Susanne Albers
Alberto Marchetti-Spaccamela
Yossi Matias
Sotiris Nikoletseas
Wolfgang Thomas (Eds.)

Automata, Languages and Programming

36th International Colloquium, ICALP 2009
Rhodes, Greece, July 2009
Proceedings, Part II

2 Part II

EATCS

Springer

ARCoSS

LNCS 5556

Advanced Research in Computing and Software Science

Lecture Notes in Computer Science 5556

Advanced Research in Computing and Software Science

Subline of Lectures Notes in Computer Science

Susanne Albers
Alberto Marchetti-Spaccamela
Yossi Matias
Sotiris Nikoletseas
Wolfgang Thomas (Eds.)

Automata, Languages and Programming

36th International Colloquium, ICALP 2009
Rhodes, Greece, July 5-12, 2009
Proceedings, Part II

Springer

Volume Editors

Susanne Albers
University of Freiburg, Department of Computer Science
Georges Köhler Allee 79, 79110, Freiburg, Germany
E-mail: salbers@informatik.uni-freiburg.de

Alberto Marchetti-Spaccamela
Sapienza University of Rome
Department of Computer and Systems Sciences
Via Ariosto 25, 00184 Roma, Italy
E-mail: alberto@dis.uniroma1.it

Yossi Matias
Tel Aviv University, School of Computer Science
Google R&D Center, Tel Aviv 69978, Israel
E-mail: matias@cs.tau.ac.il

Sotiris Nikoletseas
University of Patras and CTI
N. Kazantzaki Street 1, 26504 Rion, Patras, Greece
E-mail: nikole@cti.gr

Wolfgang Thomas
RWTH Aachen, Lehrstuhl Informatik 7
Ahornstraße 55, 52074 Aachen, Germany
E-mail: thomas@informatik.rwth-aachen.de

Library of Congress Control Number: 2009929832

CR Subject Classification (1998): F.3, F.2, G.1, G.2, D.2.8, F.1

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

ISSN 0302-9743
ISBN-10 3-642-02929-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02929-5 Springer Berlin Heidelberg New York

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12715596 06/3180 5 4 3 2 1 0

Preface

ICALP 2009, the 36th edition of the International Colloquium on Automata, Languages and Programming, was held on the island of Rhodes, July 6–10, 2009. ICALP is a series of annual conferences of the European Association for Theoretical Computer Science (EATCS) which first took place in 1972. This year, the ICALP program consisted of the established track A (focusing on algorithms, complexity and games) and track B (focusing on logic, automata, semantics and theory of programming), and of the recently introduced track C (in 2009 focusing on foundations of networked computation).

In response to the call for papers, the Program Committee received 370 submissions: 223 for track A, 84 for track B and 63 for track C. Out of these, 108 papers were selected for inclusion in the scientific program: 62 papers for track A, 24 for track B and 22 for track C. The selection was made by the Program Committees based on originality, quality, and relevance to theoretical computer science. The quality of the manuscripts was very high indeed, and many deserving papers could not be selected.

ICALP 2009 consisted of five invited lectures and the contributed papers. This volume of the proceedings contains all contributed papers presented in track B and track C together with the papers by the invited speakers Georg Gottlob (University of Oxford), Tom Henzinger (École Polytechnique Fédérale de Lausanne), and Noam Nisan (Google, Tel Aviv, and Hebrew University). A companion volume contains all contributed papers presented at the conference in track A, together with the papers by the invited speakers Kurt Mehlhorn (Max-Planck-Institut für Informatik, Saarbrücken) and Christos Papadimitriou (University of California at Berkeley).

The following workshops were held as satellite events of ICALP 2009:

ALGOSENSORS 2009—5th International Workshop on Algorithmic Aspects of Wireless Sensor Networks
DCM 2009—5th International Workshop on Developments in Computational Models
FOCLASA 2009—8th International Workshop on Foundations of Coordination Languages and Software Architectures
QUANTLOG 2009—Workshop on Quantitative Logics 2009

We wish to thank all authors who submitted extended abstracts for consideration, the Program Committees for their scholarly effort, and all referees who assisted the Program Committees in the evaluation process.

Thanks are due to the sponsors (Ministry of National Education and Religious Affairs of Greece, Research Academic Computer Technology Institute (CTI), Piraeus Bank) for their support, and to the Research Academic

Computer Technology Institute (CTI) for the local organization. We are also grateful to all members of the Organizing Committee.

Thanks also to Andrei Voronkov for his help with the conference management system EasyChair, which was used in handling the submissions and the electronic PC meeting as well as in assisting in the assembly of the proceedings.

April 2009

Susanne Albers
Alberto Marchetti Spaccamela
Yossi Matias
Paul G. Spirakis
Wolfgang Thomas

Organization

Program Committee

Track A

Susanne Albers	University of Freiburg, Germany (Chair)
Gerth Brodal	University of Aarhus, Denmark
Martin Dyer	University of Leeds, UK
Irene Finocchi	University of Rome “La Sapienza", Italy
Anna Gal	University of Texas at Austin, USA
Naveen Garg	IIT Delhi, India
Raffaele Giancarlo	University of Palermo, Italy
Andrew Goldberg	Microsoft Research, Silicon Valley, USA
Mordecai Golin	Hong Kong University
Michel Habib	LIAFA, Paris 7, France
Thore Husfeldt	Lund University, Sweden
Kazuo Iwama	University of Kyoto, Japan
Howard Karloff	AT&T Labs, USA
Yishay Mansour	Tel Aviv University and Google, Israel
Jiri Matoušek	Charles University, Prague, Czech Republic
Marios Mavronicolas	University of Cyprus, Cyprus
Piotr Sankowski	University of Warsaw and ETH Zurich, Switzerland
Raimund Seidel	University of Saarbrücken, Germany
Paul Spirakis	CTI and University of Patras, Greece
Dorothea Wagner	University of Karlsruhe, Germany
Peter Widmayer	ETH Zurich, Switzerland
Ronald de Wolf	CWI Amsterdam, The Netherlands

Track B

Albert Atserias	Universitat Politecnica de Catalunya, Barcelona, Spain
Jos Baeten	Eindhoven University of Technology, The Netherlands
Gilles Barthe	IMDEA Software, Madrid, Spain
Mikolaj Bojanczyk	Warsaw University, Poland
Christian Choffrut	University Denis Diderot, Paris, France
Thierry Coquand	Göteborg University, Sweden
Roberto di Cosmo	University Denis Diderot, Paris, France
Kousha Etessami	University of Edinburgh, Scotland, UK
Dexter Kozen	Cornell University, USA

Stephan Kreutzer	Oxford University, UK
Orna Kupferman	Hebrew University, Israel
Kim Guldstrand Larsen	Aalborg University, Denmark
Dale Miller	Ecole Polytechnique, Palaiseau, France
Markus Müller-Olm	University of Münster, Germany
Anca Muscholl	University Bordeaux 1, France
R. Ramanujam	Institute of Mathematical Sciences, Chennai, India
Simona Ronchi Della Rocca	University of Turin, Italy
Jan Rutten	CWI, Amsterdam, The Netherlands
Vladimiro Sassone	University of Southampton, UK
Peter Sewell	University of Cambridge, UK
Howard Straubing	Boston College, USA
Wolfgang Thomas	RWTH Aachen University, Germany (Chair)

Track C

Hagit Attiya	Technion, Israel
Andrei Broder	Yahoo, USA
Xiaotie Deng	City University of Hong Kong
Danny Dolev	Hebrew University, Israel
Michele Flammini	University of L'Aquila, Italy
Pierre Fraigniaud	CNRS, Paris, France
Ashish Goel	University of Stanford, USA
Matthew Hennessy	Trinity College Dublin, Ireland
Kohei Honda	University of London, UK
Robert Kleinberg	Cornell University, USA
Elias Koutsoupias	University of Athens, Greece
Alberto Marchetti Spaccamela	University of Rome “La Sapienza", Italy (Co-chair)
Yossi Matias	Google and Tel Aviv University, Israel (Co-chair)
Silvio Micali	MIT, USA
Muthu Muthukrishnan	Google, NY, USA
Moni Naor	Weizmann Institute, Israel
Mogens Nielsen	University of Aarhus, Denmark
Harald Raecke	University of Warwick, UK
Jose Rolim	University of Geneva, Switzerland
Christian Schindelhauer	University of Freiburg, Germany
Roger Wattenhofer	ETH Zurich, Switzerland
Martin Wirsing	University of Munich, Germany

Organizing Committee

- Paul G. Spirakis, Research Academic Computer Technology Institute and University of Patras, Greece (Conference Chair)
- Elias Koutsoupias, Department of Informatics and Telecommunications, University of Athens, Greece (Conference Chair)
- Christos Kaklamanis, Research Academic Computer Technology Institute and University of Patras, Greece (Conference Chair)
- Christos D. Zaroliagis, Research Academic Computer Technology Institute and University of Patras, Greece (Workshops Chair)
- Sotiris Nikoletseas, Research Academic Computer Technology Institute and University of Patras, Greece (Proceedings Chair)
- Ioannis Chatzigiannakis, Research Academic Computer Technology Institute and University of Patras, Greece (Publicity Chair)
- Rozina Efstathiadou, Research Academic Computer Technology Institute, Greece (Finance Chair)
- Lena Gourdoupi, Research Academic Computer Technology Institute, Greece (Conference Secretariat)

Referees (Track B)

Luca Aceto
Jade Alglave
Rajeev Alur
Roberto Amadio
Benjamin Aminof
Daniel Andersson
Suzana Andova
David Baelde
Anindya Banerjee
Franco Barbanera
Alberto Bertoni
Dietmar Berwanger
Jean-Camille Birget
Sandrine Blazy
Achim Blumensath
Luc Boasson
Udi Boker
Guillaume Bonfante
Henning Bordihn
Ahmed Bouajjani
Julian Bradfield
Tomas Brazdil
Arnaud Carayol
Olivier Carton
Swarat Chaudhuri
Tom Chothia
Thomas Colcombet
Bruno Courcelle
Pedro R. D'Argenio
Ferruccio Damiani
Norman Danner
Anuj Dawar
Ehab El-Salamouny
Uli Fahrenberg
Dana Fisman
Serge Grigorieff
Murdoch Gabbay
Giorgio Ghelli
Hugo Gimbert
Peter Habermehl
Rémy Haemmerlé
Sardaouna Hamadou
Tero Harju
Hendrik Jan Hoogeboom
Florian Horn
Pierre Hyvernat
Florent Jacquemard
Petr Jancar
Peter Jeavons
Daniel Kirsten
Hanna Klaudel
Andrei Krokhin
Ruurd Kuiper
K. Narayan Kumar
Cesar Kunz
Christof Löding
Peter Lammich
Robby Lampert
Cosimo Laneve
Slawomir Lasota
Pierre Lescanne
Kamal Lodaya
Alessio Lomuscio
Etienne Lozes
Yoad Lustig
Bas Luttik
Assia Mahboubi
Amaldev Manuel
Claude Marché
Jean-Yves Marion
Nicolas Markey

Jasen Markovski
Wim Martens
Richard Mayr
Damiano Mazza
Stephan Merz
Ron van der Meyden
Peter Bro Miltersen
Alexandre Miquel
Mohammad Reza Mousavi
Madhavan Mukund
Filip Murlak
Uwe Mönnich
Francesco Zappa Nardelli
Francois Nicolas
Damian Niwiński
Alexander Okhotin
Paulo Oliva
Ghassan Oreiby
Martin Otto
Joel Ouaknine
Luca Padovani
Soumya Paul
Gustavo Petri
Laure Petrucci
Mauro Piccolo
Jean-Eric Pin
Michael Pinsker
Detlef Plump
Benoit Razet
Yann Regis-Gianas
Colin Riba
Harald Ruess
Cesar Sanchez
Davide Sangiorgi
Philippe Schnoebelen
Helmut Seidl
Olivier Serre
Mihaela Sighireanu
Jan-Georg Smaus
Ana Sokolova
Jiri Srba
Sam Staton
Wolfgang Steiner
Colin Stirling
S. P. Suresh
Grégoire Sutre
Pascal Tesson
Claus Thrane
Paul van Tilburg
Stefano Tonetta
Paolo Tranquilli
Yih-Kuen Tsay
Michael Ummels
Daniele Varacca
Yde Venema
Luca Vercelli
Kumar Neeraj Verma
Aymeric Vincent
Erik de Vink
Marc Voorhoeve
Pascal Weil
Alexander Wenner
Thomas Wilke
Toby Wilkinson
Tim Willemse
James Worrell
Qiqi Yan
Konrad Zdanowski
Noam Zeilberger
Marc Zeitoun
Ugo de'Liguoro
Zoltan Esik

Referees (Track C)

Hakob Aslanyan
James Aspnes
Chen Avin
Hervé Baumann
Luca Becchetti
Alastair Beresford
Martin Berger
Vittorio Bilò
Vincenzo Bonifaci
Vladimir Braverman
Doina Bucur
Keren Censor
Ning Chen
George Christodoulou
Ivan Damgaard
Camil Demetrescu
Keren Dong
Ye Du
Raphael Eidenbenz
Amos Fiat
Wan Fokkink
Adrian Francalanza
Paolo Giulio Franciosa
Yiannis Giannakopoulos
Iftach Haitner
Tzvika Hartman
Ezra Hoch
Bracha Hod
Florian Huc
Yuval Ishai
Riko Jacob
Christian Damsgaard Jensen
Jonathan Katz
Gabriel Kliot
Xavier Koegler
Adrian Kosowski
Kishore Kothapalli
Kasper Dalgaard Larsen
Stefano Leonardi
Pierre Leone
Zvi Lotker
Fabien Mathieu
Massimo Merro
Peter Bro Miltersen
Gianpiero Monaco
Luca Moscardelli
Alfredo Navarra
Sotiris Nikoletseas
Claudio Orlandi
Rafael Ostrowsky

Aris Pagourtzis
Arpita Patra
David Peleg
Benny Pinkas
Davide Prandi
Guido Proietti
Omer Reingold
Tzachy Reinman
Tamara Resk
Amir Rothschild
Gil Segev
Mordechai Shalom
Lakshminarayanan Subramanian
Tomas Toft
Tigran Tonoyan
Shailesh Vaya
Angelina Vidali
Edsko de Vries
Jianping Wang
Guomin Yang
Jiajin Yu
Jie Zhang

Sponsoring Organizations

- Ministry of National Education and Religious Affairs of Greece
- Research Academic Computer Technology Institute (CTI)
- Piraeus Bank
- Rigorous Mathematical Connections between the Theory of Computation and Statistical Physics (RIMACO), European Research Council (ERC) Starting Grant
- Algorithmic Principles for Building Efficient Overlay Computers (AEOLUS) Project, EU/FET/Global Computing
- Papasotiriou Books and more
- Google

Table of Contents – Part II

Track B: Invited Lectures

Track B: Contributed Papers

Track C: Invited Lecture

Track C: Contributed Papers

Table of Contents – Part I

Invited Lectures

Contributed Papers

A Survey of Stochastic Games with Limsup and Liminf Objectives*

Krishnendu Chatterjee[1], Laurent Doyen[2,**], and Thomas A. Henzinger[3]

[1] Institute of Science and Technology (IST), Austria
[2] Université Libre de Bruxelles (ULB), Belgium
[3] École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

Abstract. A stochastic game is a two-player game played on a graph, where in each state the successor is chosen either by one of the players, or according to a probability distribution. We survey stochastic games with limsup and liminf objectives. A real-valued reward is assigned to each state, and the value of an infinite path is the limsup (resp. liminf) of all rewards along the path. The value of a stochastic game is the maximal expected value of an infinite path that can be achieved by resolving the decisions of the first player. We present the complexity of computing values of stochastic games and their subclasses, and the complexity of optimal strategies in such games.

1 Introduction

A *turn-based stochastic game* is played on a finite graph with three types of states: in player-1 states, the first player chooses a successor state from a given set of outgoing edges; in player-2 states, the second player chooses a successor state from a given set of outgoing edges; and in probabilistic states, the successor state is chosen according to a given probability distribution. The game results in an infinite path through the graph. Every such path is assigned a real value, and the objective of player 1 is to resolve her choices so as to maximize the expected value of the resulting path, while the objective of player 2 is to minimize the expected value. If the function that assigns values to infinite paths is a Borel function (in the Cantor topology on infinite paths), then the game is determined [17]: the maximal expected value achievable by player 1 is equal to the minimal expected value achievable by player 2, and it is called the *value* of the game.

There are several canonical functions for assigning values to infinite paths. If each state is given a reward, then the *max* (resp. *min*) function chooses the

* This research was supported in part by the Swiss National Science Foundation under the Indo-Swiss Joint Research Programme, by the European Network of Excellence on Embedded Systems Design (ArtistDesign), by the European projects COMBEST, Quasimodo, Gasics, by the PAI program Moves funded by the Belgian Federal Government, and by the CFV (Federated Center in Verification) funded by the F.R.S.-FNRS.

** Postdoctoral researcher of the F.R.S.-FNRS.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 1–15, 2009.

maximum (resp. minimum) of the infinitely many rewards along a path; the *limsup* (resp. *liminf*) function chooses the limsup (resp. liminf) of the infinitely many rewards; and the *limit-average* function chooses the long-run average of the rewards. The max and min functions are Borel level-1 functions, whereas limsup and liminf are Borel level-2 functions, and limit-average is a Borel level-3 function. Stochastic games with the limit-average condition (also called *mean-payoff* objective) have been studied extensively in the literature [11,14,20,1,15]. The study of stochastic games with max and min conditions [2,4], as well as limsup and liminf conditions [5,13,16], is more recent. The max and min functions are natural generalizations of reachability and safety objectives in the non-quantitative setting, while the limsup and liminf functions are natural generalizations of Büchi and coBüchi objectives [18,19].

In this paper, we survey algorithms and computational complexity results for computing values of turn-based stochastic games and with limsup and liminf objectives. We organize the results according to the different classes of game graphs. We successively consider (*i*) 1-player game graphs, where all states belong to one player, (*ii*) 2-player game graphs, in which there is no probabilistic state, (*iii*) $1^1\!/_2$-player game graphs (or Markov decision processes), in which there is no player-2 state, and (*iv*) $2^1\!/_2$-player game graphs, which is the general case. Along with surveying known results in the field, we also present two algorithmic improvements over the literature for the solution of 1-player and 2-player game graphs with limsup and liminf objectives. We show that 1-player game graphs with $\boldsymbol{n}$ states and $\boldsymbol{m}$ edges can be solved in time $O(\boldsymbol{n}+\boldsymbol{m})$ while the previously known algorithm of [2] runs in time $O(\boldsymbol{n}\log\boldsymbol{n}+\boldsymbol{m})$; for 2-player game graphs, our algorithm runs in time $O(\boldsymbol{m}\boldsymbol{n}\log\boldsymbol{n})$ as compared to the previously known algorithm of [2] that runs in time $O(\boldsymbol{m}\boldsymbol{n}^2)$.

2 Definitions

We consider the class of turn-based stochastic games and some of its subclasses.

Game graphs. A *turn-based probabilistic game graph* (*$2^1\!/_2$-player game graph*) $G = ((S,E),(S_1,S_2,S_P),\delta)$ consists of a finite directed graph (S,E), a partition (S_1, S_2, S_P) of the finite set S of states, and a probabilistic transition function δ: $S_P \to \mathcal{D}(S)$, where $\mathcal{D}(S)$ denotes the set of probability distributions over the state space S. The states in S_1 are the *player*-1 states, where player 1 decides the successor state; the states in S_2 are the *player*-2 states, where player 2 decides the successor state; and the states in S_P are the *probabilistic* states, where the successor state is chosen according to the probabilistic transition function δ. We assume that for $s \in S_P$ and $t \in S$, we have $(s,t) \in E$ iff $\delta(s)(t) > 0$, and we often write $\delta(s,t)$ for $\delta(s)(t)$. For technical convenience we assume that every state in the graph (S,E) has at least one outgoing edge. For a state $s \in S$, we write $E(s)$ to denote the set $\{\, t \in S \mid (s,t) \in E \,\}$ of possible successors.

Subclasses of stochastic games. The *turn-based deterministic game graphs* (*2-player game graphs*) are the special case of the $2^1\!/_2$-player game graphs with

$S_P = \emptyset$. The *Markov decision processes* ($1^1\!/\!_2$*-player game graphs*) are the special case of the $2^1\!/\!_2$-player game graphs with $S_1 = \emptyset$ or $S_2 = \emptyset$. We refer to the MDPs with $S_2 = \emptyset$ as *player*-1 MDPs, and to the MDPs with $S_1 = \emptyset$ as *player*-2 MDPs. The *transition systems* (1*-player game graphs*) are the special case of $2^1\!/\!_2$-player game graphs with (a) $S_P = \emptyset$ and (b) either $S_1 = \emptyset$ or $S_2 = \emptyset$. Observe that 1-player game graphs are subclasses of both 2-player game graphs and $1^1\!/\!_2$-player game graphs.

Size of graph. Given a game graph $G = ((S,E),(S_1,S_2,S_P),\delta)$ we use the following notations: (a) we denote by $\boldsymbol{n}$ the number of states, i.e., $\boldsymbol{n} = |S|$; (b) we denote by $\boldsymbol{m}$ the number of edges, i.e., $\boldsymbol{m} = |E|$; (c) we denote by Δ the maximum out-degree of the graph, i.e., $\Delta = \max_{s \in S} |E(s)|$.

Plays and strategies. An infinite path, or a *play*, of the game graph G is an infinite sequence $\omega = \langle s_0, s_1, s_2, \ldots \rangle$ of states such that $(s_k, s_{k+1}) \in E$ for all $k \in \mathbb{N}$. We write Ω for the set of all plays, and for a state $s \in S$, we write $\Omega_s \subseteq \Omega$ for the set of plays that start from the state s. A *strategy* for player 1 is a function $\sigma\colon S^* \cdot S_1 \to \mathcal{D}(S)$ that assigns a probability distribution to all finite sequences $w \in S^* \cdot S_1$ of states ending in a player-1 state (the sequence w represents a prefix of a play). Player 1 follows the strategy σ if in each player-1 move, given that the current history of the game is $w \in S^* \cdot S_1$, she chooses the next state according to the probability distribution $\sigma(w)$. A strategy must prescribe only available moves, i.e., for all $w \in S^*$, $s \in S_1$, and $t \in S$, if $\sigma(w \cdot s)(t) > 0$, then $(s,t) \in E$. The strategies for player 2 are defined analogously. We denote by Σ and Π the set of all strategies for player 1 and player 2, respectively.

Once a starting state $s \in S$ and strategies $\sigma \in \Sigma$ and $\pi \in \Pi$ for the two players are fixed, the outcome of the game is a random walk $\omega_s^{\sigma,\pi}$ for which the probabilities of events are uniquely defined, where an *event* $\mathcal{A} \subseteq \Omega$ is a measurable set of plays. For a state $s \in S$ and an event $\mathcal{A} \subseteq \Omega$, we write $\Pr_s^{\sigma,\pi}(\mathcal{A})$ for the probability that a play belongs to $\mathcal{A}$ if the game starts from the state s and the players follow the strategies σ and π, respectively. For a measurable function $f : \Omega \to \mathrm{I\!R}$ we denote by $\mathbb{E}_s^{\sigma,\pi}[f]$ the *expectation* of the function f under the probability measure $\Pr_s^{\sigma,\pi}(\cdot)$.

Strategies that do not use randomization are called pure. A player-1 strategy σ is *pure* if for all $w \in S^*$ and $s \in S_1$, there is a state $t \in S$ such that $\sigma(w \cdot s)(t) = 1$. A *memoryless* player-1 strategy does not depend on the history of the play but only on the current state; i.e., for all $w, w' \in S^*$ and for all $s \in S_1$ we have $\sigma(w \cdot s) = \sigma(w' \cdot s)$. A memoryless strategy for player 1 can be represented as a function $\sigma\colon S_1 \to \mathcal{D}(S)$. A *pure memoryless strategy* is a strategy that is both pure and memoryless. A pure memoryless strategy for player 1 can be represented as a function $\sigma\colon S_1 \to S$. We denote by Σ^{PM} the set of pure memoryless strategies for player 1. The pure memoryless player-2 strategies Π^{PM} are defined analogously.

Quantitative objectives. A *quantitative* objective is specified as a measurable function $f : \Omega \to \mathrm{I\!R}$. We consider *zero-sum* games, i.e., games that are strictly competitive. In zero-sum games the objectives of the two players are functions f and $-f$, respectively. We consider quantitative objectives specified as limsup and liminf objectives. These objectives are complete for the second levels of the Borel hierarchy: limsup objectives are Π_2-complete, and liminf objectives are Σ_2-complete. The definition of limsup and liminf objectives is as follows.

- *Limsup objectives.* Let $r : S \to \mathrm{I\!R}$ be a real-valued reward function that assigns to every state s the reward $r(s)$. The limsup objective assigns to every play the maximum reward that appears infinitely often in the play. Formally, for a play $\omega = \langle s_1, s_2, s_3, \ldots \rangle$ we have
$$\text{limsup}(r)(\omega) = \limsup \langle r(s_i) \rangle_{i \geq 0} = \lim_{n \to \infty} \max\{ \, r(s_i) \mid i \geq n \, \}.$$
- *Liminf objectives.* Let $r : S \to \mathrm{I\!R}$ be a real-valued reward function that assigns to every state s the reward $r(s)$. The liminf objective assigns to every play the maximum reward v such that the rewards that appear eventually always in the play are at least v. Formally, for a play $\omega = \langle s_1, s_2, s_3, \ldots \rangle$ we have
$$\text{liminf}(r)(\omega) = \liminf \langle r(s_i) \rangle_{i \geq 0} = \lim_{n \to \infty} \min\{ \, r(s_i) \mid i \geq n \, \}.$$

  The limsup and liminf objectives are complementary in the sense that for all plays ω we have $\text{limsup}(r)(\omega) = -\text{liminf}(-r)(\omega)$. If the reward function r is boolean (that is rewards are only 0 and 1), then (a) the limsup objective correspond to the classical Büchi objective with the set of states with reward 1 as the set of Büchi states; and (b) the liminf objective correspond to the classical coBüchi objective with the set of states with reward 1 as the set of coBüchi states.

Values and optimal strategies. Given a game graph G and a measurable function $f : \Omega \to \mathrm{I\!R}$ we define the *value* functions $\langle\!\langle 1 \rangle\!\rangle_{val}^{G}$ and $\langle\!\langle 2 \rangle\!\rangle_{val}^{G}$ for the players 1 and 2, respectively, as the following functions from the state space S to the set $\mathrm{I\!R}$ of reals: for all states $s \in S$, let

$$\langle\!\langle 1 \rangle\!\rangle_{val}^{G}(f)(s) = \sup_{\sigma \in \Sigma} \inf_{\pi \in \Pi} \mathbb{E}_s^{\sigma,\pi}[f];$$
$$\langle\!\langle 2 \rangle\!\rangle_{val}^{G}(-f)(s) = \sup_{\pi \in \Pi} \inf_{\sigma \in \Sigma} \mathbb{E}_s^{\sigma,\pi}[-f].$$

In other words, the value $\langle\!\langle 1 \rangle\!\rangle_{val}^{G}(f)(s)$ gives the maximal expectation with which player 1 can achieve her objective f from state s, and analogously for player 2. The strategies that achieve the values are called optimal: a strategy σ for player 1 is *optimal* from the state s for f if $\langle\!\langle 1 \rangle\!\rangle_{val}^{G}(f)(s) = \inf_{\pi \in \Pi} \mathbb{E}_s^{\sigma,\pi}[f]$. The optimal strategies for player 2 are defined analogously. We now state the classical determinacy results for $2^1\!/\!_2$-player games with limsup and liminf objectives.

Theorem 1 (Quantitative determinacy). *Let $G = ((S, E), (S_1, S_2, S_P), \delta)$ be a $2^1\!/\!_2$-player game graph. For all reward functions $r : S \to \mathrm{I\!R}$ and all states $s \in S$, we have*

$$\langle\!\langle 1 \rangle\!\rangle_{val}^{G}(\text{limsup}(r))(s) + \langle\!\langle 2 \rangle\!\rangle_{val}^{G}(\text{liminf}(-r))(s) = 0;$$
$$\langle\!\langle 1 \rangle\!\rangle_{val}^{G}(\text{liminf}(r))(s) + \langle\!\langle 2 \rangle\!\rangle_{val}^{G}(\text{limsup}(-r))(s) = 0.$$

The above results can be derived from the results in [16] or from the result of Martin [17].

3 Computational and Strategy Complexity

In this section we survey the computational complexity and the structural properties of optimal strategies in various subclasses of stochastic games. We organize our results for various classes of game graphs. The classical algorithmic solutions for stochastic games can be classified as (a) graph-theoretic algorithms or (b) value-iteration algorithms. We briefly discuss the general properties of the value-iteration algorithm and provide specific details of the algorithms for different classes of the game graphs later (in specific subsections).

