, C. Knauer, G. Rote, and C. Wenk. How difficult is it to walk the dog? In *Proc. 23rd European Workshop on Computational Geometry (EWCG'07)*, pages 170–173, 2007.

[17] K. Buchin, M. Buchin, and Y. Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proc. 20th ACM-SIAM Symposium on Discrete Algorithms (SODA'09)*, pages 645–654, 2009.

[18] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. *International Journal of Computational Geometry & Applications*, 21(3):253–282, 2011.

[19] K. Buchin, M. Buchin, W. Meulemans, and W. Mulzer. Four soviets walk the dog - with an application to Alt's conjecture. In *Proc. 25th ACM-SIAM Symposium on Discrete Algorithms (SODA'14)*, pages 1399–1413, 2014.

[20] E. W. Chambers, É. Colin de Verdière, J. Erickson, S. Lazard, F. Lazarus, and S. Thite. Homotopic Fréchet distance between curves or, walking your dog in the woods in polynomial time. *Computational Geometry*, 43(3):295–311, 2010.

[21] D. Chen, A. Driemel, L. J. Guibas, A. Nguyen, and C. Wenk. Approximate map matching with respect to the Fréchet distance. In *Proc. 13th Workshop on Algorithm Engineering and Experiments (ALENEX'11)*, pages 75–83, 2011.

[22] M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. In *Proc. 27th IEEE Conference on Computational Complexity (CCC'12)*, pages 74–84, 2012.

[23] A. Driemel and S. Har-Peled. Jaywalking your dog: computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, 2013.

[24] A. Driemel, S. Har-Peled, and C. Wenk. Approximating the Fréchet distance for realistic curves in near linear time. *Discrete & Computational Geometry*, 48(1):94–127, 2012.

[25] T. Eiter and H. Mannila. Computing discrete Fréchet distance. Technical Report CD-TR 94/64, Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria, 1994.

[26] A. Gajentaan and M. H. Overmars. On a class of $O(n^2)$ problems in computational geometry. *Computational Geometry: Theory and Applications*, 5 (3):165–185, 1995.

[27] M. Godau. A natural metric for curves - computing the distance for polygonal chains and approximation algorithms. In *Proc. 8th Symposium on Theoretical Aspects of Computer Science (STACS'91)*, volume 480 of *LNCS*, pages 127–136, 1991.

[28] J. Gudmundsson and M. Smid. Fréchet queries in geometric trees. In *Proc. 21st Annual European Symposium on Algorithms (ESA'13)*, volume 8125 of *LNCS*, pages 565–576. 2013.

[29] S. Har-Peled and B. Raichel. The Fréchet distance revisited and extended. In *Proc. 27th Annual Symposium on Computational Geometry (SoCG'11)*, pages 448–457, 2011.

[30] R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

[31] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *Journal of Computer and System Sciences*, 63(4): 512–530, 2001.

[32] D. Jerison, L. Levine, and S. Sheffield. Logarithmic fluctuations for internal DLA. *Journal of the American Mathematical Society*, 25:271–301, 2012.

[33] P. Meakin and J. M. Deutch. The formation of surfaces by diffusion limited annihilation. *The Journal of Chemical Physics*, 85:2320, 1986.

[34] M. E. Munich and P. Perona. Continuous dynamic time warping for translation-invariant curve alignment with applications to signature verification. In

Proc. 7th IEEE International Conference on Computer Vision, volume 1, pages 108–115, 1999.

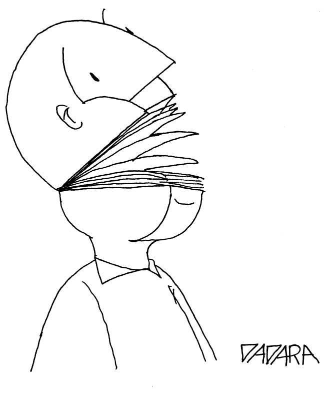
[35] M. Pătraşcu and R. Williams. On the possibility of faster SAT algorithms. In *Proc. 21nd ACM-SIAM Symposium on Discrete Algorithms (SODA'10)*, pages 1065–1075, 2010.

[36] R. Paturi, P. Pudlák, M. E. Saks, and F. Zane. An improved exponential-time algorithm for k-sat. *Journal of the ACM*, 52(3):337–364, 2005.

[37] L. Roditty and V. Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Proc. 45th Annual ACM Symposium on Symposium on Theory of Computing (STOC'13)*, pages 515–524, 2013.

