Lecture Notes in Computer Science 10608

More information about this series at http://www.springer.com/series/7407

Mohammad Reza Mousavi · Jiří Sgall (Eds.)

Topics in Theoretical Computer Science

Second IFIP WG 1.8 International Conference, TTCS 2017
Tehran, Iran, September 12–14, 2017
Proceedings

Springer

Editors
Mohammad Reza Mousavi
University of Leicester
Leicester
UK

Jiří Sgall
Charles University
Prague
Czech Republic

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-319-68952-4 ISBN 978-3-319-68953-1 (eBook)
DOI 10.1007/978-3-319-68953-1

Library of Congress Control Number: 2017956068

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

Welcome to the Second IFIP International Conference on Topics in Theoretical Computer Science (TTCS 2017), held during September 12–14, 2017, at the School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran.

This volume contains the papers accepted for presentation at TTCS 2017. For this edition of TTCS, we received 20 submissions from 10 different countries. An international Program Committee comprising 32 leading scientists from 13 countries reviewed the papers thoroughly providing on average four review reports for each paper. We accepted eight submissions, which translates into 40% of all submissions. This means that the process was selective and only high-quality papers were accepted. The program also includes four invited talks by the following world-renowned computer scientists:

- Mahdi Cheraghchi, Imperial College, UK
- Łukasz Jeż, University of Wrocław, Poland
- Jaco van de Pol, University of Twente, The Netherlands
- Peter Csaba Ölveczky, University of Oslo, Norway

Additionally, the program features two talks and one tutorial in the PhD Forum, which are not included in the proceedings.

We thank IPM, and in particular the Organizing Committee, for having provided various facilities and for their generous support. We are also grateful to our Program Committee for their professional and hard work in providing expert review reports and thorough discussions leading to a very interesting and strong program.

We also acknowledge the excellent facilities provided by the EasyChair system, which were crucial in managing the process of submission, selection, revision, and publication of the manuscripts included in these proceedings.

September 2017

Mohammad Reza Mousavi
Jiří Sgall

The original version of this book was revised: The paper starting on p. 41 was moved from the topical section heading "Logic, Semantics, and Programming Theory" to "Algorithms and Complexity". The erratum to this book is available at https://doi.org/10.1007/978-3-319-68953-1_10

Organization

General Chair

Hamid Sarbazi-azad	IPM, Iran; Sharif University of Technology, Iran

Local Organization Chair

Hamid Reza Shahrabi	IPM, Iran

Publicity Chair

Mahmoud Shirazi	IPM, Iran

Program Committee

Farhad Arbab	CWI and Leiden University, The Netherlands
Amitabha Bagchi	Indian Institute of Technology, Delhi, India
Sam Buss	University of California, San Diego, USA
Jarek Byrka	University of Wroclaw, Poland
Ilaria Castellani	Inria Sophia Antipolis, France
Amir Daneshgar	Sharif University of Technology, Iran
Anna Gal	University of Texas at Austin, USA
Fatemeh Ghassemi	University of Tehran, Iran
Mohammad T. Hajiaghayi	University of Maryland, College Park, USA
Hossein Hojjat	Rochester Institute of Technology, Rochester, New York, USA
Mohammad Izadi	Sharif University of Technology, Iran
Sung-Shik T.Q. Jongmans	Open University of The Netherlands, Imperial College London, UK
Ramtin Khosravi	University of Tehran, Iran
Jan Kretinsky	Masaryk University, Czech Republic
Amit Kumar	IIT Delhi, India
Bas Luttik	Eindhoven University of Technology, The Netherlands
Mohammad Mahmoody	University of Virginia
Larry Moss	Indiana University, USA
Mohammad Reza Mousavi	University of Leicester, UK
Rolf Niedermeier	TU Berlin, Germany
Giuseppe Persiano	Università degli Studi di Salerno, Italy
Jörg-Rüdiger Sack	Carleton University, Canada
Mehrnoosh Sadrzadeh	Queen Mary University of London, UK
Rahul Santhanam	University of Edinburgh, UK
Gerardo Schneider	Chalmers, University of Gothenburg, Sweden

Jiří Sgall	Computer Science Institute of Charles University, Czech Republic
Subodh Sharma	Indian Institute of Technology Delhi, India
Mirco Tribastone	IMT Institute for Advanced Studies Lucca, Italy
Kazunori Ueda	Waseda University, Japan
Vijay Vazirani	Georgia Tech, USA
Gerhard Woeginger	RWTH Aachen, Germany
Hamid Zarrabi-Zadeh	Sharif University of Technology, Iran

Additional Reviewers

Abd Alrahman, Yehia
Bagga, Divyanshu
Baharifard, Fatemeh
Bentert, Matthias
Klinz, Bettina
Krämer, Julia
Maggi, Alessandro
Meggendorfer, Tobias
Mirjalali, Kian
Molter, Hendrik
Neruda, Roman
van der Woude, Jaap
van Oostrom, Vincent
Vegh, Laszlo
Łukaszewski, Andrzej