Value-iteration algorithms and improvement functions. The values of stochastic games and their subclasses with limsup and liminf objectives can be characterized as fixpoint solution of certain nested fixpoint formulas. The characterization provides *symbolic* value-iteration algorithms to compute values by iterating certain *binary improvement functions* parametrized by a predecessor operator Pre that will be instantiated according to the different classes of game graphs. A *valuation* is a function $v\colon S \to \mathbb{R} \cup \{ -\infty, \infty \}$ that maps every state to a real number[1]. We write V for the set of valuations. A binary improvement function $\mathsf{Imp2}$ operates on pairs of valuations and needs to satisfy the following requirements.

Monotone For all valuation pairs $(v_1, u_1), (v_2, u_2)$, if $(v_1, u_1) \leq (v_2, u_2)$, then $\mathsf{Imp2}(v_1, u_1) \leq \mathsf{Imp2}(v_2, u_2)$ (the inequality $\leq$ is pointwise for valuations).

Continuous For every chain $C = \langle (v_0, u_0), (v_1, u_1), (v_2, u_2), \ldots \rangle$ of valuations, the sequence $\mathsf{Imp2}(C) = \langle \mathsf{Imp2}(v_0, u_0), \mathsf{Imp2}(v_1, u_1), \mathsf{Imp2}(v_2, u_2), \ldots \rangle$ is a chain of valuations by monotonicity of $\mathsf{Imp2}$. We require that $\mathsf{Imp2}(\lim C) = \lim \mathsf{Imp2}(C)$.

Directed Either $v \geq \mathsf{Imp2}(v, u) \geq u$ for all valuations v, u with $v \geq u$; or $v \leq \mathsf{Imp2}(v, u) \leq u$ for all real valuations v, u with $v \leq u$.

If the above requirements are satisfied, then we can invoke Kleene's fixpoint theorem for existence of fixpoints with the improvement functions. The binary improvement functions we consider also satisfy the following *locality* property: for all states $s \in S$ and all valuation pairs $(v_1, u_1), (v_2, u_2)$, if $v_1(s') = v_2(s')$ and $u_1(s') = u_2(s')$ for all successors $s' \in E(s)$, then $\mathsf{Imp2}(v_1, u_1)(s) = \mathsf{Imp2}(v_2, u_2)(s)$.

The description of improvement functions. Consider a reward function r, and the corresponding objectives $\mathrm{limsup}(r)$ and $\mathrm{liminf}(r)$. Given a function $\mathsf{Pre}\colon V \to V$, we define the two parametric functions $\mathsf{limsupImp}[\mathsf{Pre}]\colon V \times V \to V$ and $\mathsf{liminfImp}[\mathsf{Pre}]\colon V \times V \to V$ by

$$\begin{aligned}\mathsf{limsupImp}[\mathsf{Pre}](v,u) &= \min\{ \max\{ r, u, \mathsf{Pre}(u) \}, v, \max\{ u, \mathsf{Pre}(v) \} \};\\ \mathsf{liminfImp}[\mathsf{Pre}](v,u) &= \max\{ \min\{ r, u, \mathsf{Pre}(u) \}, v, \min\{ u, \mathsf{Pre}(v) \} \};\end{aligned}$$

for all valuations $v, u \in V$ (the functions max and min are lifted from real values to valuations in a pointwise fashion). Observe that if $v \geq u$, then $v \geq$

[1] We add $-\infty$ and ∞ to the set of reals in the range of valuations so that the set V of valuations form a complete lattice.

$\mathsf{limsupImp}[\mathsf{Pre}](v,u) \geq u$; and if $v \leq u$, then $v \leq \mathsf{liminfImp}[\mathsf{Pre}](v,u) \leq u$. Thus both $\mathsf{limsupImp}[\mathsf{Pre}]$ and $\mathsf{liminfImp}[\mathsf{Pre}]$ are directed. For different graph models, we will instantiate the parameter Pre differently. We remark that in all the cases that we will consider in this paper, we can simplify the above definitions of the binary improvement functions as follows:

$$\begin{aligned}\mathsf{limsupImp}[\mathsf{Pre}](v,u) &= \min\{\, \max\{\, r, u, \mathsf{Pre}(u) \,\}, v, \mathsf{Pre}(v) \,\};\\ \mathsf{liminfImp}[\mathsf{Pre}](v,u) &= \max\{\, \min\{\, r, u, \mathsf{Pre}(u) \,\}, v, \mathsf{Pre}(v) \,\};\end{aligned}$$

for all valuations $v, u \in V$. To see why the simplification is sound, let $u^{j+1} = \mathsf{limsupImp}[\mathsf{Pre}](v, u^j)$ (according to the original, unsimplified definition) for all $j \geq 0$. For all valuations $v \geq u^0$, if $\mathsf{Pre}(v) \geq u^0$, then for all $j \geq 0$, both $v \geq u^j$ and $\mathsf{Pre}(v) \geq u^j$, and therefore $u^{j+1} = \min\{\, \max\{\, p, u^j, \mathsf{Pre}(u^j) \,\}, v, \mathsf{Pre}(v) \,\}$. If $u^0(s) = \min_{t \in S} r(t)$ for all $s \in S$, then for all instantiations of Pre (that we will use) for all valuations $v \geq u^0$, we will have $\mathsf{Pre}(v) \geq u^0$, and thus the above simplification is sound. The case $\mathsf{liminfImp}[\mathsf{Pre}]$ and $u^0(s) = \max_{t \in S} r(t)$ for all $s \in S$ is symmetric. In some special cases of boolean reward functions r, the valuations can also be restricted to be functions from states to boolean (such as 2-player game graphs with boolean reward functions). In such cases, we can invoke Tarski-Knaster fixpoint theorem that requires only the monotonicity property. Then the improvement function can be further simplified as follows:

$$\begin{aligned}\mathsf{limsupImp}[\mathsf{Pre}](v,u) &= \min\{\, \max\{\, r, \mathsf{Pre}(u) \,\}, \mathsf{Pre}(v) \,\};\\ \mathsf{liminfImp}[\mathsf{Pre}](v,u) &= \max\{\, \min\{\, r, \mathsf{Pre}(u) \,\}, \mathsf{Pre}(v) \,\};\end{aligned}$$

The above description of the improvement functions does not satisfy the directed property. Also see [4] for a more detailed discussion about the properties of the fixpoint and the requirements of improvement functions.

Fixpoint characterization. Given the two parametric improvement functions, the value function of player 1 for limsup objective can be characterized as a nested fixpoint solution (nesting of a greatest fixpoint and a least fixpoint). In μ-calculus notation, let

$$v^{ls} = (\nu x)(\mu y)\ \mathsf{limsupImp}[\mathsf{Pre}](x,y). \tag{1}$$

Then for suitable instantiation Pre in $\mathsf{limsupImp}[\mathsf{Pre}]$ the valuation v^{ls} gives the value function for a stochastic game with limsup objective. Symmetrically, the value function for liminf objective can also be characterized as a nested fixpoint solution (nesting of a least fixpoint and a greatest fixpoint). In μ-calculus notation, let

$$v^{li} = (\mu x)(\nu y)\ \mathsf{liminfImp}[\mathsf{Pre}](x,y). \tag{2}$$

Then for suitable instantiation Pre in $\mathsf{liminfImp}[\mathsf{Pre}]$ the valuation v^{li} gives the value function for a stochastic game with liminf objective. In all cases that we consider, for the least fixpoint iterations are initialized with the valuation $\min r$ (i.e, the valuation that assigns the value $\min r$ to all states) , and for the greatest fixpoint iterations are initialized with the valuation $\max r$. We will illustrate the value-iteration algorithm and the fixpoint characterization on an example in the case of 2-player game graphs. In the following subsection we present the instantiation of Pre for different classes of game graphs.

3.1 1-Player Game Graphs

In this subsection we present the results for 1-player game graphs with limsup and liminf objectives. For simplicity we consider 1-player game graphs with $S_P = \emptyset$ and $S_2 = \emptyset$ (the results for the case when $S_P = \emptyset$ and $S_1 = \emptyset$ are similar).

Strategy complexity. Pure memoryless optimal strategies exist for 1-player game graphs with limsup and liminf objectives. The result can be obtained as a special case of the result known for 2-player game graphs (see Section 3.2) or $1^1\!/_2$-player game graphs (see Section 3.3).

Value-iteration algorithm. We present the value iteration solution for 1-player game graphs. We define the *graph predecessor operator* $\mathsf{maxPre}\colon V \to V$ as the function on valuations defined by

$$\mathsf{maxPre}(v)(s) \;=\; \max\{\, v(s') \mid s' \in E(s) \,\}$$

for all valuations $v \in V$ and all states $s \in S$; that is, the value of $\mathsf{maxPre}(v)$ at a state s is the maximal value of v at the states that are successors of s. If the parameter Pre is instantiated as maxPre, then the nested fixpoint solution of (1) gives the value function for 1-player game graphs with limsup objectives, and the solution of (2) gives the value function for liminf objectives. Each inner improvement fixpoint converges within at most $\boldsymbol{n}$ steps, and the outer improvement fixpoints converges within at most $\boldsymbol{n}$ computations of inner improvement fixpoints. Every improvement step (i.e., each application of the function $\mathsf{limsupImp}[\mathsf{Pre}]$ or $\mathsf{liminfImp}[\mathsf{Pre}]$) can be computed in $O(\boldsymbol{m})$ time. Hence the value-iteration algorithm has the time complexity $O(\boldsymbol{mn}^2)$.

Graph-theoretic algorithm. The value function for 1-player game graphs with the limsup and liminf can be obtained in $O(\boldsymbol{m})$ time. The algorithm for limsup objective is as follows. First compute the set of all maximal strongly connected components (this can be done in $O(\boldsymbol{m})$ time). For a bottom maximal strongly connected component C, the value of every state in C is $\max_{s\in C} r(s)$. Then proceed in a bottom up fashion: consider a maximal strongly connected component C' such that for every state $t \in \left(\bigcup_{s\in C'} E(s)\right) \setminus C'$ the value of state t is computed, and let this value be $v(t)$. The value of every state $s \in C'$ is as follows:

1. If either (a) $|C'| \geq 2$, or (b) $|C'| = 1$ and the only state of C' has a self-loop; then for every state $s \in C'$ the value $v(s)$ is given by

$$\max\{\, \max\{\, r(s) \mid s \in C' \,\}, \max\{\, v(t) \mid \exists s \in C' \cdot t \in E(s) \,\} \,\}.$$

2. If $|C'| = 1$ and the only state of C' does not have self-loop, then the value $v(s)$ of the only state s of C' is given by
$$\max\{\, v(t) \mid \exists s \in C' \cdot t \in E(s) \,\}.$$

Thus value of every state can be computed in $O(\boldsymbol{m})$ time. The algorithm for liminf objectives is similar. We know of no implementation of the nested value improvement scheme that matches this complexity. We summarize the results in the following theorem.

Theorem 2 (Complexity of 1-player game graphs). *For all 1-player game graphs with limsup and liminf objectives, the following assertions hold.*

1. *Pure memoryless optimal strategies exist.*
2. *The value function can be computed in $O(\boldsymbol{n}^2\boldsymbol{m})$ time by the value-iteration algorithm.*
3. *The value function can be computed in $O(\boldsymbol{m})$ time by the graph-theoretic algorithm.*

Remark 1. The graph-theoretic algorithm we present runs in $O(\boldsymbol{m})$ time, as compared to the previously known algorithm of [2] that runs in $O(\boldsymbol{m} + \boldsymbol{n} \cdot \log \boldsymbol{n})$ time. The algorithm of [2] first sorted states with respect to the rewards and then applied algorithms for Büchi (or coBüchi) objectives, whereas our algorithm does not need the sorting step of the previous algorithm.

3.2 2-Player Game Graphs

We now present the results for 2-player game graphs with limsup and liminf objectives.

Strategy complexity. Pure memoryless optimal strategies exist for 2-player game graphs with limsup and liminf objectives. The result has several different proofs. In [13] Gimbert and Zielonka present sufficient conditions on measurable functions (that specify quantitative objectives) that ensures existence of pure memoryless optimal strategies in 2-player game graphs. It was also shown in [13] that limsup and liminf objectives satisfy the required conditions, and hence existence of pure memoryless optimal strategies in 2-player game graphs with limsup and liminf objectives follows.

Value-iteration algorithm. The value-iteration solution for 2-player game graphs uses the *game graph predecessor operator* $\mathsf{maxminPre}\colon V \to V$ defined by
$$\mathsf{maxminPre}(v)(s) \;=\; \begin{cases} \max\{\, v(s') \mid s' \in E(s) \,\} & \text{if } s \in S_1; \\ \min\{\, v(s') \mid s' \in E(s) \,\} & \text{if } s \in S_2; \end{cases}$$
for all valuations $v \in V$ and all states $s \in S$. In other words, the value of $\mathsf{maxminPre}(v)$ at a player-1 state s is the maximal value of v at the successors of s, and at a player-2 state s it is the minimal value of v at the successors of s.

If the parameter Pre is instantiated as $\mathsf{maxminPre}$, then the nested fixpoint solution of (1) gives the value function for 2-player game graphs with limsup objectives, and the solution of (2) gives the value function for liminf objectives. Each inner improvement fixpoint converges within at most $\boldsymbol{n}$ steps, and the outer improvement fixpoints converges within at most $\boldsymbol{n}$ computations of inner improvement fixpoints. Every improvement step (i.e., each application of the function $\mathsf{limsupImp}[\mathsf{maxminPre}]$ or $\mathsf{liminfImp}[\mathsf{maxminPre}]$) can be computed in $O(\boldsymbol{m})$ time. Hence the value-iteration algorithm has the time complexity $O(\boldsymbol{mn}^2)$.

Example 1 (2-player game with limsup objective). Consider the deterministic game shown in Fig. 1, where the reward function r is indicated by state labels. We consider the objective limsup(r) for player 1 (the $\Box$ player). We specify valuations as value vectors; the outer initial valuation is $v^0 = \langle 15, 15, 15, 15, 15\rangle$, and the inner initial valuation is $u^0 = \langle 5, 5, 5, 5, 5\rangle$. We compute the first inner improvement fixpoint: $u_0^0 = \langle 5, 5, 5, 5, 5\rangle$, and since

$$u_0^{j+1} \;=\; \min\{\, \max\{\, r, u_0^j, \mathsf{maxminPre}(u_0^j)\,\}, v^0, \mathsf{maxminPre}(v^0)\,\}$$

for all $j \geq 0$, where $v^0 = \mathsf{maxminPre}(v^0) = \langle 15, 15, 15, 15, 15\rangle$, we obtain $u_0^1 = \langle 5, 5, 15, 10, 5\rangle$. Note that u_0^1 coincides with the reward function r. Next we obtain $u_0^2 = \max\{\, r, u_0^1, \mathsf{maxminPre}(u_0^1)\,\} = \langle 10, 5, 15, 10, 10\rangle$. Finally $u_0^3 = u_0^4 = \langle 10, 10, 15, 10, 10\rangle$, which is the first inner improvement fixpoint v^1. Intuitively, $v^i(s)$ is the largest reward that player 1 can ensure to visit at least i times from s. The second inner improvement chain starts with $u_1^0 = \langle 5, 5, 5, 5, 5\rangle$ using

$$u_1^{j+1} \;=\; \min\{\, \max\{\, r, u_1^j, \mathsf{maxminPre}(u_1^j)\,\}, v^1, \mathsf{maxminPre}(v^1)\,\},$$

where $v^1 = \langle 10, 10, 15, 10, 10\rangle$ and $\mathsf{maxminPre}(v^1) = \langle 10, 10, 10, 10, 10\rangle$. Since $\max\{\, r, u_1^0, \mathsf{maxminPre}(u_1^0)\,\} = \langle 5, 5, 15, 10, 5\rangle$, we obtain $u_1^2 = \langle 5, 5, 10, 10, 5\rangle$ and $u_1^3 = \langle 10, 5, 10, 10, 10\rangle$ Then $u_1^3 = u_1^4 = \langle 10, 10, 10, 10, 10\rangle$, which is the second inner improvement fixpoint v^2. This is also the desired outer improvement fixpoint; that is, $v^{ls} = v^2 = v^3 = \langle 10, 10, 10, 10, 10\rangle$. The player-1 strategy that chooses at state s_0 the successor s_3 ensures that against all strategies of player 2, the reward 10 will be visited infinitely often. Dually, the player-2 strategy that chooses at s_1 the successor s_0 ensures that against all strategies of player 1, the reward 15 will be visited at most once. Hence $\langle 10, 10, 10, 10, 10\rangle$ is the 2-player game valuation of the player-1 objective limsup(r): from any start state, player 1 can ensure that reward 10 will be visited infinitely often, but she cannot ensure reward 15. ■

Graph-theoretic algorithm. The value function for 2-player game graphs with the limsup and liminf objectives can be computed in $O(\frac{\boldsymbol{mn}\log(\Delta)\log(\boldsymbol{k})}{\log(\boldsymbol{n})})$ time, where $\boldsymbol{k}$ is the number of different rewards of the reward function in the game graph. The algorithm for limsup objectives is as follows: we first sort the rewards in ascending order, and let the reward values in ascending order be $r_1 < r_2 < \cdots < r_{\boldsymbol{k}}$. To check if the value at a state s is at at least r_i, for $1 \leq i \leq \boldsymbol{k}$, we consider all states with rewards at least r_i as Büchi states, and

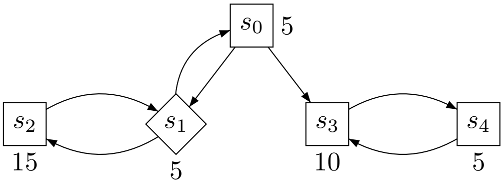


Fig. 1. Deterministic game with limsup objective

then check if player 1 can satisfy the Büchi objective from s. A game with a Büchi objective can be solved in $O(\frac{\boldsymbol{mn}\log(\Delta)}{\log(\boldsymbol{n})})$ time by graph-theoretic algorithms [6]. By a binary search over the sorted set of rewards we can compute the value in $O(\frac{\boldsymbol{mn}\log(\Delta)\log(\boldsymbol{k})}{\log(\boldsymbol{n})})$ time. The algorithm for liminf objectives is similar, and it uses solution of games with coBüchi objectives instead of Büchi objectives. We know of no implementation of the nested value improvement scheme that matches this complexity. We summarize the results in the following theorem.

Theorem 3 (Complexity of 2-player game graphs). *For all 2-player game graphs with limsup and liminf objectives, the following assertions hold.*

1. *Pure memoryless optimal strategies exist.*
2. *The value function can be computed in $O(\boldsymbol{n}^2\boldsymbol{m})$ time by the value-iteration algorithm.*
3. *The value function can be computed in $O(\frac{\boldsymbol{mn}\log(\Delta)\log(\boldsymbol{k})}{\log(\boldsymbol{n})})$ time by the graph-theoretic algorithm.*

Remark 2. Observe that for the worst case complexity for graph-theoretic algorithmic solution we have $\Delta = O(\boldsymbol{n})$ and $\boldsymbol{k} = O(n)$, and then the graph-theoretic algorithm runs in time $O(\boldsymbol{mn}\log(\boldsymbol{n}))$. The worst-case complexity of the previously known algorithm (of [2]) is $O(\boldsymbol{mn}^2)$.

3.3 1½-Player Game Graphs

We now present the results for 1½-player game graphs with limsup and liminf objectives.

Strategy complexity. Pure memoryless optimal strategies exist for 1½-player game graphs with limsup and liminf objectives. This fact can be proved by straightforward extension of the results and proof techniques for MDPs with Büchi and coBüchi objectives. The existence of pure memoryless optimal strategies in MDPs with Büchi and coBüchi objectives has been shown in [8,10].

Value-iteration algorithm. To present the value iteration solution for 1½-player game graphs, we need the *probabilistic graph predecessor operator* $\mathsf{maxPre}^P : V \to V$ defined by

$$\mathsf{maxPre}^P(v)(s) \;=\; \begin{cases} \max\{\, v(s') \mid s' \in E(s) \,\} & \text{if} \;\; s \in S_1; \\ \sum_{s' \in E(s)} v(s') \cdot \delta(s)(s') & \text{if} \;\; s \in S_P; \end{cases}$$

for all valuations $v \in V$ and all states $s \in S$. In other words, the value of $\mathsf{maxPre}^P(v)$ at a player-1 state s is the maximal value of v at the successors of s, and the value of $\mathsf{maxPre}^P(v)$ at a probabilistic state s is the average value of v at the successors of s. If the parameter Pre is instantiated as maxPre^P, then the nested fixpoint solution of (1) gives the value function for $1^1\!/_2$-player game graphs with limsup objectives, and the solution of (2) gives the value function for liminf objectives. Unlike the case of 1-player and 2-player game graphs, the inner and outer iterations do not necessarily converge in finitely many iterations, but converge only in the limit. We now present the result on the *boundedness properties* of values for rational rewards and transition probabilities that allows to compute the exact values by value-iteration algorithms.

Precision of values. We assume that all transition probabilities and rewards are given as rational numbers, and for simplicity (but without loss of generality) we assume that all rewards are positive. From the existence of pure memoryless optimal strategies, and the results of [9,20] it follows that all values in $1^1\!/_2$-player game graphs with limsup and liminf objectives are again rationals and that the denominators can be bounded. Let $\delta_u = \max\{\, d \mid \delta(s)(s') = \frac{n}{d} \text{ for } s \in S_P \text{ and } s' \in E(s) \,\}$ be the largest denominator of all transition probabilities. Let $r_u = \mathrm{lcm}\{\, d \mid r(s) = \frac{n}{d} \text{ for } s \in S \,\}$ be the least common multiple of all reward denominators. Let $r_{\max} = \max\{\, n \mid r(s) = \frac{n}{d} \text{ for } s \in S \,\}$ be the largest numerator of all rewards. Then, for all states $s \in S$, both $\langle\!\langle 1 \rangle\!\rangle^G_{val}(\mathrm{limsup}(r))(s)$ and $\langle\!\langle 1 \rangle\!\rangle^G_{val}(\mathrm{liminf}(r))(s)$ have the form $\frac{n}{d}$ for positive integers n and d with $n, d \le \boldsymbol{\gamma}$, where

$$\boldsymbol{\gamma} \;=\; \delta_u^{4\boldsymbol{m}} \cdot r_u \cdot r_{\max}.$$

This *boundedness* property of values for limsup and liminf objectives in $1^1\!/_2$-player game graphs is the key for proving computability of the two improvement fixpoints. The inner fixpoint can be computed as follows: the improvement function can be iterated for $2 \cdot \boldsymbol{\gamma}^2$ iterations, and the valuation obtained is rounded to the nearest multiple of $\frac{1}{\gamma}$ to obtain the inner fixpoint (the argument is similar to the value-iteration algorithms of [9,20]). Similarly, the valuation of the outer fixpoint can be obtained by rounding after $2 \cdot \boldsymbol{\gamma}^2$ iterations of the outer fixpoint computation. Hence the value-iteration algorithm has the time complexity $O(\boldsymbol{\gamma}^4)$.

Graph-theoretic algorithm and linear program. The value function for $1^1\!/_2$-player game graphs with the limsup and liminf objective can be computed in polynomial time. Let $\boldsymbol{k}$ be the number of different reward values. The key steps of the algorithm for limsup objective is as follows: (a) first the rewards are sorted in ascending order; (b) then *qualitative analysis* (computing the set of states with value 1) of sub-graphs of the given $1^1\!/_2$-player game graph with Büchi objectives is performed, and there are $\boldsymbol{k}$ calls to the qualitative analysis algorithm (see [5] for details) for Büchi objectives which can be performed in polynomial time using algorithms of [7]; (c) after the above analysis the value function can be obtained by solving a linear program. The algorithm for liminf objective is similar and it uses qualitative analysis for coBüchi objectives (see [5] for details). We know of no implementation of the nested value improvement scheme that runs in polynomial time. We summarize the results in the following theorem.

Theorem 4 (Complexity of $1\frac{1}{2}$-player game graphs). *For all $1\frac{1}{2}$-player game graphs with limsup and liminf objectives, the following assertions hold.*

1. *Pure memoryless optimal strategies exist.*
2. *The value function can be computed in $O(\gamma^4)$ time by the value-iteration algorithm.*
3. *The value function can be computed in polynomial time by the graph-theoretic algorithm and linear programming.*

3.4 $2\frac{1}{2}$-Player Game Graphs

Finally, in this section we present the results for $2\frac{1}{2}$-player game graphs with limsup and liminf objectives.

Strategy complexity. Pure memoryless optimal strategies exist for $2\frac{1}{2}$-player game graphs with limsup and liminf objectives. The results (Theorem 3.19 of [12]) showed that if for a quantitative objective f and its complement $-f$ pure memoryless optimal strategies exist in $1\frac{1}{2}$-player game graphs, then pure memoryless optimal strategies also exist in $2\frac{1}{2}$-player games. Since pure memoryless optimal strategies exist for both limsup and liminf objectives in $1\frac{1}{2}$-player game graphs (Theorem 4), the existence of pure memoryless optimal strategies follows for $2\frac{1}{2}$-player games with limsup and liminf objectives.

Value-iteration algorithm. To present the value-iteration solution for $2\frac{1}{2}$-player game graphs, we need the *probabilistic game graph predecessor operator* $\mathsf{maxminPre}^P \colon V \to V$ defined by

$$\mathsf{maxminPre}^P(v)(s) \;=\; \begin{cases} \max\{\, v(s') \mid s' \in E(s) \,\} & \text{if } \; s \in S_1; \\ \min\{\, v(s') \mid s' \in E(s) \,\} & \text{if } \; s \in S_2; \\ \sum_{s' \in E(s)} v(s') \cdot \delta(s)(s') & \text{if } \; s \in S_P; \end{cases}$$

for all valuations $v \in V$ and all states $s \in S$. The predecessor operator $\mathsf{maxminPre}^P$ is a generalization of game graph predecessor operator $\mathsf{maxminPre}$ and the probabilistic graph predecessor operator maxPre^P. If the parameter Pre is instantiated as $\mathsf{maxminPre}^P$, then the nested fixpoint solution of (1) gives the value functions for $2\frac{1}{2}$-player game graphs with limsup objectives, and the solution of (2) gives the value function for liminf objectives. The boundedness properties of the values of $1\frac{1}{2}$-player game graphs also holds for $2\frac{1}{2}$-player game graphs, and bounds on the number of iterations to compute the fixpoints for $1\frac{1}{2}$-player game graphs also generalize to $2\frac{1}{2}$-player game graphs. Hence if all the rewards and transition probabilities are rational, then the value function for $2\frac{1}{2}$-player game graphs with limsup and liminf objectives can be computed in $O(\gamma^4)$ time using value-iteration algorithm.

Optimal algorithm. The problem to decide, given a state s and a rational number q, whether the value function at s is at least q for $2\,^1\!/_2$-player game graphs with limsup and liminf objectives lies in NP ∩ coNP [5]. The result follows from existence of pure memoryless strategies, and the polynomial time algorithms to compute values in $1\,^1\!/_2$-player game graphs with limsup and liminf objectives. No polynomial-time algorithms are known for computing values for limsup and liminf objectives in $2\,^1\!/_2$-player game graphs. In particular, the qualitative analysis and the linear-programming approach for $1\,^1\!/_2$-player game graphs do not generalize to $2\,^1\!/_2$-player game graphs. We summarize the results in the following theorem.

Theorem 5 (Complexity of $2\,^1\!/_2$-player game graphs). *For all $2\,^1\!/_2$-player game graphs with limsup and liminf objectives, the following assertions hold.*

1. *Pure memoryless optimal strategies exist.*
2. *The value function can be computed in $O(\boldsymbol{\gamma}^4)$ time by value-iteration algorithm.*
3. *Given a state s and a rational number q, whether the value function at s is at least q can be decided in NP ∩ coNP.*

Table 1. Nested value improvement for limsup and liminf objectives. Recall that $\boldsymbol{\gamma}$ is such that $16^{\boldsymbol{n}} \in O(\boldsymbol{\gamma})$.

$\boldsymbol{n}$ states $\boldsymbol{m}$ edges	Objective limsup(r)	Objective liminf(r)
1-player graphs	Predecessor operator $\mathsf{limsupImp}[\mathsf{maxPre}](v,u)$ Value-iteration complexity $O(\boldsymbol{n}^2\boldsymbol{m})$ Best known complexity $O(\boldsymbol{m})$	Predecessor operator $\mathsf{liminfImp}[\mathsf{maxPre}](v,u)$ Value-iteration complexity $O(\boldsymbol{n}^2\boldsymbol{m})$ Best known complexity $O(\boldsymbol{m})$
2-player games	Predecessor operator $\mathsf{limsupImp}[\mathsf{maxminPre}](v,u)$ Value-iteration complexity $O(\boldsymbol{n}^2\boldsymbol{m})$ Best known complexity $O(\boldsymbol{nm}\log\boldsymbol{n})$	Predecessor operator $\mathsf{liminfImp}[\mathsf{maxminPre}](v,u)$ Value-iteration complexity $O(\boldsymbol{n}^2\boldsymbol{m})$ Best known complexity $O(\boldsymbol{nm}\log\boldsymbol{n})$
$1\,^1\!/_2$-player graphs	Predecessor operator $\mathsf{limsupImp}[\mathsf{maxPre}^P](v,u)$ Value-iteration complexity $O(\boldsymbol{\gamma}^4)$ Best known complexity is polynomial time	Predecessor operator $\mathsf{liminfImp}[\mathsf{maxPre}^P](v,u)$ Value-iteration complexity $O(\boldsymbol{\gamma}^4)$ Best known complexity is polynomial time
$2\,^1\!/_2$-player games	Predecessor operator $\mathsf{limsupImp}[\mathsf{maxminPre}^P](v,u)$ Value-iteration complexity $O(\boldsymbol{\gamma}^4)$ Best known complexity is NP ∩ coNP	Predecessor operator $\mathsf{liminfImp}[\mathsf{maxminPre}^P](v,u)$ Value-iteration complexity $O(\boldsymbol{\gamma}^4)$ Best known complexity NP ∩ coNP

4 Conclusion

In this survey, we considered stochastic games and their subclasses with limsup and liminf objectives. In Table 1, we summarize the results for the different classes of game graphs. We presented a comprehensive study of the known results in terms of the complexity of strategies, and the two classical algorithmic solutions, namely, value-iteration algorithms and graph-theoretic algorithms. For 1-player and 2-player games, we also improved the previously known graph-theoretic algorithms and their complexity.