[38] A. J. Walker. New fast method for generating discrete random numbers with arbitrary distributions. *Electronic Letters*, 10(8):127–128, 1974.

Book Introduction by the Authors



Book Introduction by the Authors

Invited by

Luca Aceto

`luca.aceto@gmail.com`
President of EATCS
Reykjavik University, Reykjavik, Iceland

SEMANTICS OF PROBABILISTIC PROCESSES
AN OPERATIONAL APPROACH

Yuxin Deng
Shanghai Jiaotong University
`yuxindeng@sjtu.edu.cn`

Subject matter

With the rapid development of computer network and communication technology, the study of concurrent and distributed systems has become increasingly important. Among various models of concurrent computation, process calculi have been widely investigated and successfully used in the specification, design, analysis and verification of practical concurrent systems. In recent years, probabilistic process calculi have been proposed to describe and analyse quantitative behaviour of concurrent systems, which calls for the study of semantic foundations of probabilistic processes.

In "Semantics of Probabilistic Processes" [4] we adopt an operational approach to describing the behaviour of nondeterministic and probabilistic processes. The semantic comparison of different systems is based on appropriate behavioural relations such as bisimulation equivalences and testing preorders.

This book mainly consists of two parts. The first part provides an elementary account of bisimulation semantics for probabilistic processes from metric, logical and algorithmic perspectives. The second part sets up a general testing framework and specialises it to probabilistic processes with nondeterministic behaviour. The resulting testing semantics is treated in depth. A few variants of it are shown to coincide, and they can be characterised in terms of modal logics and coinductively defined simulation relations. Although in the traditional (nonprobabilistic) setting, simulation semantics is in general finer (i.e. it distinguishes more processes) than testing semantics, for a large class of probabilistic processes, the gap between simulation and testing semantics disappears. Therefore, in this case, we have a semantics where both negative and positive results can be easily proved: to show that two processes are not related in the semantics, we just give a witness test, and to prove that two processes are related, we only need to establish a simulation relation.

Why yet another book?

Three decades have passed since the well-known books on process algebras by Hoare [8], Milner [10], Baeten and Weijland [3], and Hennessy [7] were published. In the meanwhile some excellent textbooks have appeared, including those by Roscoe [13, 14], Milner [11], Fokkink[6], Sangiorgi and Walker [17], Aceto et al. [1], Sangiorgi [15], as well as Sangiorgi and Rutten [16]. They are mainly about classical (nonprobabilistic) process algebras. For probabilistic concurrency theory, the book by Panangaden [12] is dedicated to the model of labelled Markov Processes and the book by Doberkat [5] treats stochastic logics in depth. Probabilistic model checking is well covered in the books by Baier and Katoen [2], and Kwiatkowska et al. [9]. This book, however, collects some recent developments in probabilistic testing semantics and gives an elementary account of probabilistic bisimulation semantics. Below we give a rough overview of the book's contents.

Mathematical preliminaries

In order to study the semantics of probabilistic processes, several mathematical concepts and results turn out to be very useful. They are collected in Chapter 2, including, for example, continuous functions over complete lattices, the Knaster-Tarski fixed-point theorem, induction and coinduction proof principles, compact sets in topological spaces, the separation theorem in geometry, the Banach fixed-point theorem in metric spaces, the π-λ theorem in probability spaces and the duality theorem in linear programming. The purpose of introducing these contents is to make the proofs in later chapters more accessible to postgraduate students and junior researchers entering the discipline of theoretical computer science.

Probabilistic bisimulation

In this book we work within a framework that features the co-existence of probability and nondeterminism. More specifically, we deal with *probabilistic labelled transition systems (pLTS's)* that are an extension of the usual *labelled transition systems* so that a step of transition is in the form $s \xrightarrow{a} \Delta$, meaning that state s can perform action a and evolve into a distribution Δ over some successor states. The diagram in Figure 1 describes a pLTS; states are represented by nodes of the form $\bullet$ and distributions by nodes of the form $\circ$.

Let s and t be two states in a pLTS. They are related by *probabilistic simulation* $\mathcal{R}$, written $s \mathrel{\mathcal{R}} t$, if for each transition $s \xrightarrow{a} \Delta$ from s there exists a transition $t \xrightarrow{a} \Theta$ from t such that Θ can somehow mimic the behaviour of Δ according to $\mathcal{R}$. To formalise the mimicking of Δ by Θ, we have to *lift* $\mathcal{R}$ to be a relation $\mathcal{R}^\dagger$ between distributions over states so that we can require $\Delta \mathrel{\mathcal{R}^\dagger} \Theta$.