Abstracts of Invited Talks

The Coding Lens in Explicit Constructions

Mahdi Cheraghchi

Department of Computing, Imperial College London, UK
m.cheraghchi@imperial.ac.uk

Abstract. The theory of error-correcting codes, originally developed as a fundamental technique for a systematic study of communications systems, has served as a pivotal tool in major areas of mathematics, computer science and electrical engineering. Understanding problems through a "coding lens" has consistently led to breakthroughs in a wide spectrum of research areas, often seemingly foreign from coding theory, including discrete mathematics, geometry, cryptography, signal processing, algorithms and complexity, to name a few. This talk will focus on the role of coding theory in pseudorandomness, and particularly, explicit construction problems in sparse recovery and signal processing.

Online Packet Scheduling

Łukasz Jeż

Institute of Computer Science, University of Wrocław, Poland

Packet Scheduling, also known as *Buffer Management with Bounded Delay*, is a problem motivated by managing the buffer of a network switch or router (hence the latter name), but also an elementary example of a job scheduling problem: a job j has unit processing time ($p_j = 1$), arbitrary weight w_j, as well as arbitrary release time $r_j \in \mathbb{Z}$ and deadline $d_j \in \mathbb{Z}$ such that $r_j < d_j$. A given set of such jobs is to be scheduled on a single machine so as to maximize the total weight of jobs completed by their deadlines.

The *online* variant is of particular interest, given the motivation: Think of an algorithm that has to schedule jobs on the fly, at time slot t knowing only those (and their parameters) which were already released. From the algorithm's perspective, the computation proceeds in rounds, corresponding to time slots; in round t, the following happen: first, jobs with deadlines t expire (and are since ignored), then any set of new jobs with release time t may arrive, and finally the algorithm can choose one pending job; next, this job is completed, yielding reward equal to its weight, and the computation proceeds to the next round.

Though an online algorithm knows nothing of the future jobs arrivals, we require worst-case performance guarantees on the complete instance when it ends. Specifically, we say an algorithm is *R-competitive* if on *every* instance I its gain is at least a $1/R$ fraction of the optimum gain on I.

It is easy to give bounds on the competitive ratio: an upper bound of 2 is attained by a simple greedy algorithm that chooses the heaviest pending job in each slot; for a lower bound, it suffices to consider an instance merely two slots long. These can of course be improved: a careful analysis of a natural generalization of the lower bound instance yields a lower bound of $\varphi \approx 1.618$, which is the best known. Better algorithms, with rather involved analyses, are also known: the best, dating back to 2007, is 1.828-competitive.

These bounds do not match, despite simple problem statement and significant effort since the early 2000s. One consequence is a number of restricted classes of instances that were considered. I will survey known results, on both deterministic and randomized algorithms, presenting some of them in more detail.

We will start by noting that packet scheduling is a special case of maximum-weight matching problem, where the jobs and the time slots form the two partitions, and each job j is connected by an edge of weight w_j to each of the time slots in $[r_j, d_j) \cap \mathbb{Z}$. This has twofold implications: Firstly, online algorithms designed for the matching problem apply, one of them (randomized) in fact the best known even for our special case. Secondly, optimal offline algorithms, though not our primary interest, grant structural insight into optimal schedules, helping in the online setting too.

Parallel Algorithms for Model Checking

Jaco van de Pol

University of Twente, Formal Methods and Tools, Enschede, The Netherlands
J.C.vandePol@utwente.nl

Model checking [1, 5] is an automated verification procedure, which checks that a model of a system satisfies certain properties. These properties are typically expressed in some temporal logic, like LTL and CTL. Algorithms for LTL model checking (linear time logic) are based on automata theory and graph algorithms, while algorithms for CTL (computation tree logic) are based on fixed-point computations and set operations.

The basic model checking procedures examine the state space of a system exhaustively, which grows exponentially in the number of variables or parallel components. Scalability of model checking is achieved by clever abstractions (for instance counter-example guided abstraction refinement), clever algorithms (for instance partial-order reduction), clever data-structures (for instance binary decision diagrams) and, finally, clever use of hardware resources, for instance algorithms for distributed and multi-core computers.

This invited lecture will provide a number of highlights of our research in the last decade on high-performance model checking, as it is implemented in the open source LTSmin tool set[1] [10], focusing on the algorithms and datastructures in its multi-core tools.

A lock-free, scalable hash-table maintains a globally shared set of already visited state vectors. Using this, parallel workers can semi-independently explore different parts of the state space, still ensuring that every state will be explored exactly once. Our implementation proved to scale linearly on tens of processors [12].