Finally, note that the 1-player game graphs with limsup and liminf objective can be viewed as weighted automata with limsup and liminf functions, and computing the value of such games can then be viewed as computing the greatest value of a word in such weighted automata, which amounts to solving the so-called quantitative emptiness problem [3].

References

1. Bjorklund, H., Sandberg, S., Vorobyov, S.: A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. In: Fiala, J., Koubek, V., Kratochvíl, J. (eds.) MFCS 2004. LNCS, vol. 3153, pp. 673–685. Springer, Heidelberg (2004)
2. Chakrabarti, A., de Alfaro, L., Henzinger, T.A., Stoelinga, M.: Resource interfaces. In: Alur, R., Lee, I. (eds.) EMSOFT 2003. LNCS, vol. 2855, pp. 117–133. Springer, Heidelberg (2003)
3. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 385–400. Springer, Heidelberg (2008)
4. Chatterjee, K., Henzinger, T.A.: Value iteration. In: Grumberg, O., Veith, H. (eds.) 25 Years of Model Checking. LNCS, vol. 5000, pp. 107–138. Springer, Heidelberg (2008)
5. Chatterjee, K., Henzinger, T.A.: Probabilistic systems with limsup and liminf objectives. In: ILC (2009)
6. Chatterjee, K., Henzinger, T.A., Piterman, N.: Algorithms for Büchi games. In: GDV (2006)
7. Chatterjee, K., Jurdziński, M., Henzinger, T.A.: Simple stochastic parity games. In: Baaz, M., Makowsky, J.A. (eds.) CSL 2003. LNCS, vol. 2803, pp. 100–113. Springer, Heidelberg (2003)
8. Chatterjee, K., Jurdziński, M., Henzinger, T.A.: Quantitative stochastic parity games. In: SODA 2004, pp. 121–130. SIAM, Philadelphia (2004)
9. Condon, A.: On algorithms for simple stochastic games. In: Advances in Computational Complexity Theory. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, pp. 51–73. American Mathematical Society (1993)
10. de Alfaro, L.: Formal Verification of Probabilistic Systems. PhD thesis, Stanford University (1997)
11. Filar, J., Vrieze, K.: Competitive Markov Decision Processes. Springer, Heidelberg (1997)
12. Gimbert, H.: Jeux positionnels. PhD thesis, Université Paris 7 (2006)

13. Gimbert, H., Zielonka, W.: Games where you can play optimally without any memory. In: Abadi, M., de Alfaro, L. (eds.) CONCUR 2005. LNCS, vol. 3653, pp. 428–442. Springer, Heidelberg (2005)
14. Karp, R.M.: A characterization of the minimum cycle mean in a digraph. Discrete Mathematics 23, 309–311 (1978)
15. Liggett, T.A., Lippman, S.A.: Stochastic games with perfect information and time average payoff. Siam Review 11, 604–607 (1969)
16. Maitra, A., Sudderth, W. (eds.): Discrete Gambling and Stochastic Games. Springer, Heidelberg (1996)
17. Martin, D.A.: The determinacy of Blackwell games. The Journal of Symbolic Logic 63(4), 1565–1581 (1998)
18. McNaughton, R.: Infinite games played on finite graphs. Annals of Pure and Applied Logic 65, 149–184 (1993)
19. Thomas, W.: Languages, automata, and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages. Beyond Words, ch. 7, vol. 3, pp. 389–455. Springer, Heidelberg (1997)
20. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoretical Computer Science 158, 343–359 (1996)

Tractable Optimization Problems through Hypergraph-Based Structural Restrictions*

Georg Gottlob[1], Gianluigi Greco[2], and Francesco Scarcello[2]

[1] Oxford University
[2] University of Calabria
georg.gottlob@comlab.ox.ac.uk, ggreco@mat.unical.it,
scarcello@deis.unical.it

Abstract. Several variants of the Constraint Satisfaction Problem have been proposed and investigated in the literature for modelling those scenarios where solutions are associated with some given costs. Within these frameworks computing an optimal solution is an NP-hard problem in general; yet, when restricted over classes of instances whose constraint interactions can be modelled via (nearly-)acyclic graphs, this problem is known to be solvable in polynomial time. In this paper, larger classes of tractable instances are singled out, by discussing solution approaches based on exploiting hypergraph acyclicity and, more generally, structural decomposition methods, such as (hyper)tree decompositions.

1 Introduction

The Constraint Satisfaction Problem (CSP) is a well-known framework [11] for modelling and solving search problems, which received considerably attention in the literature due to its applicability in various areas. Informally, a CSP instance is defined by singling out the variables of interest, and by listing the allowed combinations of values for groups of them, according to the constraints arising in the application at hand. The solutions for this instance are the assignments of domain values to variables that satisfy all such constraints. Many apparently unrelated problems from disparate areas actually turn out to be equivalent to the CSP and can be accommodated within the CSP framework. Examples are puzzles, conjunctive queries over relational databases, graph colorability, and checking whether there is a homomorphism between two finite structures.

Example 1. Figure 1 shows a combinatorial crossword puzzle (taken from [15]). A set of legal words is associated with each horizontal or vertical array of white boxes delimited by black boxes. A solution to the puzzle is an assignment of a letter to each white box such that to each white array is assigned a word from its

* G.Gottlob works at the Computing Laboratory and at the Oxford Man Institute of Quantitative Finance, Oxford University. This work was done in the context of the EPSRC grant EP/G055114/1 “Constraint Satisfaction for Configuration: Logical Fundamentals,Algorithms, and Complexity” and of Gottlob's Royal Society Wolfson Research Merit Award.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 16–30, 2009.

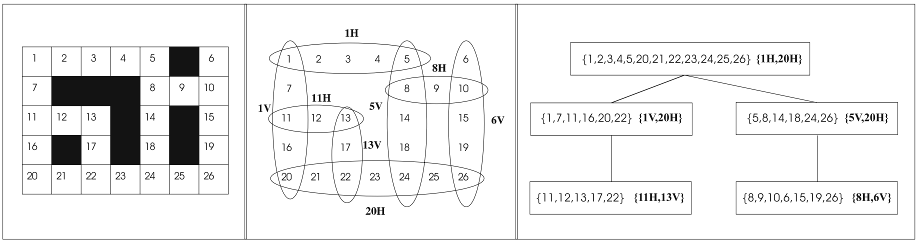

1 2 3 4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19
20 21 22 23 24 25 26
1H
8H
11H
1V
5V
13V
6V
20H
{1,2,3,4,5,20,21,22,23,24,25,26} {1H,20H}
{1,7,11,16,20,22} {1V,20H}
{5,8,14,18,24,26} {5V,20H}
{11,12,13,17,22} {11H,13V}
{8,9,10,6,15,19,26} {8H,6V}

Fig. 1. A crossword puzzle, its associated hypergraph $\mathcal{H}_{cp}$, and a hypertree decomposition of width 2 for $\mathcal{H}_{cp}$

set of legal words. This problem can be recast in a CSP by associating a variable with each white box, and by defining a constraint for each array of white boxes prescribing the legal words that are associated with it. ◁

When assignments are associated with some given cost, however, computing an arbitrary solution might not be enough. For instance, the crossword puzzle in Figure 1 may admit more than one solution, and expert solvers may be asked to single out the most difficult ones, such as those solutions that minimize the total number of vowels occurring in the used words. In these cases, one is usually interested in the corresponding *optimization problem* of computing the solution of minimum cost, whose modeling is accounted for in several variants of the basic CSP framework, such as fuzzy, probabilistic, weighted, lexicographic, valued, and semiring-based CSPs (see [25,4] and the references therein).

Since solving CSPs—and the above extensions—is an NP-hard problem, much research has been spent to identify restricted classes over which solutions can efficiently be computed. In this paper, *structural decomposition methods* are considered [15], which identify tractable classes by exploiting the structure of constraint scopes as it can be formalized either as a hypergraph (whose nodes correspond to the variables and where each group of variables occurring in some constraint induce a hyperedge) or as a suitable binary encoding of such hypergraph. In particular, we focus on the structural methods based on the notions of (generalized) hypertree width [18,19] and treewidth [28]. In both cases, the underlying idea is that solutions to CSP instances that are associated with acyclic (or nearly-acyclic) structures can efficiently be computed via dynamic programming, by incrementally processing the structure according to some of its topological orderings.

As a matter of fact, however, while in the case of classical CSPs deep and useful results have been achieved for both graph and hypergraph representations, in the case of CSP extensions tailored for optimization problems attention was mainly focused on binary encodings and, in particular, on the *primal graph* representation, where nodes correspond to variables and an edge between two variables indicates that they are related by some constraint. Discussing whether (and how) hypergraph-based structural decomposition techniques in the literature can be lifted to such optimization frameworks is the main goal of this paper. In particular, we consider three CSP extensions:

(1) First, we consider optimization problems where every mapping variable-value is associated with a cost, so that the aim is to find an assignment satisfying all the constraints and having the minimum total cost.

(2) Second, we consider the case where costs are associated with the allowed combinations of simultaneous values for the variables occurring in the constraint, rather than to individual values. Again, within this setting, we consider the problem of computing a solution having minimum total cost.

(3) Finally, we consider a scenario where the CSP instance at hand might not admit a solution at all, and where the problem is hence to find the assignment minimizing the total number of violated constraints (and, more generally, whenever a cost is assigned to each constraint, the assignment minimizing the total cost of violated constraints).

For each of the above settings, the complexity of computing the optimal solution is analyzed in this paper, by overviewing some relevant recent research and by providing novel results. In particular:

- ▶ We show that optimization problems of kind **(1)** can be solved in polynomial time on instances having bounded (generalized) hypertree-width hypergraphs. This result is based on an algorithm recently designed and analyzed in the context of *combinatorial auctions* [13].
- ▶ We show that even optimization problems of kind **(2)** are tractable on instances having bounded (generalized) hypertree-width hypergraphs. Indeed, we describe how to transform this kind of instances into equivalent instances of kind **(1)**, by preserving their structural properties.
- ▶ We observe that optimization problems of kind **(3)** remain NP-hard even over instances having an associated acyclic hypergraph. However, there is also good news: they are shown to be tractable on instances having bounded treewidth *incidence graph* encoding. The latter is a binary encoding of the constraint hypergraph with usually better structural features than the primal graph encoding (see, e.g., [15,22]). Again, proof is via a mapping to case **(1)**.

Organization. The rest of the paper is organized as follows. Section 2 discusses preliminaries on CSPs and structural restrictions, and Section 3 provides an overview of the structural decomposition methods based on treewidth and (generalized) hypertree width. Results for optimization problems of kind **(1)** and **(2)** are discussed in Section 4, whereas problems of kind **(3)** are discussed in Section 5. Finally, Section 6 draws our conclusions.

2 CSPs, Acyclic Instances, and Their Desirable Properties

An instance of a constraint satisfaction problem [11] is a triple $\mathcal{I} = \langle Var, U, \mathcal{C} \rangle$, where Var is a finite set of variables, U is a finite domain of values, and $\mathcal{C} = \{C_1, C_2, \ldots, C_q\}$ is a finite set of constraints. Each constraint C_v, for $1 \leq v \leq q$, is a pair (S_v, r_v), where $S_v \subseteq Var$ is a set of variables called the *constraint scope*, and r_v is a set of substitutions (also called *tuples*) from variables in S_v

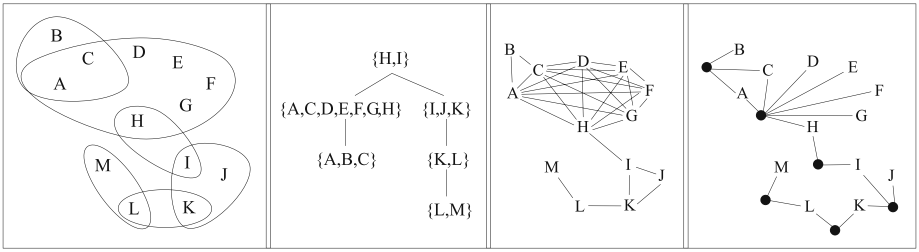


Fig. 2. A hypergraph $\mathcal{H}_1$, a join tree $\mathcal{JT}(\mathcal{H}_1)$, the primal graph $G(\mathcal{H}_1)$, and the incidence graph $inc(\mathcal{H}_1)$

to values in U indicating the allowed combinations of simultaneous values for the variables in S_v. Any substitution from a set of variables $V \subseteq Var$ to U is extensively denoted as the set of pairs of the form X/u, where $u \in U$ is the value to which $X \in V$ is mapped. Then, a solution to $\mathcal{I}$ is a substitution $\theta : Var \mapsto U$ for which q-tuples $t_1 \in r_1, ..., t_q \in r_q$ exist such that $\theta = t_1 \cup ... \cup t_q$.

Example 2. In the crossword puzzle of Figure 1, *Var* coincides with the letters of the alphabet, and a variable X_i (denoted by its index i) is associated with each white box. An example of constraint is $C_{1H} = ((1,2,3,4,5), r_{1H})$, and a possible instance for r_{1H} is $\{\langle h,o,u,s,e\rangle, \langle c,o,i,n,s\rangle, \langle b,l,o,c,k\rangle\}$—in the various constraint names, subscripts H and V stand for “Horizontal” and “Vertical,” respectively, resembling the usual naming of definitions in crossword puzzles. $\triangleleft$

The structure of a CSP instance $\mathcal{I}$ is best represented by its associated hypergraph $\mathcal{H}(\mathcal{I}) = (V, H)$, where $V = Var$ and $H = \{S \mid (S,r) \in \mathcal{C}\}$—in the following, V and H will be denoted by $\mathcal{N}(\mathcal{H})$ and $\mathcal{E}(\mathcal{H})$, respectively. As an example, the hypergraph associated with the crossword puzzle formalized above is illustrated in the central part of Figure 1.

A hypergraph $\mathcal{H}$ is *acyclic* iff it has a join tree [3]. A *join tree* $\mathcal{JT}(\mathcal{H})$ for a hypergraph $\mathcal{H}$ is a tree whose vertices are the hyperedges of $\mathcal{H}$ such that, whenever the same node $X \in V$ occurs in two hyperedges h_1 and h_2 of $\mathcal{H}$, then X occurs in each vertex on the unique path linking h_1 and h_2 in $\mathcal{JT}(\mathcal{H})$. The notion of acyclicity we use here is the most general one known in the literature, coinciding with α-acyclicity according to Fagin [9]. Note that the hypergraph $\mathcal{H}_{cp}$ of Figure 1 is not acyclic. An acyclic hypergraph is discussed below.

Example 3. Consider the hypergraph $\mathcal{H}_1$ shown on the left of Figure 2, which is associated with a CSP instance over the set of variables $\{A, ..., M\}$. In particular, six constraints are defined over the instance whose scopes precisely correspond to the hyperedges in $\mathcal{E}(\mathcal{H}_1)$; for instance, $\{A, B, C\}$ is an example of constraint scope. Note also that $\mathcal{H}_1$ is acyclic. Indeed, a join tree $\mathcal{JT}(\mathcal{H}_1)$ for it is reported in the same figure to the right of $\mathcal{H}_1$. $\triangleleft$

An important property of acyclic instances is that they can efficiently be processed by dynamic programming. Indeed, according to Yannakakis' algorithm [34] (originally conceived in the equivalent context of evaluating acyclic

Boolean conjunctive queries), they can be evaluated by processing any of their join trees bottom-up, by performing upward semijoins between the constraint relations, thus keeping the size of the intermediate result small. At the end, if the constraint relation associated with the root atom of the join tree is not empty, then the CSP instance does admit a solution. Therefore, the whole procedure is feasible in $O(n \times r_{max} \times \log r_{max})$, where n is the number of constraints and r_{max} denotes the size of the largest constraint relation.

In addition to the polynomial time algorithm for deciding whether a CSP admits a solution, acyclic instances enjoy further desirable properties:

Acyclicity is efficiently recognizable: Deciding whether a hypergraph is acyclic is feasible in linear time [31] and belongs to the class L (deterministic logspace). Indeed, this follows from the fact that hypergraph acyclicity belongs to SL [16], and that SL is equal to L [27].

Acyclic instances can be efficiently solved: After the bottom-up step described above, one can perform the reverse top-down step by filtering each child vertex from those tuples that do not match with its parent tuples. The relations obtained after the top-down step enjoy the *global consistency* property, i.e., they contain only tuples whose values are part of some solution of the CSP. Then, all solutions can be computed with a backtrack-free procedure, and thus in *total polynomial time*, i.e., in time polynomial in the input plus the output [34] (and actually also with polynomial delay). Alternatively, one may enforce pairwise consistency by taking the semijoins between all pairs of relations until a fixpoint is reached. Indeed, acyclic instances that fulfil this property also fulfil the global consistency property [2].

Acyclic instances are parallelizable: It has been shown that solving acyclic CSP instances is highly parallelizable, as this problem (actually, deciding the existence of a solution) is complete for the low complexity class LOGCFL [16]. Efficient parallel algorithms are discussed in [16] and [17].

We conclude this section by recalling that the above desirable properties of acyclic CSP instances have profitably been exploited in various application scenarios. Indeed, besides their application in the context of Database Theory, they found applications in Game Theory [14,8], Knowledge Representation and Reasoning [21], and Electronic Commerce [13], just to name a few.

3 Generalizing Acyclicity

Many attempts have been made in the literature for extending the good results about acyclic instances to relevant classes of *nearly acyclic* structures. We call these techniques *structural decomposition methods*, because they are based on the "acyclicization" of cyclic (hyper)graphs. We refer the interested reader to [29] for a detailed description of how these techniques may be useful for constraint satisfaction problems and to [22] for further results about graph-based techniques, when relational structures are represented according to various graph representations (primal graph, dual graph, incidence-graph encoding). We also want to mention recent methods such as Spread-cuts [7] and fractional hypertree decompositions [23].

A survey of most of these techniques is currently available in Wikipedia (look for "decomposition method", at http://www.wikipedia.org). In the sequel, we shall briefly overview the tree and hypertree decomposition methods.

3.1 Tree Decompositions

For classes of instances having only binary constraints or, more generally, constraints whose scopes have a fixed maximum arity, the most powerful structural method is based on the notion of treewidth.

Definition 1 ([28]). A *tree decomposition* of a graph $G = (V, E)$ is a pair $\langle T, \chi \rangle$, where $T = (N, F)$ is a tree, and χ is a labelling function assigning with each vertex $p \in N$ a set of vertices $\chi(p) \subseteq V$ such that the following conditions are satisfied: (1) for each node b of G, there exists $p \in N$ such that $b \in \chi(p)$; (2) for each edge $(b, d) \in E$, there exists $p \in N$ such that $\{b, d\} \subseteq \chi(p)$; and, (3) for each node b of G, the set $\{p \in N \mid b \in \chi(p)\}$ induces a connected subtree of T (*connectedness condition*). The *width* of $\langle T, \chi \rangle$ is the number $\max_{p \in N}(|\chi(p)| - 1)$. The *treewidth* of G, denoted by $tw(G)$, is the minimum width over all its tree decompositions. □

It is well-known that a graph G is acyclic if and only if $tw(G) = 1$. Moreover, for any fixed natural number $k > 0$, deciding whether $tw(G) \leq k$ is feasible in linear time [5].

Any CSP with primal graph G such that $tw(G) \leq k$ can be (efficiently) turned into an equivalent CSP whose primal graph is acyclic. Let $\mathcal{I} = \langle Var, U, \mathcal{C} \rangle$ be a CSP instance, let G be the primal graph of $\mathcal{H}(\mathcal{I})$, and let $\langle T, \chi \rangle$ be a tree decomposition of G having width k. We may build a new acyclic CSP instance $\mathcal{I}' = \langle Var, U, \mathcal{C}' \rangle$ over the same variables and universe as $\mathcal{I}$, but with a different set of constraints $\mathcal{C}'$, as follows. Firstly, for each vertex v of T, we create a constraint (S_v, r_v), where $S_v = \chi(v)$ and $r_v = U^{|\chi(v)|}$. Then, for every constraint $(S, r) \in \mathcal{C}$ of the original problem such that $S \subseteq \chi(v)$, we eliminate from r_v all those tuples that do not match with r. The resulting constraint is then added to $\mathcal{C}'$. It can be shown that $\mathcal{I}'$ has the same solutions as $\mathcal{I}$, and that it is acyclic. In fact, observe that, by construction, $\langle T, \chi \rangle$ is a join tree of the hypergraph $\mathcal{H}(\mathcal{I}')$ associated with $\mathcal{I}'$, because of the connectedness condition of tree decompositions. Furthermore, building $\mathcal{I}'$ from $\mathcal{I}$ is feasible in $O(n \times |U|^{k+1})$ where n is the number of vertices in T, and where the size of the largest constraint relation in the resulting instance is $|U|^{k+1}$. Since one can always consider only tree decompositions whose number of vertices is bounded by the number of variables of the problem (i.e., the nodes of the graph), it follows that deciding whether $\mathcal{I}'$ (and hence $\mathcal{I}$) is satisfiable is feasible in $O(|Var| \times |U|^{k+1} \times \log |U|^{k+1})$. In fact, as for acyclic instances, even in this case we may compute also solutions for $\mathcal{I}$ with a backtrack-free search, after the preprocessing of the instance performed according to the given tree decomposition (i.e., according to the join tree of the equivalent acyclic instance). As a consequence, all classes of CSP instances (with primal graphs) having bounded treewidth may be solved in polynomial time, even if with an exponential dependency on the treewidth.

Clearly enough, this technique is not very useful for CSP instances with large constraint scopes. In particular, the class of CSP instances whose associated constraint hypergraphs are acyclic are not tractable according to tree decompositions, because acyclic hypergraphs may have unbounded treewidth. Intuitively, in the primal graph all variables occurring in the same constraint scope are connected to each other, and thus they lead to a clique in the graph. It follows that CSP instances having constraint scopes with large arities have large treewidths, too, because the treewidth of a clique of n nodes is $n-1$. As an example, Figure 2 reports the graph $G(\mathcal{H}_1)$ associated with the acyclic hypergraph $\mathcal{H}_1$, where one may notice how the hyperedge $\{A, C, D, E, F, G, H\}$ is flattened into a clique over all its variables.

3.2 Hypertree Decompositions

Let us now turn our attention to hypergraph based decompositions. Such decompositions are similar to tree decompositions, but they use an additional covering of each set $\chi(p)$ with as few as possible hyperedges. The width is then no longer defined as the maximum cardinality of $\chi(p)$ over all decomposition nodes p, but as the maximum number of hyperedges used to cover $\chi(p)$. Intuitively, this notion of width is better, because it will allow us to expresses more accurately the computational effort needed to transform an instance into an acyclic one.

Definition 2 ([19]). A *generalized hypertree decomposition* of a hypergraph $\mathcal{H}$ is a triple $HD = \langle T, \chi, \lambda \rangle$, where $\langle T, \chi \rangle$ is a tree decomposition of the primal graph of $\mathcal{H}$, and λ is a labelling of the tree T by sets of hyperedges of $\mathcal{H}$ such that, for each vertex $p \in \mathit{vertices}(T)$, $\chi(p) \subseteq \bigcup_{h \in \lambda(v)} h$. That is, all variables in the χ labeling are covered by hyperedges (scopes) in the λ labeling. The *width* of HD is the number $\max_{p \in \mathit{vertices}(T)}(|\lambda(p)|)$. The *generalized hypertree width* of $\mathcal{H}$, denoted by $\mathit{ghw}(\mathcal{H})$, is the minimum width over all its generalized hypertree decompositions. If I is a CSP instance then $\mathit{ghw}(I) := \mathit{ghw}(\mathcal{H}(I))$. □

Clearly, for each CSP instance I, $\mathit{ghw}(I) \leq \mathit{tw}(I)$. Moreover, there are classes of CSPs having unbounded treewidth whose generalized hypertree width is bounded[19].

Finding a suitable tree decomposition whose sets $\chi(p)$ may each be covered with a few hyperedges seems to be quite a hard task even in case we have some fixed upper bound k. Indeed, it has been shown that deciding whether $\mathit{ghw}(\mathcal{H}) \leq k$ is NP-complete (for any fixed $k \geq 3$) [20]. Fortunately, since its first proposal in [18], this notion comes with a tractable variant, called hypertree decomposition, whose associated width is at most 3 times (+1) larger than the generalized hypertree width [1]. As a consequence, it can be shown that every class of CSPs that is tractable according to generalized hypertree width is tractable according to hypertree width, as well.

Definition 3 ([18]). A *hypertree decomposition* of a hypergraph $\mathcal{H}$ is a generalized hypertree decomposition $HD = \langle T, \chi, \lambda \rangle$ that satisfies the following additional condition, called *Descendant Condition* or also *special condition*: $\forall p \in vertices(T)$, $\forall h \in \lambda(p)$, $h \cap \chi(T_p) \subseteq \chi(p)$, where T_p denotes the subtree of T rooted at p, and $\chi(T_p)$ the set of all variables occurring in the χ labeling of this subtree.

The *hypertree width* $hw(\mathcal{H})$ of $\mathcal{H}$ is the minimum width over all its hypertree decompositions. □

As an example, on the right part of Figure 1 a hypertree decomposition of the hypergraph $\mathcal{H}_{cp}$ in Example 1 is reported. Note that this decomposition has width 2.

We refer the interested reader to [18,29] for more details about this notion and in particular about the descendant condition. Here, we just observe that the notions of hypertree width and generalized hypertree width are true generalizations of acyclicity, as the acyclic hypergraphs are precisely those hypergraphs having hypertree width and generalized hypertree width one. In particular, the classes of CSP instances having bounded (generalized) hypertree width have the same desirable computational properties as acyclic CSPs [16]. Indeed, from a CSP instance $\mathcal{I} = \langle Var, U, \mathcal{C} \rangle$ and a (generalized) hypertree decomposition HD of $\mathcal{H}(\mathcal{I})$ of width k, we may build an acyclic CSP instance $\mathcal{I}' = \langle Var, U, \mathcal{C}' \rangle$ with the same solutions as $\mathcal{I}$. The overall cost of deciding whether $\mathcal{I}$ is satisfiable is in this case $O((m-1) \times r_{max}^k \times \log r_{max}^k)$, where r_{max} denotes the size of the largest constraint relation and m is the number of vertices of the decomposition tree, with $m \leq |Var|$ (in that we may always find decompositions in a suitable normal form without redundancies, so that the number of vertices in the tree cannot exceed the number of variables of the given instance). To be complete, if the input consists of $\mathcal{I}$ only, we have to compute the decomposition, too. This can be done with a guaranteed polynomial-time upper bound in the case of hypertree decompositions [18].

In the following two sections, we provide some tractability results for optimization problems. For the sake of presentation, we give algorithms for the acyclic case, provided that these results may be clearly extended to any class of instances having bounded (generalized) hypertree width, after the above mentioned polynomial-time transformation.