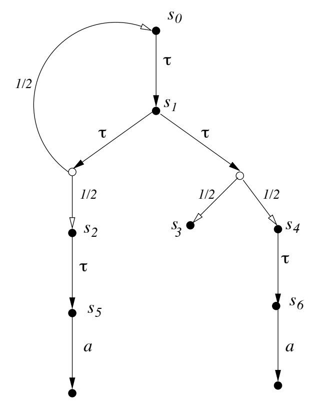


Figure 1: A pLTS

Various methods of lifting relations have appeared in the literature, but they can be reconciled. Essentially, there is only one lifting operation, which has been rediscovered in different occasions and presented in different forms. Moreover, we argue that the lifting operation is interesting in itself. This is justified by its intrinsic connection with some fundamental concepts in mathematics, notably *the Kantorovich metric*. For example, it turns out that our lifting of binary relations from states to distributions nicely corresponds to the lifting of metrics from states to distributions by using the Kantorovich metric. In addition, the lifting operation is closely related to *the maximum flow problem* in optimisation theory.

In Chapter 3 we provide three characterisations of probabilistic bisimulation, from the perspectives of modal logics, metrics, and decision algorithms.

- Our logical characterisation of probabilistic bisimulation consists of two aspects: *adequacy* and *expressivity*. A logic $\mathcal{L}$ is adequate when two states are bisimilar if and only if they satisfy exactly the same set of formulae in $\mathcal{L}$. The logic is expressive when each state s has a characteristic formula φ_s in $\mathcal{L}$ such that state t is bisimilar to s if and only if t satisfies φ_s. We introduce a probabilistic-choice modality to capture the behaviour of distributions. Intuitively, distribution Δ satisfies the formula $\bigoplus_{i\in I} p_i \cdot \varphi_i$ if there is a decomposition of Δ into a convex combination of some distributions, $\Delta = \sum_{i\in I} p_i \cdot \Delta_i$, and each Δ_i conforms to the property specified by φ_i. When the new modality is added to the Hennessy-Milner logic we obtain an adequate logic for probabilistic bisimilarity; when it is added to the modal mu-calculus we obtain an expressive logic.

- By metric characterisation of probabilistic bisimulation, we mean to give a pseudometric such that two states are bisimilar if and only if their distance is 0 when measured by the pseudometric. More specifically, we show that bisimulations correspond to pseudometrics that are postfixed points of a monotone function, and in particular bisimilarity corresponds to a pseudometric that is the greatest fixed point of the monotone function.

- As to the algorithmic characterisation, we first introduce a partition refinement algorithm to check whether two states are bisimilar. Then we provide an "on-the-fly" algorithm that checks whether two states are related by probabilistic bisimilarity. The schema of the algorithm is to approximate probabilistic bisimilarity by iteratively accumulating information about state pairs (s, t) where s and t are not bisimilar. In each iteration we dynamically construct a relation $\mathcal{R}$ as an approximant. Then we verify that every transition from one state should be matched by a transition from the other state, and that their resulting distributions are related by the lifted relation $\mathcal{R}^\dagger$. The latter involves solving the maximum flow problem of an appropriately constructed network, by taking advantage of the close relationship between our lifting operation and the above mentioned maximum flow problem.

Probabilistic testing semantics

It is natural to view the semantics of processes as being determined by their ability to pass tests; two processes are deemed to be semantically equivalent unless there is a test that can distinguish them. The actual tests used typically represent the ways in which users, or indeed other processes, can interact with the processes under examination. To formulate this idea, in Chapter 4 we set up a general testing scenario. It assumes

- a set of processes $\mathcal{P}roc$
- a set of tests $\mathcal{T}$, which can be applied to processes
- a set of outcomes O, the possible results from applying a test to a process
- a function $\mathcal{A} : \mathcal{T} \times \mathcal{P}roc \to \mathcal{P}(O)$, representing the possible results of applying a specific test to a specific process.

Here $\mathcal{P}(O)$ denotes the collection of non-empty subsets of O; so the result of applying a test T to a process P, $\mathcal{A}(T, P)$, is in general a *non-empty set* of outcomes, representing the fact that the behaviour of processes, and indeed tests, may be nondeterministic.