Parallel algorithms for NDFS. Nested Depth-First Search [6] is a linear-time algorithm to detect accepting cycles in Büchi automata. LTL model checking can be reduced to the emptiness problem of Büchi automata, i.e. the absence of accepting cycles. We introduced a parallel version of this algorithm [9], despite the fact that Depth-First Search is hard to parallelize. Our multi-core implementation is compatible with important state space reduction techniques, in particular state compression and partial-order reduction [11, 15] and generalizes to timed automata [13].

A multi-core library for Decision Diagrams, called Sylvan [7]. Binary Decision Diagrams (BDD) have been introduced as concise representations of sets of Boolean vectors. The CTL model checking operations can be expressed directly on the BDD representation [4]. Sylvan provides a parallel implementation of BDD operations for shared-memory, multi-core processors. We also provided successful experiments on

1 http://ltsmin.utwente.nl, https://github.com/utwente-fmt/ltsmin.

distributed BDDs over a cluster of multi-core computer servers [14]. Besides BDDs, Sylvan also supports Multi-way and Multi-terminal Decision Diagrams.

Multi-core algorithms to detect Strongly Connected Components. An alternative model-checking algorithm is based on the decomposition and analysis of Strongly Connected Components (SCCs). We have implemented a parallel version of Dijkstra's SCC algorithm [2, 8]. It forms the basis of model checking LTL using generalized Büchi and Rabin automata [3]. SCCs are also useful for model checking with fairness, probabilistic model checking, and implementing partial-order reduction.

References

1. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008)
2. Bloemen, V., Laarman, A., van de Pol, J.: Multi-core on-the-fly SCC decomposition. In: PPoPP'16, pp. 8:1–8:12. ACM (2016)
3. Bloemen, V., Duret-Lutz, A., van de Pol, J.: Explicit state model checking with generalized Büchi and Rabin automata. In: SPIN'17: Model Checking of Software. ACM SIGSOFT (2017)
4. Burch, J.R., Clarke, E.M., McMillan, K.L., Dill, D.L., Hwang, L.J.: Symbolic model checking: 10^20 states and beyond. Inf. Comput. **98**(2), 142–170 (1992)
5. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. The MIT Press (1999)
6. Courcoubetis, C., Vardi, M.Y., Wolper, P., Yannakakis, M.: Memory-efficient algorithm for the verification of temporal properties. Formal Methods Syst. Des. **1**, 275–288 (1992)
7. van Dijk, T., van de Pol, J.: Sylvan: multi-core framework for decision diagrams. Int. J. Softw. Tools Technol. Transfer (2016)
8. Dijkstra, E.W.: A Discipline of Programming. Prentice-Hall (1976)
9. Evangelista, S., Laarman, A., Petrucci, L., van de Pol, J.: Chakraborty, S., Mukund, M. (eds.) ATVA 2012. LNCS, vol. 7561, pp. 269–283. Springer, Heidelberg (2012)
10. Kant, G., Laarman, A., Meijer, J., van de Pol, J., Blom, S., van Dijk, T.: LTSmin: high-performance language-independent model checking. In: Baier, C., Tinelli, C. (eds.) TACAS 2015. LNCS, vol. 9035, pp. 692–707. Springer, Heidelberg (2015)
11. Laarman, A., Pater, E., van de Pol, J., Hansen, H.: Guard-based partial-order reduction. STTT **18**(4), 427–448 (2016)
12. Laarman, A., van de Pol, J., Weber, M.: Boosting multi-core reachability performance with shared hash tables. In: FMCAD 2010, pp. 247–255 (2010)
13. Laarman, A.W., Olesen, M.C., Dalsgaard, A.E., Larsen, K.G., van de Pol, J.C.: Multi-core emptiness checking of timed Büchi automata using inclusion abstraction. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 968–983. Springer, Heidelberg (2013)
14. Oortwijn, W., van Dijk, T., van de Pol, J.: Distributed binary decision diagrams for symbolic reachability. In: SPIN'17: Model Checking of Software. ACM SIGSOFT (2017)
15. Valmari, A.: A stubborn attack on state explosion. Formal Methods Syst. Des. **1**(4), 297–322 (1992)

Design and Validation of Cloud Storage Systems Using Formal Methods

Peter Csaba Ölveczky

University of Oslo, Oslo, Norway
peterol@ifi.uio.no

Abstract. To deal with large amounts of data while offering high availability and throughput and low latency, cloud computing systems rely on distributed, partitioned, and replicated data stores. Such cloud storage systems are complex software artifacts that are very hard to design and analyze. Formal specification and model checking should therefore be beneficial during their design and validation. In particular, I propose rewriting logic and its accompanying Maude tools as a suitable framework for formally specifying and analyzing both the correctness and the performance of cloud storage systems. This abstract of an invited talk gives a short overview of the use of rewriting logic at the University of Illinois' Assured Cloud Computing center on industrial data stores such as Google's Megastore and Facebook/Apache's Cassandra. I also briefly summarize the experiences of the use of a different formal method for similar purposes by engineers at Amazon Web Services.

Contents