4 Optimization Problems over CSP Solutions

In this section, we consider optimization problems where an assignment has to be singled out that satisfies all the constraints of the underlying CSP instance and that has minimum total cost; in other words, we look for a "best" solution among all the possible solutions. In particular, below, we shall firstly address the case where each possible variable-value mapping is associated with a cost (also called constraint satisfaction optimization problem); then we shall consider the case where costs are defined over the constraints tuples (weighted CSP).

```
Input: An acyclic CSOP instance $\langle \mathcal{I}, w\rangle$ with $\mathcal{I} = \langle Var, U, \mathcal{C}\rangle$, $\mathcal{C} = \{(S_1, r_1), ..., (S_q, r_q)\}$,
       and a join tree $T = (N, E)$ of the hypergraph $\mathcal{H}(\mathcal{I})$;
Output: A solution to $\langle \mathcal{I}, w\rangle$;
var $t^*$ : $Var \mapsto U$;
    $\ell^v_{t_v}$ : rational number, for each tuple $t_v \in r_v$;
    $t_{t_v,c}$ : tuple in $r_c$, for each tuple $t_v \in r_v$, and for each $(v, c) \in E$;

Procedure $BottomUp$;
begin
   $Done$ := the set of all the leaves of $T$;
   while $\exists v \in T$ such that (i) $v \notin Done$, and (ii) $\{c \mid c$ is child of $v\} \subseteq Done$ do
      $r_v := r_v - \{t_v \mid \exists(v, c) \in E$ such that $\forall t_c \in \theta_c,\ t_v \not\approx\ t_c\}$;
      if $r_v = \emptyset$ then EXIT;   (* $\mathcal{I}$ is not satisfiable *)
      for each $t_v \in r_v$ do
         $\ell^v_{t_v} := w(t_v)$;
         for each $c$ such that $(v, c) \in E$ do
            $\bar{t}_c := \arg\min_{t_c \in r_c | t_v \approx\ t_c} \left(\ell^c_{t_c} - w(t_c \cap t_v)\right)$ ;
            $t_{t_v,c} := \bar{t}_c$;   (* set best solution *)
            $\ell^v_{t_v} := \ell^v_{t_v} + \ell^c_{\bar{t}_c} - w(\bar{t}_c \cap t_v)$;
         end for
      end for
      $Done := Done \cup \{v\}$;
   end while
end;

begin (* MAIN *)
   $BottomUp$;
   let $r$ be the root of $T$;
   $\bar{t}_r := \arg\min_{t_r \in r_r} \ell^r_{t_r}$;
   $t^* := \bar{t}_r$;   (* include solution *)
   $TopDown(r, \bar{t}_r)$;
   return $t^*$;
end.

Procedure $TopDown(v$ : vertex of $N$, $t_v \in r_v)$;
begin
   for each $c \in N$ s.t. $(v, c) \in E$ do
      $\bar{t}_c := t_{t_v,c}$;
      $t^* := t^* \cup \bar{t}_c$;   (* include solution *)
      $TopDown(c, \bar{t}_c)$;
   end for
end;
```

Fig. 3. **Algorithm** ComputeOptimalSolution

4.1 Constraint Satisfaction Optimization Problems

An instance of a *constraint satisfaction optimization problem* (CSOP) consists of a pair $\langle \mathcal{I}, w\rangle$, where $\mathcal{I} = \langle Var, U, \mathcal{C}\rangle$ is a CSP instance and where $w : Var \times U \mapsto \mathbb{Q}$ is a function mapping substitutions for individual variables to rational numbers. For a substitution $\{X_1/u_1, ..., X_n/u_n\}$, we denote by $w(\{X_1/u_1, ..., X_n/u_n\})$ the value $\sum_{i=1}^{n} w(X_i, u_i)$. Then, a solution to a CSOP instance $\langle \mathcal{I}, w\rangle$ is a solution θ to $\mathcal{I}$ such that $w(\theta) \leq w(\theta')$, for each solution θ' to $\mathcal{I}$. Details on this framework can be found, e.g., in [32].

Constraint satisfaction optimization problems naturally arise in various application contexts. As an example they have recently been used in the context of combinatorial auctions [13], in order to model and solve the *winner determination problem* of determining the allocation of the items among the bidders that maximizes the sum of the accepted bid prices. In particular, in [13], it has been observed that CSOPs and, in particular, the winner determination problem, can be solved in polynomial time on some classes of acyclic instances via a dynamic programming algorithm founded on the ideas of [34]. This algorithm, named ComputeOptimalSolution, is reported in Figure 3 and will be briefly illustrated in the following.

The algorithm receives in input the instance $\langle \mathcal{I}, w\rangle$ and a join tree $T = (N, E)$ for $\mathcal{H}(\mathcal{I})$. Recall that each vertex $v \in N$ corresponds to a hyperedge of $\mathcal{H}(\mathcal{I})$ and, in its turn, to a constraint in $\mathcal{C}$; hence, we shall simply denote by (S_v, r_v) the constraint in $\mathcal{C}$ univocally associated with vertex v.

Based on $\langle \mathcal{I}, w\rangle$ and T, ComputeOptimalSolution computes an optimal solution (or checks that there is no solution) by looking for the "conformance" of the tuples in each relation r_v with the tuples in r_c, for each child c of v in T, where $t_v \in r_v$ is said to *conform* with $t_c \in r_c$, denoted by $t_v \approx t_c$, if for each $X \in S_v \cap S_c$, $X/u \in t_v \Leftrightarrow X/u \in t_c$. In more detail, ComputeOptimalSolution solves $\langle \mathcal{I}, w\rangle$ by traversing T in two phases. First, vertices of T are processed from the leaves to the root r, by means of the procedure *BottomUp* that updates the weight $\ell^v_{t_v}$ of the current vertex v. Intuitively, $\ell^v_{t_v}$ stores the cost of the best partial solution for $\mathcal{I}$ computed by using only the variables occurring in the subtree rooted at v. Indeed, if v is a leaf, then $\ell^v_{t_v} = w(t_v)$. Otherwise, for each child c of v in T, $\ell^v_{t_v}$ is updated by adding the minimum value $\ell^c_{t_c} - w(t_c \cap t_v)$ over all tuples t_c conforming with t_v. The tuple $\bar{t}_c$ for which this minimum is achieved is stored in the variable $t_{t_v,c}$ (resolving ties arbitrarily). Note that if this process cannot be completed, because there is no tuple in r_v conforming with some tuple in each relation associated with the children of v, then we may conclude that $\mathcal{I}$ is not does not admit any solution. Otherwise, after the root $r \in N$ is reached, this part ends, and the top-down phase may start.

In this second phase, the tree T is processed starting from the root. Firstly, the assignment t^* is defined as the tuple in r_r with the minimum cost over all the tuples in r_r (again, resolving ties arbitrarily). Then, procedure *TopDown* extends t^* with a tuple for each vertex of T: at each vertex v and for each child c of v, t^* is extended with the tuple $t_{t_v,c}$ resulting from the bottom-up phase.

Being based on a standard dynamic programming scheme, correctness of ComputeOptimalSolution can be shown by structural induction on the subtrees of T [13]. Moreover, by analyzing its running time, one may note that dealing with cost functions does not (asymptotically) provide any overhead w.r.t. Yannakakis's algorithm [34] for plain CSPs. Following [13], the following can be shown for the more general case of CSOP instances having bounded generalized hypertree-width hypergraphs.[1]

Theorem 1. *Let $\langle \mathcal{I}, w\rangle$ be a* CSOP *instance and HD a (generalized) hypertree decomposition of $\mathcal{H}(\mathcal{I})$. Moreover, let k be the width of HD and m be the number of vertices in its decomposition tree. Then, a solution to $\langle \mathcal{I}, w\rangle$ can be computed (or it is discovered that no solution exists) in time $O((m-1) \times r^k_{max} \times \log r^k_{max})$, where r_{max} is the size of the largest constraint relation in $\mathcal{I}$.*

[1] In all complexity results, we assume the weighting function w be explicitly listed in the input (otherwise, just add the cost of computing through w all cost values for the variable assignments of the given input instance).

4.2 Weighted CSPs: Costs over Tuples

Let us now turn to study a slight variation of the above scenario, where costs are associated with each *tuple* of the constraint relations, rather than with substitutions for individual variables. In fact, this is the setting of weighted CSPs, a well-known specialization of the more general *valued* CSP framework [30].

Formally, a *weighted* CSP (WCSP) instance consists of a tuple $\langle \mathcal{I}, w_1, ..., w_q \rangle$, where $\mathcal{I} = \langle \mathit{Var}, U, \mathcal{C} \rangle$ with $\mathcal{C} = \{C_1, C_2, \ldots, C_q\}$ is a CSP instance, and where, for each tuple $t_v \in r_v$, $w_v(t_v) \in \mathbb{Q}$ denotes the cost associated with t_v. For a solution $\theta = t_1 \cup ... \cup t_q$ to $\mathcal{I}$, we define $w(\theta) = \sum_{v=1}^{q} w_v(t_v)$ as its associated cost. Then, a solution to $\langle \mathcal{I}, w_1, ..., w_q \rangle$ is a solution θ to $\mathcal{I}$ such that $w(\theta) \leq w(\theta')$, for each solution θ' to $\mathcal{I}$.

A few tractability results for WCSPs (actually, for valued CSPs) are known in the literature when structural restrictions are considered over binary encodings of the constraint hypergraphs. Indeed, it has been observed that WCSPs are tractable when restricted on classes of instances whose associated primal graphs are acyclic or nearly-acyclic (see, e.g., [33,10,26]). However, the primal graph obscures much of the structure of the underlying hypergraph since, for instance, each hyperedge is turned into a clique there—see the discussion in Section 3.

Therefore, whenever constraints have large arities, tractability results for primal graphs are useless, and it becomes then natural to ask whether polynomial-time solvability still holds when moving from (nearly-)acyclic primal graphs to acyclic hypergraphs, possibly associated with very intricate primal graphs. Next, we shall positively answer this question, by simply recasting weighted CSPs as constraint optimization problems, and by subsequently solving them via the algorithm ComputeOptimalSolution. To this end, given a WCSP instance $\langle \mathcal{I}, w_1, ..., w_q \rangle$, we define its associated CSOP instance, denoted by CSOP($\langle \mathcal{I}, w_1, ..., w_q \rangle$), as the pair $\langle \mathcal{I}', w' \rangle$ with $\mathcal{I}' = \langle \mathit{Var}', U', \mathcal{C}' \rangle$ such that:

- $\mathit{Var}' = \mathit{Var} \cup \{D_1, ..., D_q\}$, where each D_v is a fresh auxiliary variable in $\mathcal{I}'$;
- $U' = U \cup \bigcup_{v=1}^{q} \bigcup_{t_v \in r_v} \{u_{t_v}\}$, i.e., for each constraint $(S_v, r_v) \in \mathcal{C}$, U' contains a fresh value for each tuple in r_v—intuitively, mapping the variable D_v to u_{t_v} encodes that the tuple t_v is going to contribute to a solution for $\mathcal{I}$;
- $\mathcal{C}' = \{(S_v \cup \{D_v\}, r'_v) \mid (S_v, r_v) \in \mathcal{C}\}$, where $r'_v = \{t_v \cup \{D_v/u_{t_v}\} \mid t_v \in r_v\}$;
- $w'(X/u) = w_v(t_v)$ if $X = D_v$ and $u = u_{t_v}$, for some tuple $t_v \in r_v$; otherwise, $w'(X/u) = 0$. That is, the whole cost of each tuple is determined by the mapping of its associated fresh variable D_v.

It is immediate to check that the above transformation is feasible in linear time. In addition, the transformation enjoys two relevant preservation properties: Firstly, it preserves the structural properties of the WCSP instance in that $\mathcal{H}(\mathcal{I}')$ is acyclic if and only if $\mathcal{H}(P)$ is acyclic; and secondly, it preserves its solutions, in that $\theta' = t'_1 \cup ... \cup t'_q$ is a solution to $\langle \mathcal{I}, w_1, ..., w_q \rangle$ if and only if $\theta = t_1 \cup ... \cup t_q$ is a solution to $\langle \mathcal{I}', w' \rangle$, where $t'_v = t_v \cup \{D_v/u_{t_v}\}$ for each $1 \leq v \leq q$. By exploiting these observations and Theorem 1, the following can be established.

Theorem 2. *Let $\langle \mathcal{I}, w_1, ..., w_q \rangle$ be a* WCSP *instance and HD a (generalized) hypertree decomposition of $\mathcal{H}(\mathcal{I})$. Moreover, let k be the width of HD and m be the number of vertices in its decomposition tree. Then, a solution to $\langle \mathcal{I}, w_1, ..., w_q \rangle$ can be computed (or we may state that there is no solution) in time $O((m-1) \times r_{max}^k \times \log r_{max}^k)$, where r_{max} is the size of the largest constraint relation in $\mathcal{I}$.*

5 Minimizing the Number of Violated Constraints

In this section, we shall complete our picture by considering those scenarios where problems might possibly be overconstrained and where, hence, the focus is on finding assignment minimizing the total number of violated constraints. These kinds of problems are usually referred to in the literature as Max-CSPs [12], which similarly as WCSPs are specializations of valued CSPs.

Formally, let $\theta : \mathit{Var} \mapsto U$ be an assignment for a CSP instance $\mathcal{I} = \langle \mathit{Var}, U, \mathcal{C} \rangle$. We say that the violation degree of θ, denoted by $\delta(\theta)$, is the number of relations r_v such that there is no tuple $t_v \in r_v$ with $t_v \subseteq \theta$. An assignment $\theta : \mathit{Var} \mapsto U$ is a solution to the Max-CSP instance (associated with $\mathcal{I}$) if $\delta(\theta) \leq \delta(\theta')$, for each assignment $\theta' : \mathit{Var} \mapsto U$. Note that Max-CSPs instances, by definition, do always have a solution.

5.1 Acyclic Instances Remain Intractable

After the tractability results established in Section 4.2 for WCSPs, one may expect good news for Max-CSPs, too. Surprisingly, this is not the case.

Theorem 3. *Solving* Max-CSPs *is* NP*-hard, even when restricted over classes of instances with acyclic constraint hypergraphs.*

Proof. Consider any class $\mathcal{T}$ of CSPs instances having an NP-hard satisfiability problem. Then, let $\mathcal{T}'$ be a new class of Max-CSP instances such that, for each $\mathcal{I} = \langle \mathit{Var}, U, \mathcal{C} \rangle \in \mathcal{T}$, $\mathcal{T}'$ contains an instance $\mathcal{I}' = \langle \mathit{Var}, U, \mathcal{C}' \rangle$ with $\mathcal{C}' = \mathcal{C} \cup \{(\mathit{Var}, \emptyset)\}$. That is, any instance $\mathcal{I}' \in \mathcal{T}'$ has a constraint over all variables with an empty constraint relation, and thus it is not satisfiable. Moreover, because of the big hyperedge associated with such a constraint, its hypergraph $\mathcal{H}(\mathcal{I}')$ is trivially acyclic. Also, by construction, there is an assignment for $\mathcal{I}'$ violating only one constraint if and only if $\mathcal{I}$ is satisfiable. It follows that finding an assignment minimizing the total number of violated constraints is NP-hard on the class of acyclic instances $\mathcal{T}'$. □

5.2 Incidence Graphs and Tractable Cases

Given that hypergraph acyclicity and hence its generalizations are not sufficient for guaranteeing the tractability of Max-CSPs, it makes sense to explore acyclicity properties related to suitable graph representations. In fact, as observed in Section 4.2, it is well-known that valued CSPs (and, hence, Max-CSPs) are tractable over acyclic primal graphs (e.g., [33,10,26]). More precisely, tractability has been observed in the literature to hold over primal graphs having bounded *treewidth* (see Section 3). Our main result in this section is precisely to show that tractability still holds in case the *incidence graph* of $\mathcal{H}(\mathcal{I})$ has bounded treewidth, which is a more general condition than the bounded treewidth of primal graphs and which can be used to establish better complexity bounds and to enlarge the class of tractable instances [22]. The fact that the standard CSP is tractable for instances whose incidence graphs have bounded treewidth was already shown in [6]. We here extend this tractability result to Max-CSPs.

Recall that the incidence encoding of a hypergraph $\mathcal{H}$, denoted by $inc(\mathcal{H}) = (N, E)$, is the bipartite graph where $N = \mathcal{E}(\mathcal{H}) \cup \mathcal{N}(\mathcal{H})$ and $E = \{ \{h, a\} \mid h \in \mathcal{E}(\mathcal{H})$ and $a \in h)\}$, i.e. it contains an edge between h and a if and only if the variable a occurs in the hyperedge h. As an example, Figure 2 reports on the rightmost part the incidence graph $inc(\mathcal{H}_1)$, where nodes associated with hyperedges in $\mathcal{E}(\mathcal{H}_1)$ are depicted as black circles. Note that the treewidth of $inc(\mathcal{H}_1)$ is 2, which is much smaller than the treewidth of $G(\mathcal{H}_1)$. This does not happen by chance since, for each hypergraph $\mathcal{H}$, it holds that $tw(inc(\mathcal{H})) \leq tw(G(\mathcal{H}))$; in addition, there are also classes of hypergraphs with incidence encodings of bounded treewidth and primal encodings of unbounded treewdith (see, e.g., [22]).

While enlarging the class of instances having bounded treewidth, the incidence encoding still conveys all the information needed to solve Max-CSP instances. Again, the solution algorithm consists of a transformation into a suitable CSOP instance. Formally, let $\mathcal{I} = \langle Var, U, \mathcal{C}\rangle$ be a Max-CSP instance with $\mathcal{C} = \{(S_1, r_1), ..., (S_q, r_q)\}$, and let $\langle T, \chi\rangle$ be a k-width tree decomposition of $inc(\mathcal{H}(\mathcal{I}))$—recall that for each vertex $v \in T$, $\chi(v)$ is a set of variables (i.e., nodes of $\mathcal{N}(\mathcal{H}(\mathcal{I}))$) and constraint scopes (i..e, edges in $\mathcal{E}(\mathcal{H}(\mathcal{I}))$). Then, the constraint satisfaction optimization problem instance CSOP$(\mathcal{I}, \langle T, \chi\rangle)$ is the pair $\langle \mathcal{I}', w'\rangle$, where $\mathcal{I}' = \langle Var', U', \mathcal{C}'\rangle$ and such that:

- $Var' = Var \cup \{S_1, ..., S_q\}$, that is, also the constraint scopes of $\mathcal{C}$ belong to the variables of the new problem;
- $U' = U \cup \{unsat\} \cup \{u_t \mid t \in r_i, \text{ for } 1 \leq i \leq q\}$;
- $\mathcal{C}' = \{(\chi(v), r'_v) \mid v \in T\}$ where the constraint relation r'_v is defined as follows. Let $\mu = |\chi(v) \cap Var|$, and let U^μ denote the set of all possible tuples over the μ variables in $\chi(v) \cap Var$. Let also $S_{i_1}, ...S_{i_h}$ be the scope-variables in $\chi(v)$. Then, for each tuple $\theta \in U^\mu$, the relation r'_v contains all tuples $\theta \cup \{S_{i_1}/v_{i_1}\} \cup \cdots \cup \{S_{i_h}/v_{i_h}\}$, where $v_{i_j} \in U$ $(1 \leq j \leq h)$ is a value for the scope-variable S_{i_j} such that: $v_{i_j} = u_t$ if there is a tuple $t \in r_{i_j}$ conforming with θ; and $v_{i_j} = unsat$, if no such a tuple exists in r_{i_j}.
- $w'(X/u) = 0$ if $u \neq unsat$; otherwise $w'(X/u) = 1$, that is, each constraint of $\mathcal{C}$ that is not satisfied increases the cost of a solution by a unitary factor.

Note that this transformation is feasible in time exponential in the width of $\langle T, \chi\rangle$ only. Moreover, solutions of $\mathcal{I}'$ with minimum total cost precisely correspond to assignments over $\mathcal{I}$ minimizing the total number of violated constraints. In fact, the following can be established.

Theorem 4. *Let $\mathcal{I} = \langle Var, U, \mathcal{C}\rangle$ be a* Max-CSP *instance with $tw(inc(\mathcal{H}(\mathcal{I}))) = k$. Then, a solution to $\mathcal{I}$ can be computed in time $O(|Var| \times |U|^{k+1} \times \log |U|^{k+1})$.*

6 Conclusion and Discussion

In this paper, classes of tractable CSOP, WCSP, and Max-CSP instances are singled out by overviewing and proposing solution approaches applicable to instances whose hypergraphs have bounded (generalized) hypertree width, or whose incidence graphs have bounded treewidth. The techniques described in this paper are mainly based on Algorithm ComputeOptimalSolution, which has been designed to optimize costs expressed as rational numbers and combined via the summation operation. However, it is easily seen that it remains correct if costs are specified over an arbitrary totally ordered monoid structure, where some binary operation $\oplus$ (in place of standard summation) is used in order to combine costs, provided it is commutative, associative, closed, and that it verifies identity and monotonicity. It follows that all tractable classes of CSOP, WCSP, and Max-CSP instances identified in this paper remain tractable in such extended scenarios, which indeed emerge with valued CSPs (see, e.g., [4]).

References

1. Adler, I., Gottlob, G., Grohe, M.: Hypertree-Width and Related Hypergraph Invariants. European Journal of Combinatorics 28, 2167–2181 (2007)
2. Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the desirability of acyclic database schemes. Journal of the ACM 30(3), 479–513 (1983)
3. Bernstein, P.A., Goodman, N.: The power of natural semijoins. SIAM Journal on Computing 10(4), 751–771 (1981)
4. Bistarelli, S., Montanari, U., Rossi, F., Schiex, T., Verfaillie, G., Fargier, H.: Semiring-Based CSPs and Valued CSPs: Frameworks, Properties,and Comparison. Constraints 4(3), 199–240 (1999)
5. Bodlaender, H.L., Fomin, F.V.: A Linear-Time Algorithm for Finding Tree Decompositions of Small Treewidth. SIAM Journal on Computing 25(6), 1305–1317 (1996)
6. Chekuri, C., Rajaraman, A.: Conjunctive Query Containment Revisited. MFPS 1985 239(2), 211–229 (2000); Preliminary version in: Schwentick, T., Suciu, D. (eds.): ICDT 2007. LNCS, vol. 4353, pp. 211–229. Springer, Heidelberg (2007); Afrati, F.N., Kolaitis, P.G. (eds.): ICDT 1997. LNCS, vol. 1186, pp. 56–70. Springer, Heidelberg (1996) (Full version)
7. Cohen, D.A., Jeavons, P.G., Gyssens, M.: A unified theory of structural tractability for constraint satisfaction problems. Journal of Computer and System Sciences 74(5), 721–743 (2008)
8. Daskalakis, C., Papadimitriou, C.H.: Computing pure nash equilibria in graphical games via markov random fields. In: Proc. of ACM EC 2006, pp. 91–99 (2006)
9. Fagin, R.: Degrees of acyclicity for hypergraphs and relational database schemes. J. ACM 30(3), 514–550 (1983)
10. de Givry, S., Schiex, T., Verfaillie, G.: Exploiting Tree Decomposition and Soft Local Consistency In Weighted CSP. In: Proc. of AAAI 2006 (2006)
11. Dechter, R.: Constraint Processing. Morgan Kaufmann, San Francisco (2003)
12. Freuder, E.C., Wallace, R.J.: Partial Constraint Satisfaction. Artificial Intelligence 58(1-3), 21–70 (1992)

13. Gottlob, G., Greco, G.: On the complexity of combinatorial auctions: structured item graphs and hypertree decomposition. In: Proc. EC 2007, pp. 152–161 (2007) (full version currently available as Technical Report, University of Calabria)
14. Gottlob, G., Greco, G., Scarcello, F.: Pure Nash Equilibria: Hard and Easy Games. Journal of Artificial Intelligence Research 24, 357–406 (2005)
15. Gottlob, G., Leone, N., Scarcello, F.: A comparison of structural CSP decomposition methods. Artificial Intelligence 124(2), 243–282 (2000)
16. Gottlob, G., Leone, N., Scarcello, F.: The complexity of acyclic conjunctive queries. Journal of the ACM 48(3), 431–498 (2001)
17. Gottlob, G., Leone, N., Scarcello, F.: Advanced parallel algorithms for processing acyclic conjunctive queries, rules, and constraints. In: Proc. of SEKE 2000, pp. 167–176 (2000)
18. Gottlob, G., Leone, N., Scarcello, F.: Hypertree decompositions and tractable queries. J. of Computer and System Sciences 64(3), 579–627 (2002)
19. Gottlob, G., Leone, N., Scarcello, F.: Robbers, marshals, and guards: game theoretic and logical characterizations of hypertree width. J. of Computer and System Sciences 66(4), 775–808 (2003)
20. Gottlob, G., Miklós, Z., Schwentick, T.: Generalized hypertree decompositions: NP-hardness and tractable variants. In: Proc. of PODS 2007, pp. 13–22 (2007)
21. Gottlob, G., Pichler, R., Wei, F.: Bounded Treewidth as a Key to Tractability of Knowledge Representation and Reasoning. In: Proc. of AAAI 2006 (2006)
22. Greco, G., Scarcello, F.: Non-Binary Constraints and Optimal Dual-Graph Representations. In: Proc. of IJCAI 2003, pp. 227–232 (2003)
23. Grohe, M., Marx, D.: Constraint solving via fractional edge covers. In: Proc. of SODA 2006, Miami, Florida, USA, pp. 289–298 (2006)
24. Kask, K., Dechter, R., Larrosa, J., Dechter, A.: Unifying tree decompositions for reasoning in graphical models. Artificial Intelligence 166(1-2), 165–193 (2005)
25. Meseguer, P., Rossi, F., Schiex, T.: Soft Constraints. In: Handbook of Constraint Programming. Elsevier, Amsterdam (2006)
26. Ndiaye, S., Jégou, P., Terrioux, C.: Extending to Soft and Preference Constraints a Framework for Solving Efficiently Structured Problems. In: Proc. of ICTAI 2008, pp. 299–306 (2008)
27. Reingold, O.: Undirected ST-connectivity in log-space. Journal of the ACM 55(4) (2008)
28. Robertson, N., Seymour, P.D.: Graph minors III: Planar tree-width. Journal of Combinatorial Theory, Series B 36, 49–64 (1984)
29. Scarcello, F., Gottlob, G., Greco, G.: Uniform Constraint Satisfaction Problems and Database Theory. In: Creignou, N., Kolaitis, P.G., Vollmer, H. (eds.) Complexity of Constraints. LNCS, vol. 5250, pp. 156–195. Springer, Heidelberg (2008)
30. Schiex, T., Fargier, H., Verfaillie, G.: Valued Constraint Satisfaction Problems: Hard and Easy Problems. In: Proc. of IJCAI 1995, pp. 631–639 (1995)
31. Tarjan, R.E., Yannakakis, M.: Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. SIAM Journal on Computing 13(3), 566–579 (1984)
32. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London (1993)
33. Terrioux, C., J'egou, P.: Bounded Backtracking for the Valued Constraint Satisfaction Problems. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, pp. 709–723. Springer, Heidelberg (2003)
34. Yannakakis, M.: Algorithms for acyclic database schemes. In: Proc. of VLDB 1981, pp. 82–94 (1981)

Deciding Safety Properties in Infinite-State Pi-Calculus via Behavioural Types[⋆]

Lucia Acciai and Michele Boreale

Dipartimento di Sistemi e Informatica
Università di Firenze
{lacciai,boreale}@dsi.unifi.it

Abstract. In the pi-calculus, we consider decidability of certain safety properties expressed in a simple spatial logic. We first introduce a behavioural type system that, given a process P, tries to extract a spatial-behavioural type T, in the form of a ccs term that is logically equivalent to the given process. Using techniques based on well-structured transition systems, we then prove that, for an interesting fragment of the considered logic, satisfiability ($T \models \phi$) is decidable for types. As a consequence of logical equivalence between types and processes, we obtain decidability of this fragment of the logic for all well-typed pi-processes.

Keywords: pi-calculus, behavioural types, spatial logic, decidability, safety.

1 Introduction

In recent years, *spatial logic* [6] and *behavioural type systems* [10,7,1] have gained attention as useful tools for the analysis of concurrent systems described in process calculi. Spatial logics are well suited to express properties related to concurrency and distribution, thanks to a combination of spatial and dynamic connectives. An example is the property expressing race-freedom on some channel a: “it is never the case that there are two concurrent outputs ready at channel a”. behavioural type systems are used in order to obtain abstract representation of message-passing systems and simplify their analysis. In Igarashi and Kobayashi’s work on generic type systems [10], pi-calculus processes are abstracted by means of ccs types. The main property of Igarashi and Kobayashi’s system is *type soundness*: any safety property satisfied by a type is also satisfied by processes that inhabit that type.

In [1], we have combined ideas from spatial logics and behavioural type system into a single framework. Like in [10], the language of processes we consider is the pi-calculus, while types are ccs terms. Differently from [10], though, types of [1] account for both the behavioural *and* the spatial structure of processes. This fact allows one to establish a precise correspondence between processes and their types. This correspondence makes it possible to prove type soundness theorems holding for fairly general classes of properties, not only safety invariants, although this enhancement comes at some price in terms of flexibility of the type system w.r.t. [10]. A prominent feature of [1] is that *structural congruence* is used as a subtyping relation.

[⋆] Research partly supported by the EU within the FET-GC2 initiative, project Sensoria.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 31–42, 2009.

A driving motivation in all the mentioned works is being able to combine type- and model-checking. The idea is that, rather than model checking a given property against a process, with a behavioural type system at hand, one checks the property against a simpler model: a type. Moving from processes to types certainly implies a gain in simplicity in terms of reasoning [10,1]. Unfortunately, in [11], undecidability of behavioural type systems using the simulation preorder as a sub-typing relation has been proven. The result suggests that any "reasonable" instances of the generic system of [10] based on simulation preorders might turn out to be undecidable. We may hope the situation is better for our system in [1], because this system adopts structural congruence as a subtyping relation, that, for the considered languages, is easily seen to be decidable.