Moreover, some outcomes are considered better then others; for example, the application of a test may simply succeed, or it may fail, with success being better than failure. So we can assume that O is endowed with a partial order, in which $o_1 \leq o_2$ means that o_2 is a better outcome than o_1.

When comparing the results of applying tests to processes we need to compare subsets of O. There are two standard approaches to make this comparison, based on viewing these sets as elements of either the Hoare or Smyth powerdomain of O[1]. Consequently, we have two different semantic preorders for processes:

(i) For $P, Q \in \mathcal{P}roc$ let $P \sqsubseteq_{\text{may}} Q$ if for any test T and every outcome $o_1 \in \mathcal{A}(T, P)$ there exists some $o_2 \in \mathcal{A}(T, Q)$ such that $o_1 \leq o_2$.

(ii) Similarly, let $P \sqsubseteq_{\text{must}} Q$ if for any test T and every $o_2 \in \mathcal{A}(T, Q)$ there exists some $o_1 \in \mathcal{A}(T, P)$ such that $o_1 \leq o_2$.

Let us have a look at two typical instances of the set O and its associated partial order $\leq$.

1. For probabilistic processes we consider an application of a test to a process to succeed with a given probability. Thus we take as the set of outcomes the unit interval $[0, 1]$, with the standard ordering: if $0 \leq p < q \leq 1$ then succeeding with probability q is considered better than succeeding with probability p. We refer to this approach as *scalar testing*.

2. Another approach of testing, as originally proposed by Segala, employs a countable set of special actions $\Omega = \{\omega_1, \omega_2, ...\}$ to report success. When applied to probabilistic processes, this approach uses the function space $[0, 1]^\Omega$ as the set of outcomes and the standard partial order for real functions: for any $o_1, o_2 \in O$, we have $o_1 \leq o_2$ if and only if $o_1(\omega) \leq o_2(\omega)$ for every $\omega \in \Omega$. When Ω is fixed, an outcome $o \in O$ can be considered as a vector $\langle o(\omega_1), o(\omega_2), ...\rangle$, with $o(\omega_i)$ representing the probability of success observed by action ω_i. Therefore, this approach is called *vector-based testing*.

Surprisingly, it turns out that for *finitary* systems, i.e. finite-state and finitely branching systems, scalar testing is equally powerful as vector-based testing. This is the main result shown in Chapter 4. Other variants of testing approaches, such as reward testing and extremal reward testing are also discussed. They all coincide with vector-based testing as far as finitary systems are concerned.

[1]A third approach is to use the Plotkin powerdomain, which can be obtained by combining the Hoare and Smyth powerdomains.

Testing finite probabilistic processes

Chapter 5 investigates the connection between testing and simulation semantics. The simulation semantics is based on a notion of failure simulation and a notion of forward simulation; a distinguishing feature of them is to allow for the comparison of states with distributions. We say a relation $\mathcal{R} \subseteq S \times \mathcal{D}(S)$ is a *failure simulation* if $s \mathrel{\mathcal{R}} \Theta$ implies

(i) for any action α, if $s \xrightarrow{\alpha} \Delta$ then there exists some Θ' such that $\Theta \overset{\alpha}{\Longrightarrow} \Theta'$ and $\Delta \mathrel{\mathcal{R}^{\dagger}} \Theta'$

(ii) for any set of actions A, if s fails to perform any action in A, then so does some Θ' with $\Theta \overset{\tau}{\Longrightarrow} \Theta'$.

Here we write $\overset{\alpha}{\Longrightarrow}$ for a weak transition that abstracts away the internal action τ. Similarly, we define *forward simulation* by dropping the clause (ii) above.

For *finite processes*, i.e. processes whose behaviour can be described by pLTS's with finite tree structures, testing semantics is not only sound but also complete for simulation semantics. More specifically, may testing preorder coincides with forward simulation preorder and must testing preorder coincides with failure simulation preorder. Therefore, unlike the traditional (nonprobabilistic) setting, here there is no gap between testing and simulation semantics. To prove this result we make use of logical characterisations of testing preorders. For example, each state s has a characteristic formula φ_s in the sense that another state t can simulate s if and only if t satisfies φ_s. We can then turn this formula φ_s into a characteristic test T_s so that if t is not related to s via the may testing preorder then T_s is a witness test that distinguishes t from s. Similarly for the case of failure simulation and must testing. We also give a complete axiom system for the testing preorders in the finite fragment of a probabilistic process algebra.