In the present paper, our goal is to show decidability of a fragment of Spatial Logic over a pi-calculus with replication, introduced in Section 2. The fragment in question is expressive enough to capture interesting safety invariants. We achieve our goal in two steps. In the first one, we devise a behavioural type system whose purpose is, basically, to extract behavioural ccs types T out of given processes P. The types extracted this way are logically equivalent to the original processes. This part of the work is based on behavioural type techniques similar to those discussed in [1] and is reported in Section 3.

In the second one (Section 6), we show that it is actually decidable whether a ccs type T satisfies a formula in the fragment introduced in Section 4. This part, which is largely independent from the first one, heavily relies on the technique of *well-structured transition systems* (wsts) introduced by Finkel and Schnoebelen [8] and overviewed in Section 5. Our result generalizes a previous result by Busi et al. [4], who had proven decidability in ccs with replication of *weak barbs*. As a corollary of the logical correspondence given by the type system, decidability of the considered logic carries over to well-typed pi-processes.

It is worth to stress that, in the economy of the proof, being able to go from the pi-calculus to ccs, via the behavioural type system, is crucial. In particular, the wsts technique does not apply to pi-calculus directly. The technical reason is that there is no upper bound on the nesting depth of restrictions in pi-terms as they evolve, a fact that prevents the definition of a syntax-based wqo in pi-calculus. Instead, there is such a bound for ccs.

2 Processes

The language we consider is a synchronous polyadic pi-calculus [13] with guarded summations and replications. We presuppose a countable set of *names* $\mathcal{N}$ and let $a,b,\ldots,x,\ldots$ range over names. Processes $P,Q,R,\ldots$ are defined by the grammar below

$$\alpha ::= a(\tilde{b}) \,\big|\, \overline{a}\langle\tilde{b}\rangle \,\big|\, \tau \qquad P ::= \textstyle\sum_{i\in I} \alpha_i.P_i \,\big|\, P|P \,\big|\, (\nu b : \mathsf{t})P \,\big|\, !a(\tilde{b}).P$$

where $\tilde{b}$ is a tuple of names and $\mathsf{t} = (\tilde{x} : \tilde{\mathsf{t}}')T$ is a *channel type* where: $(\tilde{x} : \tilde{\mathsf{t}})$ is a binder with scope T; $\tilde{x}$ and $\tilde{\mathsf{t}}$ represent the formal parameters and types of objects carried by the channel; T is a process type (see Section 3) prescribing a usage of those parameters. The calculus is equipped with standard notions of free and bound names (fn(·), bn(·)). Notice that we let $\mathrm{fn}((\nu b : \mathsf{t})P) = (\mathrm{fn}(P) \cup \mathrm{fn}(\mathsf{t})) \setminus \{b\}$ and that terms are identified up to alpha-equivalence, defined as usual. To prevent arity mismatch, we will only consider well-sorted terms in some fixed sorting system (see e.g. [13]), and call $\mathcal{P}$ the resulting set of *processes*.

Table 1. Laws for structural congruence $\equiv$ on processes

$$(\nu y)\mathbf{0} \equiv \mathbf{0} \quad (P|Q)|R \equiv P|(Q|R) \quad P|Q \equiv Q|P \quad P|\mathbf{0} \equiv P \quad (\nu x:\mathsf{t})P|Q \equiv (\nu x:\mathsf{t})(P|Q) \text{ if } \tilde{x} \notin \mathrm{fn}(Q)$$

Table 2. Rules for the reduction relation $\rightarrow$ on processes

$$(\textsc{com})\frac{\alpha_l = a(\tilde{x}) \quad \alpha'_n = \overline{a}\langle\tilde{b}\rangle \quad l \in I \quad n \in J}{\sum_{i\in I}\alpha_i.P_i | \sum_{j\in J}\alpha'_j.Q_j \rightarrow P_l[\tilde{b}/\tilde{x}]|Q_n} \quad (\textsc{tau})\frac{j \in I \quad \alpha_j = \tau}{\sum_{i\in I}\alpha_i.P_i \rightarrow P_j} \quad (\textsc{res})\frac{P \rightarrow P'}{(\nu x:\mathsf{t})P \rightarrow (\nu x:\mathsf{t})P'}$$

$$(\textsc{rep})\frac{\alpha_n = \overline{a}\langle\tilde{b}\rangle \quad n \in J}{!a(\tilde{x}).P|\sum_{j\in J}\alpha_j.Q_j \rightarrow !a(\tilde{x}).P|P[\tilde{b}/\tilde{x}]|Q_n} \quad (\textsc{par})\frac{P \rightarrow P'}{P|Q \rightarrow P'|Q} \quad (\textsc{struct})\frac{P \equiv Q \quad Q \rightarrow Q' \quad Q' \equiv P'}{P \rightarrow P'}$$

In the following, we write $\mathbf{0}$ for the empty summation, omit trailing $\mathbf{0}$'s and sometimes abbreviate $(\nu b_1 : \mathsf{t}_1)\cdots(\nu b_n : \mathsf{t}_n)P$ as $(\nu\tilde{b}_i : \tilde{\mathsf{t}}_i)_{i\in 1..n}P$, or $(\nu\tilde{b} : \tilde{\mathsf{t}})P$, or $(\nu\tilde{b})P$.

Over $\mathcal{P}$, we define a *reduction semantics*, based as usual on a notion of structural congruence and on a reduction relation. These relations are defined as the least congruence $\equiv$ and as the least relation $\rightarrow$ generated by the axioms in Table 1 and Table 2, respectively. Concerning Table 1, note that we have dropped the law $(\nu x : \mathsf{t})(\nu y : \mathsf{t}')P = (\nu y : \mathsf{t}')(\nu x : \mathsf{t})P$, which allows one to swap restrictions: the reason is that swapping t and t', which may contain free names, would require unpleasant side conditions. The rules in Table 2 are standard.

In the sequel, we say that a process P has a *barb* a (written $P\searrow_{\overline{a}}$) if $P\equiv(\nu\tilde{b})(\sum_i \alpha_i.P_i + \overline{a}.Q|R)$, with $a \notin \tilde{b}$. $P \searrow_a$ is defined similarly. By $P \xrightarrow{\langle a\rangle} Q$ we denote a reduction $P \rightarrow Q$ arising from a synchronization on the channel name (subject) $a \in \mathrm{fn}(P)$.

3 Type System

Types. Types are essentially ccs terms, bearing some extra annotation on input prefixes and restrictions. Let $\mathfrak{a}, \mathfrak{b}, \ldots$ range over finite set of names. The set $\mathcal{T}$ of types is generated by the following grammar:

$$\mu ::= a^{\mathfrak{a}} \mid \overline{a} \mid \tau \qquad T, S, U ::= \textstyle\sum_{i\in I}\mu_i.T_i \mid !a^{\mathfrak{a}}.T \mid T|T \mid (\nu a^{\mathfrak{a}})T\,.$$

In $a^{\mathfrak{a}}.T$ and $(\nu a^{\mathfrak{a}})T$, the annotations $\mathfrak{a}$ contribute to the set of free names of a type, indeed $\mathrm{fn}(a^{\mathfrak{a}}.S) = \{a\} \cup \mathfrak{a} \cup \mathrm{fn}(S)$. In the type system, annotations will be employed so as to ensure that for processes P and their types T scope extrusion, hence structural congruence, works in the same manner in both P and in T (see [1] for more details). In the sequel, we shall often omit the channel type $()\mathbf{0}$, writing e.g. $(x)\overline{x}$ instead of $(x : ()\mathbf{0})\overline{x}$, and annotations on input prefixes and restrictions when unnecessary. We will often denote guarded summations and replications by the letters $G, F, \ldots$. Notions of free and bound names ($\mathrm{fn}(\cdot)$ and $\mathrm{bn}(\cdot)$), alpha-equivalence, structural congruence and reduction for types parallel those of processes.

Typing rules. Judgements of type system are of the form $\Gamma \vdash P : T$, where: $P \in \mathcal{P}$, $T \in \mathcal{T}$ and Γ is a *context*: a finite partial map from names to channel types. We write $\Gamma \vdash a : \mathsf{t}$ if

Table 3. Typing rules for the local system

$$\text{(T-Inp)}\ \frac{\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})T \quad \mathrm{fn}(\tilde{\mathsf{t}}) \cup \mathrm{fn}(T) \setminus \tilde{x} = \mathfrak{a} \qquad \Gamma, \tilde{x} : \tilde{\mathsf{t}} \vdash P : T|T' \quad \tilde{x} \notin \mathrm{fn}(T')}{\Gamma \vdash a(\tilde{x}).P : a^{\mathfrak{a}}.T'} \qquad \text{(T-Out)}\ \frac{\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})T \quad \Gamma \vdash \tilde{b} : \tilde{\mathsf{t}} \quad \Gamma \vdash P : S}{\Gamma \vdash \overline{a}\langle \tilde{b} \rangle.P : \overline{a}.(T[^{\tilde{b}}/_{\tilde{x}}] \,|\, S)}$$

$$\text{(T-Res)}\ \frac{\Gamma, a : \mathsf{t} \vdash P : T \quad \mathfrak{a} = \mathrm{fn}(\mathsf{t})}{\Gamma \vdash (\nu a : \mathsf{t})P : (\nu a^{\mathfrak{a}})T} \qquad \text{(T-Par)}\ \frac{\Gamma \vdash P : T \quad \Gamma \vdash Q : S}{\Gamma \vdash P|Q : T|S} \qquad \text{(T-Eq)}\ \frac{\Gamma \vdash P : T \quad T \equiv S}{\Gamma \vdash P : S}$$

$$\text{(T-Sum)}\ \frac{|I| \neq 1 \quad \forall i \in I :\ \Gamma \vdash \alpha_i.P_i : \mu_i.T_i}{\Gamma \vdash \sum_{i \in I} \alpha_i.P_i : \sum_{i \in I} \mu_i.T_i} \qquad \text{(T-Rep)}\ \frac{\Gamma \vdash a(\tilde{x}).P : a^{\mathfrak{a}}.T}{\Gamma \vdash !a(\tilde{x}).P : !a^{\mathfrak{a}}.T} \qquad \text{(T-Tau)}\ \frac{\Gamma \vdash P : T}{\Gamma \vdash \tau.P : \tau.T}$$

$a \in \mathrm{dom}(\Gamma)$ and $\Gamma(a) = \mathsf{t}$. We say that a context is *well-formed* if whenever $\Gamma \vdash a : (\tilde{x} : \tilde{\mathsf{t}})T$ then $\mathrm{fn}(T, \tilde{\mathsf{t}}) \subseteq \tilde{x} \cup \mathrm{dom}(\Gamma)$. In what follows *we shall only consider well-formed contexts.*

The type system can be thought of as a procedure that, given P, builds a ccs approximation T of P, with a little help from a context Γ prescribing channel usage. See [2] for further details. In the following we say that a process P *is* Γ*-well-typed* if $\Gamma \vdash P : T$ for some $T \in \mathcal{T}$.

Results. This paragraph introduces the main properties of the type system. Theorem 1 and 2 guarantee the reduction-based correspondence between processes and the corresponding types, while Proposition 1 guarantees the structural one. Note that the structural correspondence is shallow, in the sense that in general it breaks down underneath prefixes. Finally, Proposition 2 guarantees decidability of $\vdash$.

Theorem 1 (subject reduction). *$\Gamma \vdash P : T$ and $P \rightarrow P'$ implies that there exists a T' such that $T \rightarrow T'$ and $\Gamma \vdash P' : T'$.*

Theorem 2 (type subject reduction). *$\Gamma \vdash P : T$ and $T \rightarrow T'$ implies that there exists a P' such that $P \rightarrow P'$ and $\Gamma \vdash P' : T'$.*

Proposition 1 (structural correspondence). *Suppose $\Gamma \vdash P : T$.*

1. *$P \searrow_{\alpha}$, with $\alpha ::= a \,\big|\, \overline{a}$, implies $T \searrow_{\alpha}$; vice-versa for T and P.*
2. *$P \equiv (\nu\tilde{a} : \tilde{\mathsf{t}})R$ implies $T \equiv (\nu\tilde{a}^{\tilde{\mathfrak{a}}})S$, with $\tilde{\mathfrak{a}} = \mathrm{fn}(\tilde{\mathsf{t}})$ and $\Gamma, \tilde{a} : \tilde{\mathsf{t}} \vdash R : S$; vice-versa for T and P.*
3. *$P \equiv P_1|P_2$ implies $T \equiv T_1|T_2$, with $\Gamma \vdash P_i : T_i$, for $i = 1, 2$; vice-versa for T and P.*

Proposition 2. *Let Γ be a context. It is decidable whether P is Γ-well-typed.*

4 Shallow Logic and Type-Process Correspondence

The logic for the pi-calculus we introduce below can be regarded as a fragment of Caires and Cardelli's Spatial Logic [6]. In [1] we have christened this fragment *Shallow Logic*, as it allows us to speak about the dynamic as well as the "shallow" spatial structure of processes and types. In particular, the logic does not provide for modalities that allows one to "look underneath" prefixes.

Definitions. The set $\mathcal{F}$ of *Shallow Logic* formulae $\phi, \psi, \ldots$ is given by the grammar $\phi ::= \mathbf{T} \,\big|\, a \,\big|\, \overline{a} \,\big|\, \phi|\phi \,\big|\, \neg\phi \,\big|\, \mathrm{H}^*\phi \,\big|\, \phi \wedge \phi \,\big|\, \phi \vee \phi \,\big|\, \langle a \rangle\phi \,\big|\, \Diamond^*\phi$, where $a \in \mathcal{N}$.

The set of logical operators includes spatial ($a, \overline{a}, |, \mathrm{H}^*$) as well as dynamic ($\langle a \rangle, \Diamond^*$) connectives, beside the usual boolean connectives, including a constant **T** for “true”. We have included both disjunction and conjunction to present more smoothly “monotone” properties, that is, properties whose satisfaction is preserved when adding “more structure” to terms. The names of a formula ϕ, written $\mathrm{n}(\phi)$, are defined as expected. The interpretation of $\mathcal{F}$ over processes and types is given below.

$$\begin{array}{rlrl}
\llbracket \mathbf{T} \rrbracket = \mathcal{U} & & \llbracket \langle a \rangle \phi \rrbracket = \{A & \mid \exists B : A \xrightarrow{\langle a \rangle} B,\ B \in \llbracket \phi \rrbracket\} \\
\llbracket \phi_1 \vee \phi_2 \rrbracket = \llbracket \phi_1 \rrbracket \cup \llbracket \phi_2 \rrbracket & & \llbracket \phi_1 \wedge \phi_2 \rrbracket = \llbracket \phi_1 \rrbracket \cap \llbracket \phi_2 \rrbracket & \\
\llbracket \neg \phi \rrbracket = \mathcal{U} \setminus \llbracket \phi \rrbracket & & \llbracket \mathrm{H}^* \phi \rrbracket = \{A & \mid \exists \tilde{a}, B : A \equiv (\nu \tilde{a})B, \tilde{a} \notin \mathrm{n}(\phi),\ B \in \llbracket \phi \rrbracket\} \\
\llbracket a \rrbracket = \{A \mid A \searrow_a \} & & \llbracket \phi_1 | \phi_2 \rrbracket = \{A & \mid \exists A_1, A_2 : A \equiv A_1 | A_2,\ A_1 \in \llbracket \phi_1 \rrbracket,\ A_2 \in \llbracket \phi_2 \rrbracket\} \\
\llbracket \overline{a} \rrbracket = \{A \mid A \searrow_{\overline{a}} \} & & \llbracket \Diamond^* \phi \rrbracket = \{A & \mid \exists B : A \rightarrow^* B,\ \text{and}\ B \in \llbracket \phi \rrbracket\}
\end{array}$$

We let $\mathcal{U}$ be the set including all processes and all types. We write $A \models \phi$ if $A \in \llbracket \phi \rrbracket$, where $A \in \mathcal{U}$. Connectives and spatial modalities are interpreted as usual. Concerning the dynamic part, $\langle a \rangle \phi$ checks if an interaction with subject a may lead A to a state where ϕ is satisfied; $\Diamond^* \phi$ checks if any number, including zero, of reductions may lead A to a state where ϕ is satisfied. In this paper, we shall mainly focus on safety properties, that is, properties of the form “nothing bad will ever happen”. The following definition is useful to syntactically identify classes of formulae that correspond to safety properties.

Definition 1 (monotone and anti-monotone formulae). *We say a formula ϕ is* monotone *if it does not contain occurrences of $\neg$ and* anti-monotone *if it is of the form $\neg \psi$, with ψ monotone.*

Safety invariants can often be written as anti-monotone formulae $\neg \Diamond^* \psi$ with ψ a monotone formula representing the bad event one does not want to occur. This can also be written as $\Box^* \neg \psi$, where $\Box^* = \neg \Diamond^* \neg$.

Example 1. The following formulae define properties depending on generic names, a and l. $\mathit{NoRace}(a) \stackrel{\triangle}{=} \neg \Diamond^* \mathrm{H}^* (\overline{a} | \overline{a})$ says that it will never be the case that there are two concurrent outputs competing for synchronization on a. $\mathit{Linear}(a) \stackrel{\triangle}{=} \neg \Diamond^* \langle a \rangle \Diamond^* \langle a \rangle$ says that it is never the case that a is used more than once in a computation. In $\mathit{Lock}(a, l) \stackrel{\triangle}{=} \neg \Diamond^* \mathrm{H}^* (l | \langle a \rangle)$, a represents a shared resource and l a lock: this formula says that it is never the case that the resource a is acquired in the presence of l, that is, without prior acquisition of the lock.

Logical correspondence between processes and types. The following theorem is crucial: it basically asserts that, under a condition of well-typing, model checking on processes can be reduced to model checking on types. The proof is based on the structural and operational correspondences seen in Section 3.

Theorem 3 (type-process correspondence). *Suppose $\Gamma \vdash P : T$. Let ϕ be any formula. Then $P \models \phi$ if and only if $T \models \phi$.*

This correspondence can be enhanced by the next result, saying that, under certain circumstances, model checking can be safely carried out against a more abstract version of the type T, with a further potential gain in efficiency. This more abstract version is obtained by “masking”, by means of the $\downarrow_{\tilde{x}}$ operator (see [2] for the details), the free names of the type that are not found in the formula. Moreover, if this masking produces a top-level sub-term in the type with no free name, this term can be safely discarded.

Proposition 3. *(a) Suppose* $\Gamma \vdash P : T$ *and let* ϕ *be an anti-monotone formula with* $\mathrm{n}(\phi) \subseteq \tilde{x}$*. Then* $T \downarrow_{\tilde{x}} \models \phi$ *implies that* $T \models \phi$*. (b) Suppose* $\mathrm{fn}(U) = \emptyset$*. Then, for any* T *and* ϕ*,* $T|U \models \phi$ *if and only if* $T \models \phi$*.*

Example 2. Consider the formula $NoRace(a)$ introduced in Example 1 and the process $P = \overline{b}\langle a\rangle + \overline{a}\,|\,b(x).(\nu c)(\overline{c}\,|\,!c.\overline{x}.\overline{c})\,|\,!a.\overline{f}\,|\,!f.\overline{n}$. Here, at runtime, the number of occurrences of $\overline{n}$ "counts" the number of interactions performed on a. For a suitable Γ, one finds $\Gamma \vdash P : T$, where (ignoring annotations) $T = \overline{b}.(\nu c)(\overline{c}\,|\,!c.\overline{a}.\overline{c}) + \overline{a}\,|\,b\,|\,!a.\overline{f}\,|\,!f.\overline{n}$ and

$$T \downarrow_a = (\overline{b}.(\nu c)(\overline{c}\,|\,!c.\overline{a}.\overline{c}) + \overline{a}\,|\,b\,|\,!a.\overline{f}\,|\,!f.\overline{n}) \downarrow_a = \tau.(\nu c)(\overline{c}\,|\,!c.\overline{a}.\overline{c}) + \overline{a}\,|\,\tau\,|\,!a.\tau\,|\,!\tau.\tau\ .$$

$\tau.(\nu c)(\overline{c}|!c.\overline{a}.\overline{c}) + \overline{a}|!a.\tau \models NoRace(a)$ and $P \models NoRace(a)$ (Proposition 3 and Theorem 3).

5 A Well-Structured Transition System for Behavioural Types

Background. We review below some background material about well-structured transition systems [8] and well quasi-ordering over trees and forests.

Definition 2 (wqo). *Let* S *be a set. A* quasi-ordering *(*QO*, aka* preorder*) on* S *is a reflexive and transitive binary relation over* S*. A* QO $\preceq$ *on* S *is a* well quasi-ordering *(*WQO*) if for any sequence of elements of* S*,* $(s_i)_{i\geq 0}$*, there are* i *and* j*, with* $i < j$*, s.t.* $s_i \preceq s_j$*.*

Recall that a transition system is a pair $Tr = (S, \rightarrow)$, where S is the set of states and $\rightarrow \subseteq S \times S$ is the transition relation. Tr is *finitely-branching* if for each $s \in S$ the set of successors $\{s' | s \rightarrow s'\}$ is finite.

Definition 3 (WSTS, [8]). *A* well-structured transition system *(*WSTS *for short) is a pair* $\mathcal{W} = (\preceq, Tr)$ *where: (a)* $Tr = (S, \rightarrow)$ *is a finitely-branching transition system, and (b)* $\preceq$ *is a* WQO *over* S *that is compatible with* $\rightarrow$*; that is: whenever* $s_1 \preceq s_2$ *and* $s_1 \rightarrow s'_1$ *then there is* s'_2 *such that* $s_2 \rightarrow s'_2$ *and* $s'_1 \preceq s'_2$*.*

Otherwise said, a WSTS is a finitely-branching transition system equipped with a WQO that is a simulation relation. Let Tr be a transition system equipped with a QO $\preceq$. Let $I \subseteq S$ be a set of states. We let the *upward closure* of I, written $\uparrow I$, be $\{s \in S \,|\, s' \preceq s$ for some $s' \in I\}$. The set $\uparrow \{s\}$ will be abbreviated as $\uparrow s$. A *basis* of (an upward-closed) set $Y \subseteq S$ is a set I such that $Y = \uparrow I$. We let the *immediate predecessors* of I, $\mathrm{Pred}(I)$, be the set $\{s \in S \,|\, s \rightarrow s'$ for some $s' \in I\}$ and the set of *predecessors* of I, $\mathrm{Pred}^*(I)$, be $\{s \in S \,|\, s \rightarrow^* s'$ for some $s' \in I\}$. We say $\mathcal{W}$ has an *(effective) pred-basis* if there is a (computable) function $\mathrm{pb}(\cdot) : S \rightarrow 2^S$ such that for each $s \in S$, $\mathrm{pb}(s)$ is a finite basis of $\uparrow \mathrm{Pred}(\uparrow s)$.

Proposition 4 ([8]). *Let* $\mathcal{W}$ *be a* WSTS *such that: (a)* $\preceq$ *is decidable, and (b)* $\mathcal{W}$ *has an effective pred-basis. Then there is a computable function that, for any finite* $I \subseteq S$*, returns a finite basis of* $\mathrm{Pred}^*(\uparrow I)$*.*

The above proposition entails decidability of a number of reachability-related problems in WSTS's (see [8]). Indeed, saying that the set I is reachable from a given state s is equivalent to saying that $s \in \mathrm{Pred}^*(\uparrow I)$: this can be decided, if one has at hand a finite basis B for $\mathrm{Pred}^*(\uparrow I)$, by just checking whether $s \geq s'$ for some $s' \in B$.

We will also rely upon some definitions and results on trees. Let L be a set. We define *ordered forests* $\mathcal{F}, \mathcal{G}, \ldots$ with labels in L (from now on, simply *forests*) to be the set of objects inductively defined as follows: (i) the empty sequence ϵ is a forest; (ii) if $\mathcal{F}_1, \ldots, \mathcal{F}_k$ are forests ($k \geq 0$) then the sequence $(a_1,\mathcal{F}_1)\cdots(a_k,\mathcal{F}_k)$, with $a_i \in L$, is a forest with roots $a_1, \ldots, a_k$. A forest of the form $(a,\mathcal{F})$ is called an *(ordered, rooted) tree*. A tree of the form (a,ϵ) is called a *leaf*. The multiset of leaves occurring in $\mathcal{F}$ is denoted by $\mathrm{L}(\mathcal{F})$, while the corresponding set is denoted $\mathrm{l}(\mathcal{F})$. The *height* of a forest $\mathcal{F}$, written $\mathrm{h}(\mathcal{F})$, is defined as the maximal length of a path from a root to a leaf, defined as expected; the height of a leaf is 0. We will often use the familiar pictorial representation of trees and forests. The following theorem provides us with a WQO on forests, hence on trees, called *rooted tree embedding*. One can think of this WQO as saying that $\mathcal{F}_1 \preceq \mathcal{F}_2$ if $\mathcal{F}_1$ can be mapped into a sub-forest of $\mathcal{F}_2$, provided that the mapping respects the roots of $\mathcal{F}_1$. The proof of the theorem can be given relying on a result on WQO on sequences due to Higman [9] (see [4] for a similar proof); or even generalizing the Kruskal tree theorem [12] to forests, again via Higman's lemma.

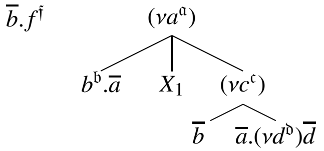
Theorem 4 (rooted tree embedding). *Let $\mathfrak{F}$ be the set of all forests with labels in a certain nonempty set. Consider the following* QO $\preceq$ *over $\mathfrak{F}$: $(a_1,\mathcal{F}_1)\cdots(a_k,\mathcal{F}_k) \preceq (b_1,\mathcal{G}_1)\cdots(b_h,\mathcal{G}_h)$ iff there are distinct indices $1 \leq i_1 < \cdots < i_k \leq h$ s.t. for each j, $1 \leq j \leq k$, $a_j = b_{i_j}$ and $\mathcal{F}_i \preceq \mathcal{G}_{i_j}$. Let $\mathfrak{G} \subseteq \mathfrak{F}$ be such that: there is a finite bound on the height of the forests in $\mathfrak{G}$ and there is a finite L s.t. the labels of all forests in $\mathfrak{G}$ are included in L. Then $\preceq$ is a* WQO *on $\mathfrak{G}$.*

A WSTS *for behavioural types.* Let $(X_i)_{i \geq 1}$ be an infinite sequence of *variables* disjoint from $\mathcal{N}$ and consider the grammar of types in Section 3, augmented with the clause $T ::= X$, where X ranges over variables. Let $\mathfrak{T}$ be the set of terms generated by this grammar – by "term" we mean here a proper term, *not* an alpha-equivalence class of terms – where each variable occurs at most once in a term and only in the scope of restrictions or parallel compositions. E.g. $(\nu a^{\mathfrak{a}})(X_1|a^{\mathfrak{a}}.\overline{b}.\overline{c})|X_2$ is in $\mathfrak{T}$, while $\overline{a}.X_1$ is not. In other words, we are considering open terms representing *static contexts*, with the variables X_i acting as the "holes". We let C range over $\mathfrak{T}$, reserving the letters S,T for the subset of closed terms (types) and will sometimes write $C[\tilde{X}]$ to indicate that C's variables are *exactly* $\tilde{X} = (X_{i_1}, \ldots, X_{i_k})$. In this case, taken $\tilde{T} = (T_1, \ldots, T_k)$, we will denote by $C[\tilde{T}]$ the term obtained by textually replacing each X_{i_j} with T_j in $C[\tilde{X}]$.

Each term C can be seen as a forest $\mathcal{F}_C$, with restrictions $(\nu a^{\mathfrak{a}})$ as internal labels and either guarded summations/replications G or variables X_i as leaves, and parallel composition $|$ interpreted as concatenation[1], as shown in the following example.

Example 3. Consider the term $C = \overline{b}.f^{\mathfrak{f}}\,|\,(\nu a^{\mathfrak{a}})(b^{\mathfrak{b}}.\overline{a}|X_1\,|\,(\nu c^{\mathfrak{c}})(\overline{b}|\overline{a}.(\nu d^{\mathfrak{d}})\overline{d}))$. The forest $\mathcal{F}_C$ associated to C is depicted below.

[1] More formally, each C is mapped to a forest $\mathcal{F}_C$ as follows: $\mathcal{F}_{X_i} = (X_i,\epsilon)$, $\mathcal{F}_G = (G,\epsilon)$, $\mathcal{F}_{T|S} = \mathcal{F}_T \cdot \mathcal{F}_S$ and $\mathcal{F}_{(\nu a^{\mathfrak{a}})T} = ((\nu a^{\mathfrak{a}}),\mathcal{F}_T)$.


$\overline{b}.f^{\mathfrak{f}}$
$(\nu a^{\mathfrak{a}})$
$b^{\mathfrak{b}}.\overline{a}$
X_1
$(\nu c^{\mathfrak{c}})$
$\overline{b}$
$\overline{a}.(\nu d^{\mathfrak{b}})\overline{d}$

Via this correspondence, we can identify terms with forests, and in what follows we shall not notationally distinguish between the two. In the following we will sometimes use a ground version of the function L(·), written GL(·), returning the multiset of *ground*, i.e. non-variables, leaves of a term. In the example above: $\mathrm{L}(C) = \{\!|\, \overline{b}.f^{\mathfrak{f}},\ b^{\mathfrak{b}}.\overline{a},\ X_1,\ \overline{b},\ \overline{a}.(\nu d^{\mathfrak{b}})\overline{d}\,|\!\}$ and $\mathrm{GL}(C) = \{\!|\, \overline{b}.f^{\mathfrak{f}},\ b^{\mathfrak{b}}.\overline{a},\ \overline{b},\ \overline{a}.(\nu d^{\mathfrak{b}})\overline{d}\,|\!\}$. The QO defined in the statement of Theorem 4 is also inherited by $\mathfrak{T}$, that is, we can set: $C \le C'$ iff $\mathcal{F}_C \le \mathcal{F}_{C'}$. To make this a *well* QO, we have to restrict ourselves to some subset of $\mathfrak{T}$ with bounded height and set of labels. This will be obtained by tailoring out of $\mathfrak{T}$ a superset of all terms that are reachable from a given initial closed term T. To this purpose, we introduce a few more additional notations directly on terms. Given a C, let us write dp(C) for the maximal *nesting depth of restrictions* in C, defined thus (max over an empty set yields 0):

$$\mathrm{dp}(X_i) = 0 \quad \mathrm{dp}(\textstyle\sum_{i\in I}\mu_i.C_i) = \max_{i\in I}\mathrm{dp}(C_i) \quad \mathrm{dp}(!a^{\mathfrak{a}}.C) = \mathrm{dp}(C)$$
$$\mathrm{dp}(C_1|C_2) = \max\{\mathrm{dp}(C_1),\mathrm{dp}(C_2)\} \quad \mathrm{dp}((\nu a^{\mathfrak{a}})C) = 1 + \mathrm{dp}(C)\,.$$

In the example above, dp(C) = 3. We denote by sub(C) the set of variables, summations and replications that occur as subterms of C: this is of course a finite set. Finally, we denote by res(C) the set of restrictions $(\nu a^{\mathfrak{a}})$ occurring in C. The set of terms we are interested in is defined below.

Definition 4 **($\mathfrak{T}_T[\tilde{X}]$).** *Fix a type T and a set of variables $\tilde{X} = (X_{j_1}, ..., X_{j_k})$, then*

$$\mathfrak{T}_T[\tilde{X}] \triangleq \{\, C \in \mathfrak{T} \,\big|\, \mathrm{l}(C) \subseteq \mathrm{sub}(T) \cup \tilde{X},\ \mathrm{res}(C) \subseteq \mathrm{res}(T),\ \mathrm{dp}(C) \le \mathrm{dp}(T)\,\}\,.$$

In the following, we abbreviate $\mathfrak{T}_T[\tilde{X}]$ as $\mathfrak{T}_T$ when $\tilde{X} = \epsilon$. Consider now the rooted-tree embedding $\le$ described above, we have the following result.

Proposition 5. *For any T and $\tilde{X}$, the relation $\le$ is a* WQO *over $\mathfrak{T}_T[\tilde{X}]$.*

Proof. Terms in $\mathfrak{T}_T[\tilde{X}]$, by definition, have bounded height: indeed, for any $C \in \mathfrak{T}_T[\tilde{X}]$, we have $\mathrm{h}(C) \le \mathrm{dp}(C) \le \mathrm{dp}(T)$. Moreover, they are built using a finite set of labels: $\tilde{X} \cup \mathrm{res}(T) \cup \mathrm{sub}(T)$. Theorem 4 ensures then that $\le$ is a WQO over $\mathfrak{T}_T[\tilde{X}]$.

We want to show now that $\mathfrak{T}_T$ can be endowed with WSTS structure. In what follows, we shall consider the traditional CCS transition relation over closed terms, denoted here $\overset{\mu}{\mapsto}$; in particular, we shall write $\overset{\tau}{\mapsto}$ as $\mapsto$. The relation $\mapsto$ is preferable to $\rightarrow$ in the present context, because it avoids alpha-equivalence, structural congruence and is finitely branching for the considered fragment. In Section 6, we shall argue that $\mapsto$ is equivalent to $\rightarrow$ for the purpose of defining the satisfaction relation $S \models \phi$. The set $\mathfrak{T}_T$ of closed terms enjoys the following crucial properties, which can be easily inferred by induction on the structure of the term. Note in particular that, by the second property, the restriction nesting depth of any term is *not* increased by $\overset{\mu}{\mapsto}$. This is a crucial property that does not hold in the pi-calculus. E.g. (type annotations omitted):

$$(\nu b_1)\overline{a}\langle b_1\rangle \,|\, !a(y).(\nu b_2)(y.b_2 \,|\, \overline{c}\langle b_2\rangle) \,|\, !c(x).\overline{a}\langle x\rangle \rightarrow^*$$
$$(\nu b_1)(\nu b_2)(b_1.b_2|(\nu b_3)(b_2.b_3|\cdots(\nu b_{n+1})(b_n.b_{n+1}|\overline{c}\langle b_{n+1}\rangle)\cdots))|!a(y).(\nu b_2)(y.b_2|\overline{c}\langle b_2\rangle)|!c(x).\overline{a}\langle x\rangle.$$

Proposition 6. *(1) For any $S \in \mathfrak{T}_T$ and S', $S \overset{\mu}{\mapsto} S'$ implies that $S' \in \mathfrak{T}_T$. (2) The relation $\preceq$ is a simulation relation over $\mathfrak{T}_T$. As a consequence: (3) For any T, let Tr be the transition system $(\mathfrak{T}_T, \overset{\tau}{\mapsto})$. Then $\mathcal{W}_T \triangleq (\preceq, Tr)$ is a* WSTS.

Concerning the decidability issues, we note that: (a) the WQO $\preceq$ is decidable, indeed its very inductive definition yields a decision algorithm; (b) the transition relation $\overset{\mu}{\mapsto}$ is decidable for the fragment of CCS that corresponds to the language of types.

6 Decidability

Decidability of a fragment of Shallow Logic relies on applying Proposition 4 to $\mathcal{W}_T$. The WQO $\preceq$ has already seen to be decidable. In order to be able to apply this proposition, we have to fulfill obligation (b), that is, show that $\mathcal{W}_T$ has an effective pred-basis. Moreover, we have to show that each denotation $[[\phi]]$ can be presented via an effectively computable finite basis playing the role of "I" in the proposition.

Pred-basis. Informally, the pred basis function, $\mathrm{pb}_T(S)$, works in two steps. First, all decompositions of S as $S = C[\tilde{U}]$, with $|\tilde{U}| = 0, 1$ or 2, are considered – there are finitely many of them. Then, out of each C, all contexts C' are built that have the same ground leaves as C, but possibly more holes, up to 2. Again, there are finitely many such contexts. The contexts C' are then filled with ground leaves, in such a way that the resulting terms posses a reduction to S, up to $\succeq$. In what follows, we shall also admit as a possible context C the 0-hole forest ϵ, which gives rise only to the decomposition $S = \epsilon[S]$.

Definition 5 (pred-basis). *Let T be a type, $S \in \mathfrak{T}_T$ and C, C' range over $\mathfrak{T}_T[X_1, X_2]$.*

$$\mathrm{pb}_T(S) \triangleq \bigcup_{S=C[\tilde{U}]} \{C'[\tilde{G}] \in \mathfrak{T}_T \mid C' \succeq C, \mathrm{GL}(C') = G\mathrm{L}(C), \tilde{G} \subseteq \mathrm{sub}(T), C'[\tilde{G}] \mapsto \succeq S\}$$

The construction of $\mathrm{pb}_T(S)$ is effective. In particular, given C, there are finitely many ways of adding one or two holes to C, resulting into a $C' \succeq C$, and they can all be tried in turn. In what follows we let $\mathrm{Pred}_T(\cdot)$ stand for $\mathrm{Pred}(\cdot) \cap \mathfrak{T}_T$.

Theorem 5. *Suppose $T \in \mathcal{T}$. Then for any $S \in \mathfrak{T}_T$, $\uparrow \mathrm{pb}_T(S) = \uparrow \mathrm{Pred}_T(\uparrow S)$. Moreover, $\mathrm{pb}_T(\cdot)$ is effective.*

Proof. (Outline) Effectiveness has already been discussed. Moreover, by construction, $\uparrow \mathrm{pb}_T(S) \subseteq \uparrow \mathrm{Pred}_T(\uparrow S)$. Let us examine the other inclusion. Suppose first $V \mapsto \succeq S$, we show that there is $U \in \mathrm{pb}_T(S)$ s.t. $V \succeq U$: this will be sufficient to accommodate also the most general case $V \succeq \mapsto \succeq S$, since $\mathcal{W}_T$ is a WSTS. Assume that the reduction in V originates from two communicating prefixes (the τ-prefix case is easier). That is, assume $V = C[G_1, G_2] \mapsto C[S_1, S_2] \succeq S$. It is then easy to prove that $S = C''[\tilde{S}']$, with $C \succeq C''$ and $(S_1, S_2) \succeq \tilde{S}'$. It is possible to build out of C'' a 2-holes context $C' \in \mathfrak{T}_T[X_1, X_2]$ s.t. $C \succeq C' \succeq C''$. Take $U = C'[G_1, G_2]$.

We can extend pb_T to finite sets $I \subseteq \mathfrak{T}_T$, by setting $\mathrm{pb}_T(I) \triangleq \cup_{S \in I} \mathrm{pb}_T(S)$. By doing so, we obtain the following corollary, which says that $\mathcal{W}_T$ has an effective pred-basis.

Corollary 1. *There is a computable function $\mathrm{pb}_T(\cdot)$ such that, for any finite $I \subseteq \mathfrak{T}_T$, $\uparrow \mathrm{pb}_T(I) = \uparrow \mathrm{Pred}_T(\uparrow I)$.*

Remark 1. Consider the labelled version of the reduction relation, $\overset{\langle\lambda\rangle}{\longmapsto}$, $\lambda ::= a|\epsilon$. For any fixed label $\langle a\rangle$, Corollary 1 still holds if considering the transition system given by $\overset{\langle a\rangle}{\longmapsto}$, rather than $\mapsto$. We shall name $\mathrm{pb}^{\langle a\rangle}_{T}(\cdot)$ the corresponding pred-basis function.

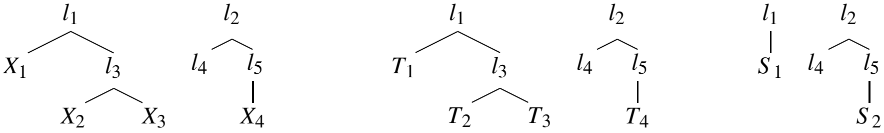
Applying Proposition 4, we get the result we were after.

Corollary 2. *There exists a computable function* $\mathrm{pb}^{*}_{T}(\cdot)$ *such that, for any finite set* $I \subseteq \mathfrak{T}_T$, $\mathrm{pb}^{*}_{T}(I)$ *is a finite basis of* $\mathrm{Pred}^{*}_{T}(\uparrow I)$.

Finite bases for plain formulae. Our first task is showing that, for certain formulae ϕ, the satisfaction relation $S \models \phi$ can be defined relying solely on $\mapsto$ and on context decomposition, in particular, with no reference to structural congruence and $\rightarrow$. In the proposition below, we show that this is indeed possible for *plain* formulae.

Definition 6 (plain formulae). *We say a formula* ϕ *is* plain *if it does not contain* $\diamond^*$ *underneath* H^*.

Let us say a context C is *pure* if $\mathrm{l}(C) \subseteq (X_i)_{i\geq 1}$; we let D range over pure contexts; e.g. $D = (\nu a)(X_1|X_2)|X_3$ is pure. Given a context $C[\tilde{X}, \tilde{Y}]$ and two sequences $\tilde{G}$, $\tilde{F}$ s.t. $|\tilde{X}| = |\tilde{G}|$ and $|\tilde{Y}| = |\tilde{F}|$, we say *$C$ links $\tilde{G}$ and $\tilde{F}$* if there are an internal node (νa^{α}) of C seen as a forest, $X_i \in \tilde{X}$ and $Y_j \in \tilde{Y}$ such that both X_i and Y_j are in the scope of this node, and $a \in \mathrm{fn}(G_i) \cap \mathrm{fn}(F_j)$. Given a sequence $\tilde{G}$, we denote by $\prod \tilde{G}$ the parallel composition of the terms in $\tilde{G}$, in some arbitrary order. Given a term $C[\tilde{T}]$ and $\tilde{S} \preceq \tilde{T}$, fix any injection $f : \{1, \ldots, k\} \rightarrow \{1, \ldots, |\tilde{T}|\}$ $(k = |\tilde{S}|)$ such that $S_j \preceq T_{f(j)}$, for $1 \leq f(1) < \cdots < f(k) \leq |\tilde{T}|$. We write $C[\tilde{S} \triangleleft_f \tilde{T}]$ for the closed term obtained from $C[^{S_j}/X_{f(j)}]_{j=1,\cdots,k}$ by pruning all sub-trees having only variables as leaves. In the following we will write $C[\tilde{S} \triangleleft \tilde{T}]$ for $C[\tilde{S} \triangleleft_f \tilde{T}]$, when f is the identity. As an example, take $\tilde{T} = T_1, T_2, T_3, T_4$, $\tilde{S} = S_1, S_2$ and suppose $f(1) = 1$ and $f(2) = 4$. C, $C[\tilde{T}]$ and $C[\tilde{S} \triangleleft_f \tilde{T}]$ are depicted below.



Proposition 7. *Assume* $S \in \mathfrak{T}_T$, ϕ *plain and monotone and* $\mathrm{bn}(T) \cap \mathrm{n}(\phi) = \emptyset$. *Then we have the following equivalences, where* $\tilde{G}, \tilde{G}_1, \tilde{G}_2$ *are assumed to be included in* $\mathrm{sub}(T)$.

$$\begin{array}{ll}
S \models \langle a\rangle\phi & \mathit{iff}\ \exists U : S \overset{\langle a\rangle}{\longmapsto} U \mathit{\ and\ } U \models \phi \qquad\qquad S \models \diamond^*\phi \ \mathit{iff}\ \exists U : S \mapsto^* U \mathit{\ and\ } U \models \phi \\
S \models a & \mathit{iff}\ \exists D, \tilde{G}, U : S = D[\tilde{G}] \mathit{\ and\ for\ some\ } G \in \tilde{G} : G = !a^{\alpha}.U \mathit{\ or\ } a^{\alpha}.U \mathit{\ is\ a\ summand\ of\ } G \\
S \models \overline{a} & \mathit{iff}\ \exists D, \tilde{G}, U : S = D[\tilde{G}] \mathit{\ and\ for\ some\ } G \in \tilde{G} : \overline{a}.U \mathit{\ is\ a\ summand\ of\ } G \\
S \models \mathrm{H}^*\phi & \mathit{iff}\ \exists D, \tilde{G} : S = D[\tilde{G}] \mathit{\ and\ } \prod \tilde{G} \models \phi \\
S \models \phi_1|\phi_2 & \mathit{iff}\ \exists D, \tilde{G}_1, \tilde{G}_2 : S = D[\tilde{G}_1, \tilde{G}_2],\ D \mathit{\ not\ linking\ } \tilde{G}_1 \mathit{\ and\ } \tilde{G}_2,\ D[\tilde{G}_i \triangleleft \tilde{G}_1, \tilde{G}_2] \models \phi_i \, (i = 1, 2)
\end{array}$$

As discussed at the beginning of this section, in order to take advantage of Corollary 2, we have to show that each set $[[\phi]]$, or, more accurately, each set $[[\phi]]_T \triangleq [[\phi]] \cap \mathfrak{T}_T$, can be presented via an effectively computable finite basis in $\mathcal{W}_T$. We define this basis below, by induction on the structure of ϕ: the $\diamond^*$ and $\langle a\rangle$ cases take advantage of the pred-basis functions defined in the last paragraph, the other cases basically follow the corresponding cases of the previous proposition or, in the case of $\vee$ and $\mathbf{T}$, the expected boolean interpretation. The only exception to this scheme is the $\wedge$ connective, which is nontrivial and will be commented below. Some more terminology first. Given a set $I \subseteq \mathfrak{T}_T$, we denote by minimal($I$) the set of minimal elements in I, w.r.t. the wqo $\preceq$. For any ordered sequence $\tilde{G}$, we denote by $\langle\tilde{G}\rangle$ the multiset obtained if ignoring order.

Definition 7 (finite basis). *Let T be a type and ϕ be a plain and monotone formula, such that* $\mathrm{bn}(T) \cap \mathrm{n}(\phi) = \emptyset$. *The* finite basis $\mathrm{Fb}_T(\phi)$ *is inductively defined below, where $G, \tilde{G}, \tilde{G}_1$ and $\tilde{G}_2$ are assumed to be included in* $\mathrm{sub}(T)$.

$$\begin{aligned}
\mathrm{Fb}_T(a) &\triangleq \{D[G] \in \mathfrak{T}_T \;\big|\; G = !a^{\mathfrak{a}}.U \textit{ or } a^{\mathfrak{a}}.U \textit{ is a summand of } G\textit{, for some } U\}\\
\mathrm{Fb}_T(\overline{a}) &\triangleq \{D[G] \in \mathfrak{T}_T \;\big|\; \overline{a}.U \textit{ is a summand of } G\textit{, for some } U\}\\
\mathrm{Fb}_T(\phi_1|\phi_2) &\triangleq \textstyle\bigcup_{S_1 \in \mathrm{Fb}_T(\phi_1), S_2 \in \mathrm{Fb}_T(\phi_2)} \{D[\tilde{G}_1, \tilde{G}_2] \in \mathfrak{T}_T \;\big|\; \textit{for } i = 1,2 : \langle\tilde{G}_i\rangle = \mathrm{L}(S_i)\textit{, } D \textit{ does not}\\
&\qquad\qquad \textit{link } \tilde{G}_1 \textit{ and } \tilde{G}_2 \textit{ and } D[\tilde{G}_i \blacktriangleleft \tilde{G}_1, \tilde{G}_2] \succeq S_i\}\\
\mathrm{Fb}_T(\mathrm{H}^*\phi) &\triangleq \textstyle\bigcup_{S \in \mathrm{Fb}_T(\phi)} \{D[\tilde{G}] \in \mathfrak{T}_T \;\big|\; \langle\tilde{G}\rangle = \mathrm{L}(S)\} \qquad \mathrm{Fb}_T(\mathbf{T}) \triangleq \{D[G] \in \mathfrak{T}_T \;\big|\; G \in \mathrm{sub}(T)\}\\
\mathrm{Fb}_T(\phi_1 \vee \phi_2) &\triangleq \mathrm{Fb}_T(\phi_1) \cup \mathrm{Fb}_T(\phi_2) \qquad \mathrm{Fb}_T(\langle a\rangle\phi) \triangleq \mathrm{pb}_T^{\langle a\rangle}(\mathrm{Fb}_T(\phi))\\
\mathrm{Fb}_T(\phi_1 \wedge \phi_2) &\triangleq \mathrm{minimal}([\![\phi_1]\!] \cap [\![\phi_2]\!]) \qquad \mathrm{Fb}_T(\diamond^*\phi) \triangleq \mathrm{pb}_T^{*}(\mathrm{Fb}_T(\phi))
\end{aligned}$$

Note that minimal($[\![\phi_1]\!] \cap [\![\phi_2]\!]$) is finite: if not, one would find an infinite sequence of pairwise incomparable elements, thus violating the condition of wqo.

Theorem 6. *Consider T and ϕ like in Definition 7. Then* $\mathrm{Fb}_T(\phi)$ *is a finite basis for* $[\![\phi]\!]_T$*, that is* $\uparrow \mathrm{Fb}_T(\phi) = [\![\phi]\!]_T$. *Moreover,* $\mathrm{Fb}_T(\cdot)$ *is computable.*

Proof. (Outline) The first part of the statement is quite easy, indeed one inclusion, $\uparrow \mathrm{Fb}_T(\phi) \subseteq [\![\phi]\!]_T$, is valid by construction, while the opposite direction is proved by induction on ϕ, relying on the characterization of $\models$ provided by Proposition 7 for the spatial and dynamic connectives, the boolean ones being trivial to handle. Proving that $\mathrm{Fb}_T(\cdot)$ is computable is more difficult, because of the clause for conjunction. This is accommodated by introducing an effective operator $\|$ that over-approximates the "minimal" operator: $\mathrm{Fb}_T(\phi_1) \parallel \mathrm{Fb}_T(\phi_2) \supseteq \mathrm{minimal}([\![\phi_1]\!]_T \cap [\![\phi_2]\!]_T)$. The operator $\|$ produces a finite set of terms by appropriately merging terms, seen as forests, drawn from $\mathrm{Fb}_T(\phi_1)$ and $\mathrm{Fb}_T(\phi_2)$. We refer the reader to [2] for the details.

By virtue of the above theorem, we can decide if $S \models \phi$, with $S \in \mathfrak{T}_T$, by checking if there is $U \in \mathrm{Fb}_T(\phi)$ s.t. $S \succeq U$: since $\preceq$ is decidable, this can be effectively carried out, and we obtain Corollary 3. Finally, Corollary 4 is a consequence of Proposition 2.

Corollary 3 (decidability on types). *Let ϕ be plain and monotone. It is decidable whether $T \models \phi$. Hence, decidability also holds for ϕ plain and anti-monotone.*

Corollary 4 (decidability on pi-processes). *Let Γ be a context. Given a Γ-well-typed P and ϕ plain and (anti-)monotone, it is decidable whether $P \models \phi$.*

7 Conclusion and Related Work

We have proven the decidability of a fragment of Spatial Logic that includes interesting safety properties for a class of infinite-control pi-processes. The proof relies heavily on both behavioural type systems [10,7,1] and well-structured transition system techniques [8]. Implementation issues are not in the focus of this paper. Whether a practical algorithm may be obtained or not from the theoretical discussion presented here is an interesting topic, that is left for future work.

Our proof of decidability generalizes the result in [4] that "weak" barbs $\diamond^* a$ are decidable in ccs with replication. Variations and strengthening of these results have recently been obtained by Valencia et al. [14]. It is worth to notice that weak barbs are *not* decidable in the pi-calculus, [3]. On the other hand, our results show that they become decidable when restricting to well-typed pi-processes.

Also related to our approach is [5], where Caires proves that model-checking Spatial Logic formulae for bounded pi-calculus processes, and in particular finite-control processes, is decidable. Note that the class of processes we have considered here properly includes bounded processes.

Acknowledgments. We wish to thank Luis Caires, Roland Meyer and Gianluigi Zavattaro for very stimulating discussions on the topics of the paper.

References

1. Acciai, L., Boreale, M.: Spatial and behavioral types in the pi-calculus. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 372–386. Springer, Heidelberg (2008) (Full version submitted, 2009)
2. Acciai, L., Boreale, M.: Deciding safety properties in infinite-state pi-calculus via behavioural types. Extended version, http://gdn.dsi.unifi.it/~acciai/papers/decFull.pdf
3. Amadio, R., Meyssonnier, C.: On decidability of the control reachability problem in the asynchronous pi-calculus. Nordic Journal of Computing 9(2), 70–101 (2002)
4. Busi, N., Gabbrielli, M., Zavattaro, G.: Comparing recursion, replication, and iteration in process calculi. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 307–319. Springer, Heidelberg (2004)
5. Caires, L.: Behavioural and Spatial Observations in a Logic for the pi-Calculus. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 72–89. Springer, Heidelberg (2004)
6. Caires, L., Cardelli, L.: A spatial logic for concurrency (part I). Inf. Comput. 186(2), 194–235 (2003)
7. Chaki, S., Rajamani, S.K., Rehof, J.: Types as models: model checking message-passing programs. In: Proc. of POPL 2002, pp. 45–57 (2002)
8. Finkel, A., Schnoebelen, P.: Well-Structured Transition Systems Everywhere! Theoretical Computer Science 256(1-2), 63–92 (2001)
9. Higman, G.: Ordering by divisibility in abstract algebras. Proc. London Math. Soc. 2, 326–366 (1952)
10. Igarashi, A., Kobayashi, N.: A generic type system for the Pi-calculus. Theoretical Computer Science 311(1-3), 121–163 (2004)
11. Kobayashi, N., Suto, T.: Undecidability of 2-Label BPP Equivalences and behavioural Type Systems for the Pi-Calculus. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 740–751. Springer, Heidelberg (2007)
12. Kruskal, J.B.: Well-quasi-ordering, the tree theorem, and Vázsonyi's conjecture. Trans. American Math. Soc. 95, 210–225 (1960)
13. Milner, R.: The polyadic π-calculus: a tutorial. In: Logic and Algebra of Spec., pp. 203–246 (1993)
14. Valencia, F., Aranda, J., Versari, C.: On the Expressive Power of Restriction and Priorities in CCS with Replication. In: de Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 242–256. Springer, Heidelberg (2009)

When Are Timed Automata Determinizable?

Christel Baier[1], Nathalie Bertrand[2], Patricia Bouyer[3], and Thomas Brihaye[4]

[1] Technische Universität Dresden, Germany
[2] INRIA Rennes Bretagne Atlantique, France
[3] LSV, CNRS & ENS Cachan, France
[4] Université de Mons, Belgium

Abstract. In this paper, we propose an abstract procedure which, given a timed automaton, produces a language-equivalent deterministic infinite timed tree. We prove that under a certain boundedness condition, the infinite timed tree can be reduced into a classical *deterministic* timed automaton. The boundedness condition is satisfied by several subclasses of timed automata, some of them were known to be determinizable (event-clock timed automata, automata with integer resets), but some others were not. We prove for instance that strongly non-Zeno timed automata can be determinized. As a corollary of those constructions, we get for those classes the decidability of the universality and of the inclusion problems, and compute their complexities (the inclusion problem is for instance EXPSPACE-complete for strongly non-Zeno timed automata).

1 Introduction

Timed automata have been proposed by Alur and Dill in the early 90s as a model for real-time systems [2]. A timed automaton is a finite automaton which can manipulate real-valued variables called clocks, that evolve synchronously with the time, can be tested and reset to zero. One of the fundamental properties of this model is that, although the set of configurations is in general infinite, checking reachability properties is decidable. From a language-theoretic point of view, this means that checking emptiness of the timed language accepted by a timed automaton can be decided (and is a PSPACE-complete problem). The proof relies on the construction of the so-called region automaton, which finitely abstracts behaviours of a timed automaton. Since then, its appropriateness as a model for the verification of real-time systems has been confirmed, with the development of verification algorithms and dedicated tools.

There are however two weaknesses to that model: a timed automaton cannot be determinized, and inclusion (and universality) checking is undecidable [2], except for deterministic timed automata. This basically forbids the use of timed automata as a specification language. Understanding and coping with these weaknesses have attracted lots of research, and, for instance, testing whether a timed automaton is determinizable has been proved undecidable [6]. Also, the undecidability of universality has been further investigated, and rather restricted classes of timed automata suffer from that undecidability result [1]. On

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 43–54, 2009.

the other hand, classes of timed automata have been exhibited, that either can be effectively determinized (for instance event-clock timed automata [3], or timed automata with integer resets [9]), or for which universality can be decided (for instance single-clock timed automata [7]).

In this paper, we describe a generic construction that is applicable to every timed automaton, and which, under certain conditions, yields a deterministic timed automaton, which is language-equivalent to the original timed automaton. The idea of the procedure is to unfold the timed automaton into a finitely-branching infinite tree that records the timing constraints that have to be satisfied using one clock per level of the tree (hence infinitely many clocks). When reading a finite timed word in that infinite tree, we may reach several nodes of the tree, but the timing information stored in the clocks is independent of the run in the tree. Thanks to this kind of *input-determinacy* property, we can determinize this infinite object, yielding another finitely-branching infinite tree. And, under a boundedness condition on the amount of timing information we need to store, we will be able to fold back the tree into a deterministic timed automaton. This boundedness condition is not a syntactical condition on the original timed automaton, but will be satisfied by large classes of timed automata: event-clock timed automata [3], timed automata with integer resets [9], and strongly non-Zeno timed automata. Furthermore, our construction yields automata of exponential-size in the first case, and doubly-exponential-size automata otherwise. In particular, our approach provides an EXPSPACE algorithm to check universality (and inclusion) for a large class of timed automata, and we prove that this complexity is tight. Our algorithm can easily be adapted into a PSPACE one, in the special case of event-clock timed automata, allowing to recover the known result of [3].

2 Timed Automata

Preliminaries. Given X a finite or infinite set of clocks and M a non-negative integer, a clock *valuation* over X bounded by M is a mapping $v : X \to \mathbb{T}_M$ where $\mathbb{T}_M = [0, M] \cup \{\bot\}$. We assume furthermore that $\bot > M$. The notation $\bot$ is for abstracting values of clocks that are above some fixed value M. This is rather non-standard (though used for instance in [8]) but it will be convenient in this paper. We note $\bar{0}$ the valuation that assigns 0 to all clocks. If v is a valuation over X and bounded by M, and $t \in \mathbb{R}_+$, then $v + t$ denotes the valuation which assigns to every clock $x \in X$ the value $v(x) + t$ if $v(x) + t \leq M$, and $\bot$ otherwise (in particular, if $v(x) = \bot$, then $(v + t)(x) = \bot$). For $Y \subseteq X$ we write $[Y \leftarrow 0]v$ for the valuation equal to v on $X \setminus Y$ and to $\bar{0}$ on Y, and $v_{|Y}$ for the valuation v restricted to clocks in Y. A(n M-bounded) *guard* (or *constraint*) over X is a finite conjunction of constraints of the form $x \sim c$ where $x \in X$, $c \in \mathbb{N} \cap [0, M]$ and $\sim \in \{<, \leq, =, \geq, >\}$. We denote by $\mathcal{G}_M(X)$ the set of M-bounded guards over X. Given a valuation v and a guard g we write $v \models g$ whenever v satisfies g.