Testing finitary probabilistic processes

In Chapter 6 we extend the results in the previous chapter from finite processes to finitary processes. Testing preorders can still be characterised as simulation preorders and admit modal characterisations. The soundness and completeness proofs inherit the general schemata from Chapter 5. However, the technicalities are much more subtle and more interesting. For example, the weak transition relation $\overset{\tau}{\Longrightarrow}$ needs to be carefully defined so as to abstract away infinitely many internal transition steps, and we make a significant use of subdistributions. A crucial topological property shown in this chapter is that from any given subdistribution, the set of stable subdistributions reachable from it by weak transitions can be finitely generated. Consider the pLTS in Figure 1 again. Both the point distribution $\overline{s_5}$ and the distribution $(\frac{1}{2}\overline{s_3} + \frac{1}{2}\overline{s_6})$ are stable and reachable from s_0. In fact, by taking linear combinations of them we obtain the set of all stable subdistributions reachable from s_0. The proof is highly non-trivial and involves techniques from *Markov decision processes* such as rewards and static policies. This result enables us to approximate coinductively defined relations by stratified inductive relations. As a consequence, if two processes behave differently we can tell them apart by a finite test.

We also introduce a notion of real-reward testing that allows for negative rewards. It turns out that real-reward may preorder is the inverse of real-reward must preorder, and vice versa. More interestingly, for finitary convergent processes, real-reward must testing preorder coincides with nonnegative-reward testing preorder.

Weak probabilistic bisimulation

In Chapter 7 we introduce a notion of weak probabilistic bisimulation simply by taking the symmetric form of the forward simulation preorder given in Chapter 6. It provides a sound and complete proof methodology for an extensional behavioural equivalence, a probabilistic variant of the traditional *reduction barbed congruence* well-known in concurrency theory.

More information

More information on this book can be found at
`www.springer.com/978-3-662-45197-7`

Acknowledgements Most of the work reported in this book was carried out during the last few years with a number of colleagues including Rob van Glabbeek, Matthew Hennessy, Carroll Morgan, Chenyi Zhang, and Wenjie Du. Thanks go also to Barry Jay, Matthew Hennessy and Carroll Morgan for having read parts of the first draft and offered useful feedback. The Springer staff have provided wonderful cooperation in the process of editing and publishing the book. In particular, I am thankful to Jane Li. My research on probabilistic concurrency theory has been sponsored by the National Natural Science Foundation of China under grants 61173033 and 61261130589. Finally, I thank Luca Aceto for his comments on this summary.

References

[1] L. Aceto, A. Ingólfsdóttir, K.G. Larsen and J. Srba. *Reactive Systems: Modelling, Specification and Verification.* Cambridge University Press, 2007.

[2] C. Baier and J.-P. Katoen. *Principles of Model Checking*. MIT Press, 2008.

[3] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science, vol. 18. Cambridge University Press, 1990.

[4] Y. Deng. *Semantics of Probabilistic Processes: An Operational Approach*. Springer, 2014.

[5] E.E. Doberkat. *Stochastic Coalgebraic Logic*. Springer, 2010.

[6] W. Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2000.

[7] M. Hennessy. *Algebraic Theory of Processes*. The MIT Press, 1988.

[8] C.A.R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[9] M. Kwiatkowska, G. Norman, D. Parker and J.J.M.M. Rutten. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*. Crm Monograph Series. American Mathematical Society, 2004.

[10] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[11] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[12] P. Panangaden. *Labelled Markov Processes*. Imperial College Press, London, 2009.

[13] A.W. Roscoe. *The Theory and Practice of Concurrency*. Prentice-Hall, 1997.

[14] A.W. Roscoe. *Understanding Concurrent Systems*. Springer, 2010.

[15] D. Sangiorgi. *Introduction to Bisimulation and Coinduction*. Cambridge University Press, 2011.

[16] D. Sangiorgi and J. Rutten (editors). *Advanced Topics in Bisimulation and Coinduction*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2011.

[17] D. Sangiorgi and D. Walker. *The π-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

E uropean

A ssociation for

T heoretical

C omputer

S cience

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
- Copenhagen, Denmark 2014
- Kyoto, Japan 2014

(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.

BNP Paribas Fortis Bank, Driekoningenstraat 122, 2600 Berchem - Antwerpen, Belgium
Account number: 220–0596350–30–01130
IBAN code: BE 15 2200 5963 5030, *SWIFT code: GEBABE BB 18A*