A timed word over Σ is a finite sequence of pairs $(a_1, t_1)(a_2, t_2) \ldots (a_k, t_k)$ such that for every i, $a_i \in \Sigma$ and $(t_i)_{1 \leq i \leq k}$ is a nondecreasing sequence in $\mathbb{R}_+$.

Timed automata. A *timed automaton* is a tuple $\mathcal{A} = (L, \ell_0, L_{\mathsf{acc}}, X, M, E)$ such that: (*i*) L is a finite set of locations, (*ii*) $\ell_0 \in L$ is the initial location, (*iii*) $L_{\mathsf{acc}} \subseteq L$ is the set of final locations, (*iv*) X is a finite set of clocks, (*v*) $M \in \mathbb{N}$, and (*vi*) $E \subseteq L \times \mathcal{G}_M(X) \times \Sigma \times 2^X \times L$ is a finite set of edges. Constant M is called the maximal constant of $\mathcal{A}$.

The semantics of a timed automaton $\mathcal{A}$ is given as a timed transition system $\mathcal{T}_{\mathcal{A}} = (S, s_0, S_{\mathsf{acc}}, (\mathbb{R}_+ \times \Sigma), \rightarrow)$ with set of states $S = L \times \mathbb{T}_M^X$, initial state $s_0 = (\ell_0, \bar{0})$, set of accepting states $S_{\mathsf{acc}} = L_{\mathsf{acc}} \times \mathbb{T}_M^X$, and transition relation $\rightarrow \subseteq S \times (\mathbb{R}_+ \times \Sigma) \times S$ composed of moves of the form $(\ell, v) \xrightarrow{\tau,a} (\ell', v')$ whenever there exists an edge $(\ell, g, a, Y, \ell') \in E$ such that $v+\tau \models g$ and $v' = [Y \leftarrow 0](v+\tau)$.

A run ϱ of $\mathcal{A}$ is a finite sequence of moves, *i.e.*, $\varrho = s_0 \xrightarrow{\tau_1,a_1} s_1 \dots \xrightarrow{\tau_k,a_k} s_k$. It is said initial whenever $s_0 = (\ell_0, \bar{0})$. An initial run is accepting if it ends in an accepting location. The timed word $u = (a_1,t_1)(a_2,t_2)\dots(a_k,t_k)$ is said to be read on ϱ whenever $t_i = \sum_{j=1}^{i} \tau_j$ for every $1 \le i \le k$. We write $\mathcal{L}(\mathcal{A})$ for the set of timed words (or timed language) accepted by $\mathcal{A}$, that is the set of timed words u such that there exists an initial and accepting run ϱ which reads u.

A timed automaton $\mathcal{A}$ is *deterministic* whenever for every timed word u, there is at most one initial run which reads u. It is *strongly non-Zeno* whenever there exists $K \in \mathbb{N}$ such that for every run $\varrho = s_0 \xrightarrow{\tau_1,a_1} s_1 \dots \xrightarrow{\tau_k,a_k} s_k$ in $\mathcal{A}$, $k \ge K$ implies $\sum_{i=1}^{k} \tau_i \ge 1$. This condition is rather standard in timed automata [4].

Example 1. An example of timed automaton is depicted in Fig. 1. This automaton will be used as a running example throughout the paper in order to illustrate the different steps of our construction. This automaton is not deterministic and accepts the timed language $\{(a,t_1)(a,t_2)\cdots(a,t_{2n}) \mid n \ge 1,\ 0 < t_1 < t_2 < \cdots < t_{2n-1} \text{ and } t_{2n} - t_{2n-2} = 1\}$, with the convention that $t_0 = 0$. The timed word $(a,0.5)(a,1.6)(a,2.9)$ can be read on the initial run $(\ell_0, 0) \xrightarrow{0.5,a} (\ell_3, 0) \xrightarrow{1.1,a} (\ell_0, 0) \xrightarrow{1.3,a} (\ell_1, \bot)$ but is not accepted. The last configuration of the above run is $(\ell_1, \bot)$ because the value of clock x should be 1.3, but as it is larger than the maximal constant 1, we abstract the precise value into $\bot$.

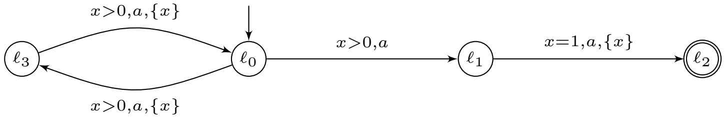


Fig. 1. A timed automaton $\mathcal{A}$

On timed bisimulations. A *strong timed (resp. time-abstract) simulation relation* between two timed transition systems $\mathcal{T}_i = (S_i, s_{i,0}, S_{i,\mathsf{acc}}, (\Sigma \cup \mathbb{R}_+), \rightarrow_i)$ for $i \in \{1,2\}$ is a relation $\mathfrak{R} \subseteq S_1 \times S_2$ such that if $s_1 \mathrel{\mathfrak{R}} s_2$ and $s_1 \xrightarrow{t_1,a} s_1'$ for some $t_1 \in \mathbb{R}_+$ and $a \in \Sigma$, then there exists $s_2' \in S_2$ (resp. $t_2 \in \mathbb{R}_+$ and $s_2' \in S_2$) such that $s_2 \xrightarrow{t_1,a} s_2'$ (resp. $s_2 \xrightarrow{t_2,a} s_2'$) and $s_1' \mathrel{\mathfrak{R}} s_2'$. A *strong timed (resp. time-abstract) bisimulation relation* between two timed transition $\mathcal{T}_i$ for $i \in \{1,2\}$ is a relation $\mathfrak{R} \subseteq S_1 \times S_2$ such that both $\mathfrak{R}$ and $\mathfrak{R}^{-1}$ are strong timed (resp. time-abstract) simulation relations. The above relations *preserve* initial (resp.

accepting) states whenever $s_{1,0} \mathfrak{R} s_{2,0}$ (resp. $s_1 \mathfrak{R} s_2$ and $s_i \in S_{i,\mathsf{acc}}$ implies $s_{3-i} \in S_{3-i,\mathsf{acc}}$). Note that the notion of strong timed bisimulation which preserves initial and accepting states is stronger than that of language equivalence.

The classical region construction. We let X be a finite set of clocks, and $M \in \mathbb{N}$. We define the equivalence relation $\equiv_{X,M}$ between valuations in $\mathbb{T}_M$ as follows: $v \equiv_{X,M} v'$ iff (i) for every clock $x \in X$, $v(x) \leq M$ iff $v'(x) \leq M$; (ii) for every clock $x \in X$, if $v(x) \leq M$, then $\lfloor v(x) \rfloor = \lfloor v'(x) \rfloor$, and (ii) for every pair of clocks $(x, y) \in X^2$ such that $v(x) \leq M$ and $v'(x) \leq M$, $\{v(x)\} \leq \{v(y)\}$ iff $\{v'(x)\} \leq \{v'(y)\}$. [1] The equivalence relation is called the *region equivalence* for the set of clocks X w.r.t. M, and an equivalence class is called a *region*. We note Reg_M^X for the set of such regions. A region r' is a time-successor of a region r if there is $v \in r$ and $t \in \mathbb{R}_+$ such that $v + t \in r'$. If v is a valuation, we will write $[v]$ for the region to which v belongs (when X and M are clear in the context).

It is a classical result [2] that given a timed automaton $\mathcal{A}$ with maximal constant M and set of clocks X, the relation $\mathfrak{R}_{X,M}$ between configurations of $\mathcal{A}$ defined by $(\ell, v) \mathfrak{R}_{X,M} (\ell, v')$ iff $v \equiv_{X,M} v'$ is a time-abstract bisimulation.

3 Some Transformations

In this section, we describe a general construction that aims at determinizing a timed automaton. We know however that not all timed automata can be determinized [2], and even that we cannot decide whether a timed automaton can be determinized [6]. We will thus give conditions that will ensure (i) that our procedure can be properly applied, and (ii) that the resulting timed automaton is deterministic and accepts the same timed language as the original automaton. We will then analyze the complexity of the procedure, and apply it to several subclasses of timed automata, some of which were known to be determinizable, some other were not known to be determinizable.

This construction consists in four steps: (i) an unfolding of the original automaton into an infinite timed tree, (ii) a region abstraction, (iii) a symbolic determinization, and (iv) a reduction of the number of clocks, allowing to fold the tree back into a timed automaton. These steps are described in the following subsections. Due to page limitation, we will give no formal definitions of the objects we build in our construction, and better illustrate the construction on the running example. All details can be found in the technical report [5].

3.1 Construction of an Equivalent Infinite Timed Tree

In this first step, we unfold the timed automaton $\mathcal{A}$ into a finitely-branching *infinite timed tree* $\mathcal{A}^\infty$ that has infinitely many clocks (one clock per level of the tree), we call $Z = \{z_0, z_1, \ldots\}$ this infinite set of clocks. The idea of this unfolding is to use a fresh clock reset at each level of the tree in order to record the timing constraints that have to be satisfied in $\mathcal{A}$. Each node n of $\mathcal{A}^\infty$ is

[1] Where $\lfloor \alpha \rfloor$ (resp. $\{\alpha\}$) denotes the integral (resp. fractional) part of α.

labelled by a pair $(\ell, \sigma) \in L \times Z^X$ where ℓ records the location of $\mathcal{A}$ that node n simulates and σ describes how the clocks of $\mathcal{A}$ are encoded using the clocks of $\mathcal{A}^\infty$ (if $\sigma(x) = z_i$, the value clock x would have in $\mathcal{A}$ is the current value of clock z_i). The advantage of this infinite timed tree is that it enjoys some *input-determinacy* property: when reading a finite timed word u in $\mathcal{A}^\infty$, there may be several runs in the tree that read u, but the timing information stored in the clocks is independent of the run in the tree (see Remark 4).

Example 2. Part of the infinite timed tree $\mathcal{A}^\infty$ associated with the timed automaton $\mathcal{A}$ of Fig. 1 is depicted in Fig. 2. Notice that a fresh clock is reset at each level; for instance z_2 is reset on all edges from level-1 to level-2 nodes (*i.e.* $n_1 \to n_3$ and $n_2 \to n_4$). The timed tree $\mathcal{A}^\infty$ corresponds to the unfolding of $\mathcal{A}$: the two branches starting from the node n_0 represent the possible choice in state ℓ_0 of $\mathcal{A}$; the same phenomenon also happens in n_4. The label of n_4 is (ℓ_0, z_2); it means that node n_4 represents the location ℓ_0 of $\mathcal{A}$ and that the value of clock x can be recovered from the current value of clock z_2. It is important to observe how the second component of the label evolves. First consider the edge $n_4 \to n_5$; it represents the transition from ℓ_0 to ℓ_1 in $\mathcal{A}$, which does not reset clock x; the reference for clock x is the same in n_5 as it is in n_4, that is why the label of n_5 is (ℓ_1, z_2). Now consider the edge $n_4 \to n_6$; it represents the transition from ℓ_0 to ℓ_3 in $\mathcal{A}$, which resets clock x; the reference for clock x thus becomes z_3, the clock which has just been reset, that is why the label of n_6 is (ℓ_3, z_3).

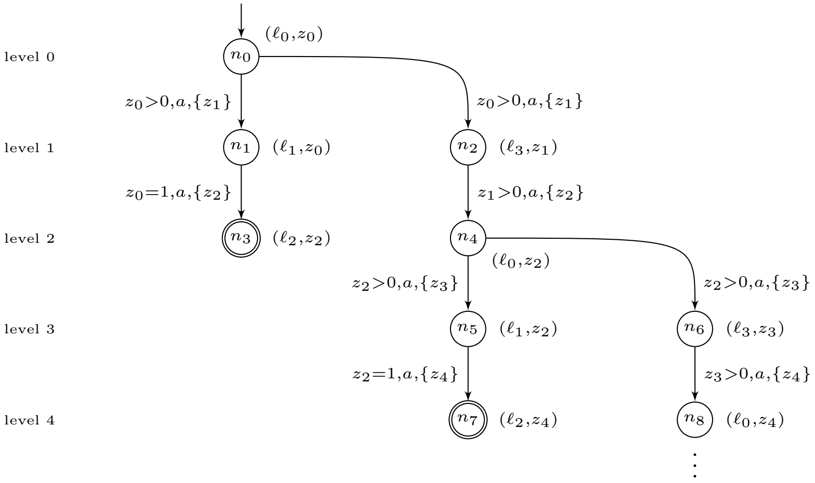


Fig. 2. The infinite timed tree $\mathcal{A}^\infty$ associated with the timed automaton $\mathcal{A}$ of Fig. 1

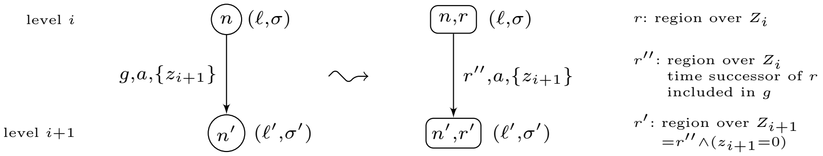
The correctness of this unfolding is stated in the follow lemma.

Lemma 3. *The relation $\mathfrak{R}_1$ between states of $\mathcal{A}$ and states of $\mathcal{A}^\infty$ defined by $(\ell, v \circ \sigma)\ \mathfrak{R}_1\ (n, v)$ if label$(n) = (\ell, \sigma)$ is a strong timed bisimulation.*

Remark 4. In $\mathcal{A}^\infty$, for every finite timed word u, there is a unique valuation $v_u \in \mathbb{T}^Z$ such that for every initial run ϱ in $\mathcal{A}^\infty$ that reads u, ϱ ends in some configuration (n, v_u) with $\mathit{level}(n) = |u|$. Indeed, if the timed word u is of the form $(a_1, t_1)...(a_{|u|}, t_{|u|})$, any initial run ϱ reading u necessarily ends in a configuration (n, v_u) where $\mathit{level}(n) = |u|$ and $v_u(z_j) = t_{|u|} - t_j$ for any $j \leq |u|$.

3.2 A Region Abstraction

In this second step, we extend in a natural way the classical region equivalence to the above infinite timed tree: at level i of the tree, only clocks in $Z_i = \{z_0, \cdots, z_i\}$ are relevant (all other clocks have not been used yet), we thus consider regions over that set of clocks. We use $R(\mathcal{A}^\infty)$ to denote this region abstraction, and we interpret it in a timed manner. We do not illustrate this transformation step on our running example, since $R(\mathcal{A}^\infty)$ is easily obtained from $\mathcal{A}^\infty$, but only depict the transformation on an edge, see below:



It is worth noting that, in $R(\mathcal{A}^\infty)$, any state reached after a transition is of the form $((n, r), v)$, where n is a node of $\mathcal{A}^\infty$ (of some level, say i), r is a region over Z_i, and v is a valuation over Z_i which belongs to r. It is not difficult to see that, as in the standard region construction in timed automata, two states $((n, r), v_1)$ and $((n, r), v_2)$ with $v_1, v_2 \in r$ are time-abstract bisimilar. Furthermore, $R(\mathcal{A}^\infty)$ will satisfy the same input-determinacy property as $\mathcal{A}^\infty$ (see Remark 4). The correctness of $R(\mathcal{A}^\infty)$ can then be stated as follows.

Lemma 5. *The relation $\mathfrak{R}_2$ between states of $\mathcal{A}^\infty$ and states of $R(\mathcal{A}^\infty)$ defined by $(n, v)\ \mathfrak{R}_2\ ((n, r), v)$ if $v \in r$ is a strong timed bisimulation.*

3.3 Symbolic Determinization

This third step is the crucial step of our construction. We will symbolically determinize the infinite timed tree $R(\mathcal{A}^\infty)$ using a rather standard subset construction, and we denote by $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ the resulting infinite tree. However there will be a subtlety in the subset construction: useless clocks will be forgotten 'on-the-fly'. More precisely, at each node, we only consider active clocks, *i.e.* clocks that appear in the label of the node (other clocks record values that do not impact on further behaviours of the system). The determinization is then performed on the 'symbolic' alphabet composed of regions over active clocks and actions, and thanks to the input-determinacy property of $R(\mathcal{A}^\infty)$, this symbolic determinization coincides with the determinization of the underlying timed transition system. Let us explain this crucial step on our running example.

Example 6. The construction of $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ is illustrated on Fig. 3. The determinization is performed using a classical subset construction. For example starting from node n_0, both n_1 and n_2 can be reached *via* a transition with guard $0 < z_0 < 1$. This is reflected in the leftmost $\{n_1, n_2\}$-node at the first level. It is also important to understand the meaning of active clocks. In $\mathcal{A}^\infty$, the only active clock in node n_4 is z_2. Therefore, guards on transitions leaving the node $(\{n_4\}, z_2 = 0)$ in $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ are regions over this unique clock z_2. If we consider a node combining n_5 and n_6, active clocks will consist in the union of active clocks in both nodes, hence z_2 and z_3. For sake of readability, we have mostly omitted labels of nodes on Fig. 3, but they can be naturally inferred from those in $R(\mathcal{A}^\infty)$; for instance, the label of the top-rightmost node is $\{(\ell_1, z_0), (\ell_3, z_1)\}$, the union of the labels of n_1 and n_2 in $R(\mathcal{A}^\infty)$.

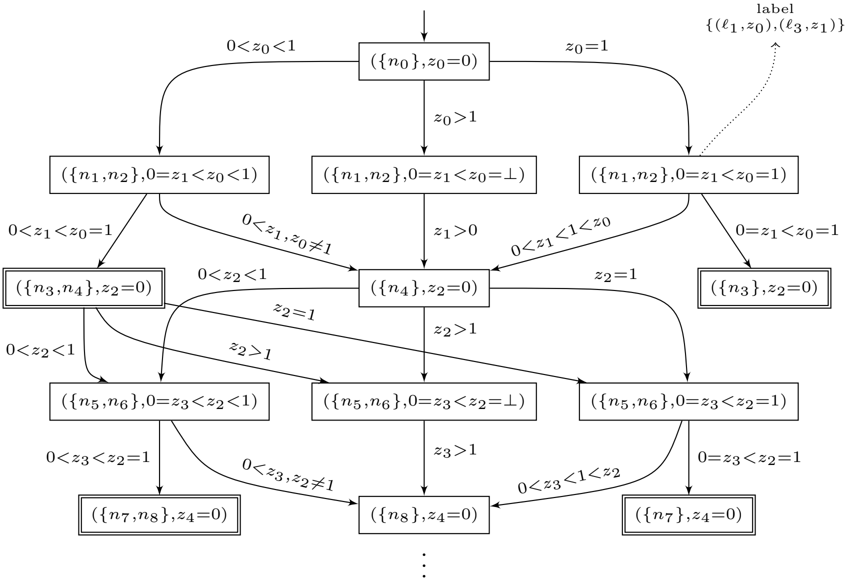


Fig. 3. The DAG induced by the infinite timed tree $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$

The subset construction induces a DAG (as seen on Fig. 3). However the rest of the construction will require a tree instead of a DAG; we thus add markers to nodes, so that we can have several copies of a node, depending on the ancestors. A node in $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ is thus a tuple $(\star, K, r)$ where $\star$ is a marker, K is a subset of node names in $R(\mathcal{A}^\infty)$ (they all have same level), and r is a region over the set $Act(K) = \bigcup_{n \in K, label(n)=(\ell,\sigma)} \sigma(X)$, the set of active clocks in K.

The correctness of $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ is stated in the following proposition.

Proposition 7. $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ *is a deterministic timed tree, and for every node* $N = (\star, K, r)$ *and for every valuation* $v \in \mathbb{T}^{Act(K)}$ *with* $v \in r$*,*

$$\mathcal{L}(\mathsf{SymbDet}(R(\mathcal{A}^\infty)), (N, v)) = \bigcup_{n \in K} \mathcal{L}(R(\mathcal{A}^\infty), ((n, r), v))$$

Remark 8. In case $\mathcal{A}$ has a single clock x, a level-i node of $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ carries the following information: a finite set of pairs of the form $(\ell, x \mapsto z_j)$ for some $j \leq i$ and a region for clocks in Z_i. We skip details, but with this information, we can easily recover the well-quasi-order that gives the decidability of the universality problem in single-clock timed automata [7].

3.4 Reduction of the Number of Clocks

$\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ is an infinite object (it is an infinite timed tree and it has infinitely many clocks). Our aim is to fold this tree back into a deterministic timed automaton. Obviously we cannot do so for all timed automata, and so far we have not made any assumption on $\mathcal{A}$. Given $\gamma \in \mathbb{N}$, we say that $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ is γ-*clock-bounded* if in every node, the number of active clocks is bounded by γ. Under this hypothesis, we will be able to quotient $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ by an equivalence of finite index, and get a deterministic timed automaton $\mathcal{B}_{\mathcal{A},\gamma}$ which accepts the same language as the original timed automaton $\mathcal{A}$.

The idea will be to fix a finite set of clocks $X_\gamma = \{x_1, \cdots, x_\gamma\}$, and starting from the level-0 node of $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ to rename the active clocks into clocks in X_γ following a deterministic policy. Under the γ-clock-boundedness assumption, each time we will require a new clock (because a new one has become active), there will be (at least) one free clock in X_γ. Of course, we rename clocks in guards and regions as well, and change the labels of the nodes accordingly (an element of the label of a node is now a pair (ℓ, σ) where ℓ is a location of $\mathcal{A}$ and $\sigma \colon X \mapsto X_\gamma$ assigns to each clock of $\mathcal{A}$ its representative in the tree). The new object is still infinite, but it has finitely many clocks. A node is now a tuple $(\star, K, r)$ where $\star$ is a marker, K is a subset of nodes in $R(\mathcal{A}^\infty)$ and r is a region over (a subset of) X_γ. Now it is just a matter of noticing that two nodes with the same region and the same labels are isomorphic and strongly timed bisimilar (in particular they are language-equivalent). Timed automaton $\mathcal{B}_{\mathcal{A},\gamma}$ is obtained by merging such nodes.

Example 9. In Fig. 3, it is easy to see that $\mathsf{SymbDet}(\mathcal{A}^\infty)$ is 2-clock-bounded. So one can rename the clocks to $X_2 = \{x_1, x_2\}$, for instance we can map clocks with even indices to x_1 and clocks with odd indices to x_2. After this renaming, nodes sharing the same label (that is: set of locations of $\mathcal{A}$, mappings from X to $\{x_1, x_2\}$ and regions over $\{x_1, x_2\}$) can be merged. Indeed, one can show that subtrees rooted at nodes with the same label are strongly timed bisimilar. For instance, in our running example, nodes $(\{n_0\}, z_0 = 0)$ and $(\{n_4\}, z_2 = 0)$, labelled respectively by $\{(\ell_0, z_0)\}$ and $\{(\ell_0, z_2)\}$ in $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$, are merged into a single location with region $x_1 = 0$. The resulting timed automaton is depicted on Fig. 4. In general, a location of this automaton is of the form $(\{(\ell_j, \sigma_j) \mid j \in J\}, r)$ where J is a finite set, ℓ_j is a location of $\mathcal{A}$, $\sigma_j \colon X \to X_2$, and r is a region over a subset of X_2. In our running example, there is a single clock x, hence we assimilate σ_j with the value $\sigma_j(x)$.

The correctness of the construction is stated in the following proposition.

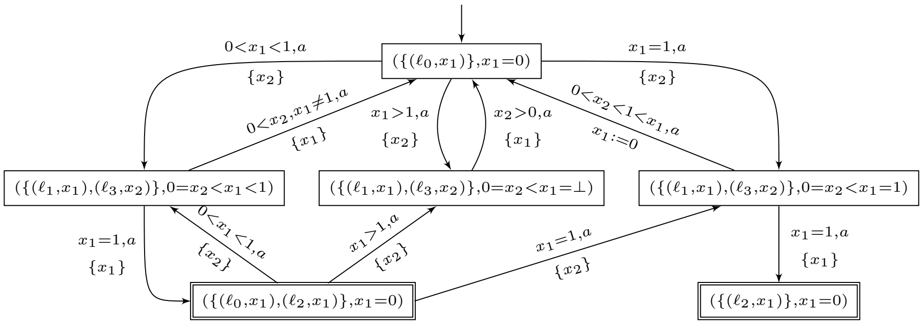

$(\{(\ell_0,x_1)\},x_1=0)$
$0<x_1<1,a$
$\{x_2\}$
$x_1=1,a$
$\{x_2\}$
$0<x_2,x_1\neq 1,a$
$\{x_1\}$
$x_1>1,a$
$\{x_2\}$
$x_2>0,a$
$\{x_1\}$
$0<x_2<1<x_1,a$
$x_1:=0$
$(\{(\ell_1,x_1),(\ell_3,x_2)\},0=x_2<x_1<1)$
$(\{(\ell_1,x_1),(\ell_3,x_2)\},0=x_2<x_1=\bot)$
$(\{(\ell_1,x_1),(\ell_3,x_2)\},0=x_2<x_1=1)$
$x_1=1,a$
$\{x_1\}$
$0<x_1<1,a$
$\{x_2\}$
$x_1>1,a$
$\{x_2\}$
$x_1=1,a$
$\{x_2\}$
$x_1=1,a$
$\{x_1\}$
$(\{(\ell_0,x_1),(\ell_2,x_1)\},x_1=0)$
$(\{(\ell_2,x_1)\},x_1=0)$

Fig. 4. The deterministic version of $\mathcal{A}$: the timed automaton $\mathcal{B}_{\mathcal{A},\gamma}$

Proposition 10. *Assume that* $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ *is* γ*-clock-bounded. Then,* $\mathcal{B}_{\mathcal{A},\gamma}$ *is a deterministic timed automaton, and* $\mathcal{L}(\mathcal{B}_{\mathcal{A},\gamma}) = \mathcal{L}(\mathcal{A})$*.*

3.5 Algorithmic Issues and Complexity

In this subsection, we shortly discuss the size of the effectiveness of its construction. If $\mathcal{A} = (L, \ell_0, L_{\mathsf{acc}}, X, M, E)$ is a timed automaton such that $\mathsf{SymbDet}$ $(R(\mathcal{A}^\infty))$ is γ-clock-bounded (for some $\gamma \in \mathbb{N}$), then the timed automaton $\mathcal{B}_{\mathcal{A},\gamma}$ has roughly $\alpha(\mathcal{A},\gamma) = 2^{|L|} \cdot \gamma^{|X|} \cdot \Big((2M+2)^{(\gamma+1)^2} \cdot \gamma!\Big)$ locations because a location is characterized by a finite set of pairs (ℓ, σ) with ℓ a location of $\mathcal{A}$, $\sigma \colon X \to X_\gamma$, and a region over X_γ.

The procedure we have described goes through the construction of infinite objects. However, if we abstract away the complete construction, we know precisely how locations and transitions are derived. Hence, $\mathcal{B}_{\mathcal{A},\gamma}$ can be computed on-the-fly by guessing new transitions, and so can its complement (since $\mathcal{B}_{\mathcal{A},\gamma}$ is deterministic). A location of the automaton $\mathcal{B}_{\mathcal{A},\gamma}$ can be stored in space logarithmic in $\alpha(\mathcal{A},\gamma)$, and we will thus be able to check for universality (e.g.) in nondeterministic space $\log(\alpha(\mathcal{A},\gamma))$.

4 Our Results

We will now investigate several classes of timed automata for which the procedure described in Section 3 applies.

4.1 Some Classes of Timed Automata Are Determinizable

Automata satisfying the p-assumption (TA_p). Given $p \in \mathbb{N}$, we say that a timed automaton $\mathcal{A}$ satisfies the *p-assumption* if for every $n \geq p$, for every run $\varrho = (\ell_0, v_0) \xrightarrow{\tau_1, a_1} (\ell_1, v_1) \ldots \xrightarrow{\tau_n, a_n} (\ell_n, v_n)$ in $\mathcal{A}$, for every clock $x \in X$, either x is reset along ϱ or $v_n(x) = \bot$. This assumption will ensure that we can apply the previous procedure, because if $\mathcal{A}$ satisfies the p-assumption, $\mathsf{SymbDet}(R(\mathcal{A}^\infty))$ is

p-clock-bounded. Then we observe that any strongly non-Zeno timed automaton (we write SnZTA for this class) satisfies the p-assumption for some $p \in \mathbb{N}$ which is exponential in the size of the automaton. We thus get the following result:

Theorem 11. *For every timed automaton $\mathcal{A}$ in* SnZTA *or in* TA_p*, we can construct a deterministic timed automaton $\mathcal{B}$, whose size is doubly-exponential in the size of $\mathcal{A}$, and which recognizes the same language as $\mathcal{A}$.*

Event-clock timed automata (ECTA) [3]. An *event-clock timed automaton* is a timed automaton that contains only event-recording clocks: for every letter $a \in \Sigma$, there is a clock x_a, which is reset at every occurrence of a. It is easy to see that the deterministic timed tree associated with such an automaton is $|\Sigma|$-clock-bounded. Thus, applying our procedure, we recover the result of [3], with the same complexity bound.

Theorem 12. *For every timed automaton $\mathcal{A}$ in* ECTA*, we can construct a deterministic timed automaton $\mathcal{B}$, whose size is exponential in the size of $\mathcal{A}$, and which recognizes the same language as $\mathcal{A}$.*

Timed automata with integer resets (IRTA) [9]. A *timed automaton with integer resets* is a timed automaton in which every edge $e = (\ell, g, a, Y, \ell')$ is such that Y is non empty if and only if g contains at least one atomic constraint of the form $x = c$, for some clock x. In that case, we observe that the deterministic timed tree associated with such an automaton is $(M + 1)$-clock-bounded. We thus recover the result of [9], with the same complexity bound.

Theorem 13. *For every timed automaton $\mathcal{A}$ in* IRTA*, we can construct a deterministic timed automaton $\mathcal{B}$, whose size is doubly-exponential in the size of $\mathcal{A}$, and which recognizes the same language as $\mathcal{A}$.*

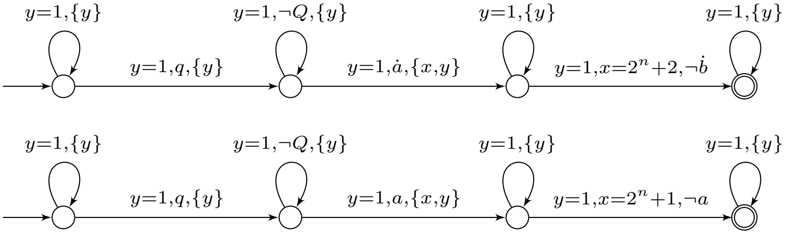
4.2 Deciding Universality and Inclusion

The universality and the inclusion problems are undecidable for the general class of timed automata [2]. Given $\mathcal{A}$ and $\mathcal{B}$ two timed automata, the *universality problem* asks whether $\mathcal{L}(\mathcal{A})$ is the set of all finite timed words, and the inclusion problem asks whether $\mathcal{L}(\mathcal{B}) \subseteq \mathcal{L}(\mathcal{A})$. When $\mathcal{A}$ belongs to one of the above determinizable classes, we will be able to decide the universality and the inclusion problems (there is no need to restrict automaton $\mathcal{B}$). We establish now the precise complexity of those problems, and start by providing a lower bound for the universality problem.

Proposition 14. *Checking universality in timed automata either satisfying the p-assumption for some p or with integer resets is EXPSPACE-hard.*

Proof (sketch). The idea of the proof is as follows. Given an exponential-space Turing machine $\mathcal{M}$ with input word w_0, we define a timed automaton $\mathcal{A}_{\mathcal{M},w_0}$ such that $\mathcal{A}_{\mathcal{M},w_0}$ is universal if and only if $\mathcal{M}$ does not halt on input w_0. An execution of $\mathcal{M}$ over w_0 is encoded by a timed word, and $\mathcal{A}_{\mathcal{M},w_0}$ will accept all finite timed words that are not encodings of halting executions of $\mathcal{M}$ on w_0. Assuming $|w_0| = n$, the maximal length of the tape is 2^n, and a configuration of $\mathcal{M}$ can be seen as a pair $\langle q, w\rangle$, where q is a control state of $\mathcal{M}$ and w is a word of length 2^n that represents the content of the tape (the position of the tape head is marked by a dotted letter). We furthermore require that actions are separated by precisely one time unit, which entails for instance that control states should be separated by precisely $2^n + 1$ time units.

A finite timed word might not be the encoding of an halting computation in $\mathcal{M}$ for several reasons: it is not the encoding of a proper execution in $\mathcal{M}$, or it does not end in the halting state, or actions do not occur at integer time points, or control states are not separated by 2^n+1 time units, *etc.* All these properties can be described using either timed automata satisfying the p-assumption, or timed automata with integer resets. For instance, a rule of the form $(q, a, b, \mathsf{right}, q')$ can be unfaithfully mimicked for two reasons: either the dotted letter (representing the position of the head) is not transferred properly (first automaton below), or the rest of the configuration is not copied properly (second automaton below).



All other cases can be handled in a similar way, which concludes the proof. □

This lower bound applies as well for the inclusion problem in the very same classes of timed automata. Note that strongly non-Zeno timed automata are never universal, but we can modify the above proof to show that the inclusion problem is EXPSPACE-hard as well for strongly non-Zeno timed automata.

Summary of the results. We can summarize our results in the following table. The column on the left indicates the subclass we consider. New results are in black and italic, and in particular we can notice that there was no lower bound known for the class IRTA.

	size of the det. TA	universality problem	inclusion problem
TA_p	*doubly exp.*	*EXPSPACE-complete*	*EXPSPACE-complete*
SnZTA	*doubly exp.*	trivial	*EXPSPACE-complete*
ECTA [3]	exp.	PSPACE-complete	PSPACE-complete
IRTA [9]	doubly exp.	EXPSPACE-*complete*	EXPSPACE-*complete*

5 Conclusion

In this paper, we proposed a general framework for the determinization of timed automata by means of an infinite timed tree. We showed that for a wide range of timed automata this infinite tree is language-equivalent to a deterministic timed automaton. The construction of this deterministic timed automaton yields the basis for algorithms to check universality or language inclusion. Concerning the complexity, these algorithms applied to event-clock timed automata [3] and timed automata with integer resets [9] provide tight bounds. In addition, our general framework yields the decidability of the universality problem for strongly non-Zeno timed automata, which was not known before.

We have focused on finite timed words, but we believe the procedure can be extended to timed automata over infinite timed words (with an ω-regular acceptance condition), by incorporating a Safra-like construction in our procedure. In that framework the strong non-Zenoness assumption will even make more sense, and we thus claim that strongly non-Zeno timed automata are determinizable!

References

1. Adams, S., Ouaknine, J., Worrell, J.: Undecidability of universality for timed automata with minimal resources. In: Raskin, J.-F., Thiagarajan, P.S. (eds.) FORMATS 2007. LNCS, vol. 4763, pp. 25–37. Springer, Heidelberg (2007)
2. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
3. Alur, R., Fix, L., Henzinger, T.A.: A determinizable class of timed automata. In: Dill, D.L. (ed.) CAV 1994. LNCS, vol. 818, pp. 1–13. Springer, Heidelberg (1994)
4. Asarin, E., Maler, O., Pnueli, A., Sifakis, J.: Controller synthesis for timed automata. In: Proc. IFAC Symposium on System Structure and Control, pp. 469–474. Elsevier Science, Amsterdam (1998)
5. Baier, C., Bertrand, N., Bouyer, P., Brihaye, T.: When are timed automata determinizable? Research Report LSV-09-08, Laboratoire Spécification & Vérification, ENS de Cachan, France (2009)
6. Finkel, O.: Undecidable problems about timed automata. In: Asarin, E., Bouyer, P. (eds.) FORMATS 2006. LNCS, vol. 4202, pp. 187–199. Springer, Heidelberg (2006)
7. Ouaknine, J., Worrell, J.: On the language inclusion problem for timed automata: Closing a decidability gap. In: Proc. 19th Annual Symposium on Logic in Computer Science (LICS 2004), pp. 54–63. IEEE Computer Society Press, Los Alamitos (2004)
8. Ouaknine, J., Worrell, J.: On the decidability and complexity of metric temporal logic over finite words. Logical Methods in Computer Science 3(1:8) (2007)
9. Suman, P.V., Pandya, P.K., Krishna, S.N., Manasa, L.: Timed automata with integer resets: Language inclusion and expressiveness. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 78–92. Springer, Heidelberg (2008)

Faithful Loops for Aperiodic E-Ordered Monoids*
(Extended Abstract)

Martin Beaudry[1] and François Lemieux[2]

[1] Département d'informatique, Université de Sherbrooke, Sherbrooke (Qc) Canada, J1K 2R1
martin.beaudry@usherbrooke.ca
[2] Département d'informatique et de mathématique, Université du Québec à Chicoutimi, Chicoutimi (Qc) Canada, G7H 2B1
flemieux@uqac.ca

1 Introduction

One of the main objectives of the algebraic theory of regular languages concerns the classification of regular languages based on Eilenberg's variety theorem [10]. This theorem states that there exists a bijection between varieties of regular languages and varieties of finite monoids[1]. For example, the variety of star-free regular languages (the closure of finite languages under Boolean operations and concatenation) is related to the monoid variety of aperiodic monoids (those with no nontrivial subgroups)[21].

Even if the theorem of varieties is a fundamental tool, it has some limitations since many interesting classes of languages are closed under left and right quotients, inverse morphisms, union and intersection but not under complementation. These classes are called *positive varieties of languages*. For example, the class of cofinite languages forms a positive variety. Another example is the *polynomial closure of group languages*. A group language is a regular language whose syntactic monoid is a finite group. The smallest class of languages that contains all group languages and is closed under union and concatenation is called the *polynomial closure of* G and is denoted $\mathrm{Pol}(G)$ [18]. It has been proved that $\mathrm{Pol}(G)$ is also the class of (regular) *open languages* for the group topology over the free monoid. As a consequence, $\mathrm{Pol}(G)$ is not closed under complementation but forms a positive variety of languages.

A counterpart to Eilenberg's variety theorem has been established in [17] for positive varieties of languages and the varieties of ordered monoids. An *ordered monoid* is a monoid M equipped with a partial order $\leq$ such that $x \leq y$ and $s \leq t$ implies $xs \leq yt$, for all $x, y, s, t \in M$. For example, a language is in $\mathrm{Pol}(G)$ if and only if it is recognized by an ordered monoid satisfying $e \leq 1$ for all idempotent $e \in M$ [18]. Such monoids are said to be *E-ordered*.

* Work supported by Québec FQRNT and NSERC of Canada.

[1] Varieties of semigroups will not be considered in this paper

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 55–66, 2009.

In [3], another algebraic characterization of Pol(G) was given in term of finite loops. A *loop* is a binary operation defined over a set B and, like a group, it satisfies the *cancellation laws*: $(ax = ay) \Rightarrow (x = y)$ and $(xa = ya) \Rightarrow (x = y)$ for all $a, x, y \in B$. Loops have mainly been investigated from two principal perspectives: algebra and combinatorics. The algebraic study of loops was initiated in the first half of the 20th century with the works of Moufang [14], Albert [1,2] and Bruck [6,7]. Group-theoretical notions like subgroups, quotients, morphisms, normal subgroups, nilpotency, and solvability extend smoothly into loop theory. Meanwhile, most combinatorial properties of loops are based on the fact that the multiplication table of a loop is a latin square. The study of latin squares was initiated by Euler with its famous *36 officers problem* (e.g. see [9]). Many techniques have been developed, for example, to build latin squares of any size or to complete a partially defined latin square. It is only recently that loops have been investigated from a language theoretic point of view.

Given a loop B and a word $w \in B^*$, we define the evaluation function $\eta_B : B^* \to \mathcal{P}(B)$ such that $\eta_B(w)$ is set of elements in B that can be obtained when evaluating w using all possible parenthesizations (for example, $\eta_B(abc) = \{a(bc), (ab)c\}$). We say that a language $L \subseteq A^*$ is recognized by B if there exists a morphism $\phi : A^* \to B^*$ and set $F \subseteq B$ such that $L = \{w \in A^* \mid \eta_B \circ \phi(w) \cap F \neq \emptyset\}$. The main result in [3] states that *both the finite E-ordered monoids and the finite loops recognize exactly the class of languages Pol(G).*

This result was refined in [4], where the class of *group-free loops* was studied, those loops B which do not have any nontrivial group *divisor* (i.e. a morphic image of a subloop of B). It was proved that aperiodic loops can only recognize star-free open languages, that is, those whose syntactic ordered monoid is aperiodic and E-ordered. The converse question remained open, whether any star-free open language can be recognized by a finite group-free loop. The initial motivation for the research reported on in this paper was to obtain an answer to this question, and this we did.

Theorem 1. *A language is star-free and open if and only if it is recognized by a finite aperiodic loop.*

Meanwhile, it was shown in [5] that to any loop B one can associate in a natural manner an E-ordered monoid $D(B)$, called the *derived monoid* of B. This monoid shares many important properties with B; in particular, every language recognized by B is also recognized by $D(B)$, and every group that divides $D(B)$ also divides B. In other words, *the language-recognizing capabilities of the (non-associative) loop B are totally encoded in the (associative) monoid $D(B)$.*

Proving Theorem 1 led us to define a counterpart to the concept of derived monoid: to an E-ordered monoid M, we associate a loop B such that every language recognized by M is also recognized by B, and every group which divides B also divides M; we say that B is *faithful* to M. We prove Theorem 1 by showing that faithful group-free loops exist for every aperiodic E-ordered monoid.

Our proof is constructive: from an E-ordered monoid M, we explicitely build a loop B that satisfies the desired properties. However, this implies the construction of a latin square that respects a particular set of constrains, and this turns out to be a complicated combinatorial task. For this reason, we first build from M a less rigid algebraic structure H and then embed H in a loop B.

A *halfloop* is a partially defined loop or, more formally, a partial binary function over a set H such that each product to the left or the right is a partial injective function. In particular, any loop is a halfloop. It will be convenient to assume that undefined products actually evaluate to an absorbing element, denoted by 0 or $\perp$. Recognition of languages by halfloops has the additional restriction that 0 must belong to the accepting set. The next proposition indicates that there is no loss of generality in replacing loops with halfloops:

Proposition 1. *Any halfloop H can be embedded in a loop G such that:*

1. *Every group which divides G also divides H,*
2. *Every language recognized by H is also recognized by G.*

The problem of embedding a halfloop into a loop was first solved by Evans [11]. His method does not, however, offer any guarantee on the structure and combinatorial properties of this loop. In our construction, no new group is created and no recognized language is lost. Theorem 1 will now follow from:

Proposition 2. *For any aperiodic E-ordered monoid M, there exists a group-free halfloop H such that every language recognized by M is also recognized by H.*

To prove this proposition, we introduce a concept which, we believe, is an interesting contribution of this paper. Homomorphisms are an essential tool in monoid theory, but they are of limited utility in our context. If $S(L)$ is the syntactic monoid of a regular language L and M is any monoid that also recognizes L then there must exist a morphism from a submonoid of M onto $S(L)$ (see [16,19]). However, homomorphisms preserve associativity, so that this property does not extend in general to halfloops. Nevertheless, if M is E-ordered, we can use a mapping which is *almost* a morphism.

Definition 1. *Let H be a halfloop and M an E-ordered monoid. A* semi-morphism *$\mu : H \to M$ is a surjective mapping satisfying $\mu(1) = 1$ and $\mu(a)\mu(b) \leq \mu(ab)$.*

A special case of semi-morphisms will be central to our proof. Let $\varphi : M^* \to M$ be the natural morphism and let η be the evaluation function on H.

Definition 2. *A semi-morphism $\mu : H \to M$ is* faithful *if there exists a monoid morphism $\alpha : M^* \to H^*$ extending an injective mapping $\alpha : M \to H$ such that:*

1. *$\alpha(1) = 1$*
2. *$\mu \circ \alpha = Id_M$, where Id_M is the identity mapping on M.*
3. *For all $w \in M^*$, $\varphi(w) \in \mu \circ \eta \circ \alpha(w)$.*

Lemma 1. *If $\mu : H \to M$ is a faithful semi-morphism, then H recognizes all the languages recognized by M.*

	1	2	3	4	5
1	1	2	3	4	5
2	2	3	1	5	4
3	3	4	5	1	2
4	4	5	2	3	1
5	5	1	4	2	3

Fig. 1.

Example 1. Let B_5 be the loop described in Figure 1 and let U_1 be the E-ordered monoid defined over the two elements 1 (identity) and 0 (absorbing) with the order $0 \leq 1$.

Define the mapping $\mu : B_5 \rightarrow U_1$ such that $\mu(1) = 1$ and $\mu(b) = 0$ for all $b \neq 1$. Then, μ is a faithful semi-morphism from B_5 to U_1. We can conclude that B_5 recognizes the language A^*aA^*.

In this paper, we assume that the reader is familiar with the algebraic theory of automata and regular languages, see e.g. [16,19]. The essential facts on finite loops are given in the next section, see [8,15] for more background. Sections 3 and 4 contain an outline for the proofs of Propositions 1 and 2, respectively. All details are given in the full paper. We conclude with some comments on future work.

2 Background on Loops

A *groupoid* (also called *magma*) is given by a set and a binary operation. In this paper, we assume that all groupoids contain a two-sided identity element usually denoted 1. Therefore, a monoid is an associative groupoid while a finite loop is a cancellative groupoid. A finite groupoid that is both associative and cancellative is a group. Excepting the free monoids, all groupoids considered in this paper are finite. Upon introducing a groupoid, we will either use a notation of the form $(G, \star)$ to specify that the set of values is G and that the operation is denoted by $\star$, or we will simply write G.

A groupoid G is said to *divide* another groupoid L, which we denote by $G \prec L$, if G is a homomorphic image of a subgroupoid of L. It is easily verified that if L is a loop and $G \prec L$, then G is also a loop.

Given a groupoid G and a word $w \in G^*$, we use $\eta_G(w)$ to denote the set of those values of G that can be obtained by evaluating w in G in all possible ways, i.e. using all possible parenthetizations. As a particular case, we define $\eta(\epsilon) = \{1\}$. When G is associative, then $\eta_G(w)$ is a singleton. When it is clear from the context which groupoid the evaluation takes place in, the subscript G is omitted.

A loop is said to be *group-free* if has no nontrivial group divisor. The loop B_5 of Figure 1 is the smallest group-free loop. Also, the smallest non-associative loops have size 5; from now on, therefore, all loops we work on are assumed to have size at least 5.

Groupoids are used to to recognize subsets of A^*, in the following manner. We say that $L \subseteq A^*$ is recognized by the groupoid G if there exist an *alphabetic* morphism $\phi : A^* \to G^*$ (i.e. extending a mapping $A \to G$) and a subset F of G such that $L = \{x \in A^* \mid \eta(\phi(x)) \cap F \neq \emptyset\}$. Note that when G is associative, we are back to the definition given for monoids. In terms of language recognition, the power of finite groupoids and of finite loops has been characterized precisely, while knowledge on the languages recognized by finite group-free loops was incomplete.

Lemma 2.

1. [13] *A language L is recognizable by a finite groupoid iff it is context-free.*
2. [3] *L is recognizable by a finite loop iff L is an open regular language.*
3. [4] *L is recognizable by a finite group-free loop only if L is an open star-free language.*

3 Semi-morphisms and Faithful Halfloops

For $s \in M$, we will often use the notation $H(s) = \mu^{-1}(s)$ and for $S \subseteq M$ we define $H(S) = \bigcup_{s \in S} H(s)$. We now give a proof of Lemma 1 :

Proof (of Lemma 1). It suffices to consider languages of the form $L = \varphi^{-1}(F)$ where F is an ordered ideal of M. We use $K = H(F) \cup \{\perp\}$ as accepting set and show that $L = \{w \in M^* \mid \eta \circ \alpha(w) \cap K \neq \emptyset\}$. If $w \in L$ then $\varphi(w) \in \mu(\eta \circ \alpha(w)) \cap F$, since μ is faithful. Hence, $\eta \circ \alpha(w) \cap K \neq \emptyset$ and H accepts $\varphi(w)$. It remains to show that H does not accept w whenever $w \notin L$. This is a consequence of the fact that for all $w \in M^*$, $\eta \circ \alpha(w) \subseteq \overline{\varphi(w)}$ which can easily be proved by induction on the length of w.

Definition 3. *A loop or halfloop G is said to be* faithful *to an E-ordered monoid M if there exists a faithful semi-morphism $\mu : G \to M$.*

In this paper, we are particularly interested to the case where G is a group-free loop and M a E-ordered aperiodic monoid.

We now give a more involved example of the application of lemma 1. For any $n \geq 2$, we define the Brandt monoids BA_n over the set $\{0, 1\} \cup \{a_{ij} \mid 1 \leq i, j \leq n\}$. Here, 1 is the identity, 0 the absorbing element, and the product $a_{ij}a_{k\ell}$ is $a_{i\ell}$ if $j = k$, or 0 otherwise. We will consider BA_n as an E-ordered monoid, where the stable order satisfies $a_{ii} \leq 1$ for all $1 \leq i \leq n$ and $0 \leq m$ for all m.

Theorem 2. *There exists a group-free halfloop faithful to the Brandt monoid BA_n.*

Proof. We first observe that BA_n is "almost" a halfloop. The only situations where the cancellation law is violated are $a_{ii}a_{ij} = a_{ij} = 1a_{ij}$ and $a_{ij}a_{jj} = a_{ij} = a_{ij}1$.

Let $B_5 = \{1, 2, 3, 4, 5\}$ be the group-free loop depicted in Figure 1 . Let $S = (\{a_{ij}|i \neq j\} \times B_5) \cup (\{a_{ii}|1 \leq i \leq 5\} \times \{2, 3, 4, 5\})$. We define a halfloop H over the set $S \cup \{\epsilon, \perp\}$ where ϵ is the identity, $\perp$ is absorbing, and the value of $\langle a_{ij}, p\rangle\langle a_{k\ell}, q\rangle$ is $\perp$ if $j \neq k$, $\langle a_{i\ell}, pq\rangle$ if $j = k$ and $i \neq \ell$, and ϵ if $j = k$, $i = \ell$ and $pq = 1$, that is, where the result would have been $\langle a_{ii}, 1\rangle$ if this pair had been in S. We claim that H is faithful to BA_n.

Define $\mu : H \to BA_n$ by $\mu(\perp) = 0$, $\mu(\epsilon) = 1$ and $\mu(\langle x, y\rangle) = x$. Define also $\alpha : BA_n \to H$ by $\alpha(0) = \perp$, $\alpha(1) = \epsilon$ and $\alpha(a_{ij}) = \langle a_{ij}, 2\rangle$. Clearly, μ is a semimorphism. The only nontrivial item upon verifying that it is faithful, consists in verifying that a word $w \in (BA_n)^*$ such that $\varphi(w) = a_{ii}$ (φ is the natural homomorphism) satisfies $\eta \circ \alpha(w) \neq \{\epsilon\}$. To see this, observe that if $\varphi(w) = a_{ii}$, then $\alpha(w)$ has the form $\langle a_{ij}, 2\rangle \cdots \langle a_{ki}, 2\rangle$. If $|w| = 1$, observe that $\langle a_{ii}, 2\rangle$ is the only $x \in H^*$ of length 1 which satisfies $a_{ii} \in \mu \circ \eta(x)$; that if $|w| = 2$, then $\alpha(w) = \langle a_{ij}, 2\rangle\langle a_{ji}, 2\rangle$ evaluates to $\langle a_{ii}, 3\rangle$, and that for $|w| \geq 3$, since $|\eta(2^n)| \geq 2$ for all $n \geq 3$, the subset $\eta \circ \alpha(w)$ of H cannot be a singleton.

4 Embedding a Halfloop into a Loop

In this section, we give a proof Proposition 1. Our constructions relies heavily on the fact that the loop's table is a latin square. We will use the following classical theorems from the theory of latin squares; a *latin rectangle* over n objects is a $r \times s$ table, $r, s \leq n$, where each object occurs at most once per line or column.

Theorem 3 (Hall [12]). *For any $m \times n$ latin rectangle over n objects, $m \leq n$, there exists a latin square over the same n objects whose first m lines are exactly those of the latin rectangle. In other words, any such latin rectangle can be extended into a latin square.*

Theorem 4 (Ryser [20]). *Consider a $r \times s$ latin rectangle with entries from the set $\{1, \ldots, q\}$. Let $N(i)$ denote the number of times the entry i occurs in the latin rectangle. A necessary and sufficient condition that this rectangle can be extended to a $q \times q$ latin square, is that $N(i) \geq r + s - q$ for each i.*

Our proof consists in three steps. First, we embed H into a halfloop H' where each nonabsorbing element has a left and a right inverse. Then, we embed H' in a loop G'. Finally, we verify that G' has the claimed properties.

Step 1. Let $H = \{a_1, \ldots, a_{n-1}, \epsilon, \perp\} = A \cup \{\epsilon, \perp\}$, where ϵ denotes the identity, and denote by "$\cdot$" the operation of the halfloop H. Let p be the number of lines (and columns) of the table of H where ϵ is absent. Define $\{a^1_1, \ldots, a^1_p\} = \{\ a \in A \mid \forall\, b \in H,\ ab \neq \epsilon\ \}$ and $\{a^2_1, \ldots, a^2_p\} = \{\ a \in A \mid \forall\, b \in H,\ ba \neq \epsilon\ \}$. We define on the set $H' = H \cup \{b_1, \ldots, b_p\}$ the operation $\star$ by

- if $c, d \in H$, then $c \star d = c \cdot d$,
- for each $c \in H'$, $\epsilon \star c = c \star \epsilon = c$,
- for each $i \in \{1, \ldots, p\}$, $a^1_i \star b_i = \epsilon$ and $b_i \star a^2_i = \epsilon$,
- everything else evaluates to $\perp$.

	Γ_0	Γ_1
Γ_0	Z_0	Z_1
Γ_1	Z_1'	Z_0'

	Γ_0	Γ_1	Γ_2	Γ_3	Γ_4	Γ_5
Γ_0	Z_0	Y_2	Y_3	Y_4	Y_5	Z_1
Γ_1	Y_2	X_1	Y_0	Y_5	Y_3	Y_4
Γ_2	Y_5	Y_4	X_2	Y_0	Y_1	Y_3
Γ_3	Y_4	Y_5	Y_1	X_3	Y_0	Y_2
Γ_4	Z_2	Y_3	Y_5	Y_2	X_4	Z_3
Γ_5	Y_3	Y_0	Y_4	Y_1	Y_2	X_5

Fig. 2. The main subtables of H' (left) and G (right)

It is readily verified that H' with the operation $\star$ is a halfloop. Since $b \star b = \bot$ for each $b \notin H$, no element of $H' \setminus H$ can belong to a loop, and the subloops of H' are exactly those of H (note that there are several sub*groupoids* of H').

Step 2. The second step consists in embedding H' into a loop G'. For this, we use Theorem 4. Let $\Gamma_0 = H' \setminus \{\epsilon, \bot\}$ and let $m = n + p - 1$ denote the size of Γ_0. Moreover. let Γ_1 be a copy of Γ_0 and build an arbitrary $m \times m$ latin square Q over Γ_1

We superimpose the table of H' onto Q, that is, we replace the occurrences of $\bot$ in the table with the corresponding entry in Q. This creates a $(m+1) \times (m+1)$ latin rectangle over $G' = \{\epsilon\} \cup \Gamma_0 \cup \Gamma_1$. Observe that this can be done in such a way that each element of Γ_1 occurs at least once in this rectangle, so that the conditions for Theorem 4 are satisfied. The rectangle can thus be extended into a latin square over G'. Through appropriate permutations of the lines and columns indexed with elements of Γ_1, this latin square can be regarded as the table of a loop G'.

We partition its main subtable into four $m \times m$ zones (see Figure 2). By construction, each line and column of Z_0 contains an occurrence of ϵ, so that we can assert that no such occurrence exists in Z_1 or Z_1'. There are not much constraints on the whereabouts of ϵ inside zone Z_0', however, and therefore the loop G' can have unwanted divisors.

Step 3. The general structure of the final loop G that we want to build is illustrated on Figure 2. Here, $\Gamma_2 \cdots \Gamma_5$ are four copies of Γ_0. In this picture, a line (resp. column) labelled Γ_i represents $|\Gamma_i|$ lines of the actual table, one for each element of Γ_i. For $1 \leq i \leq 5$, X_i is the main subtable of a loop over $\Gamma_i \cup \{\epsilon\}$ isomorphic to a group-free loop B_{m+1} and, for $0 \leq j \leq 5$, Y_j represents any latin square over Γ_i.

Regions Z_0 and Z_1 (at the top left and right or the table) are identical to those in G'. In the columns labelled with Γ_0 and Γ_5, the remaining occurrences of elements of Γ_0 and Γ_1 are gathered in regions denoted Z_2 and Z_3 (on the line of Γ_4). To build Z_2 and Z_3 we first suppose that all lines labelled Γ_4 in Figure 2 are empty. Using Theorem 3 to complete the square will force the elements of $\Gamma_0 \cup \Gamma_1$ to be placed in Z_2 and Z_3.

To verify that the only nontrivial groups contained in G are those that were in H, the reader can convince himself by inspecting Figure 2, that any pair $\{g, h\}$ where g and h do not belong to the same subset Γ_i generates the whole of G, that a subset of Γ_i, $i \neq 0$, generates a subloop of $\Gamma_i \cup \{\epsilon\}$, and that a subset of Γ_0 generates either a subset of Γ_0, i.e. a subloop of H, or the whole of G in the case where some combination of its elements can evaluate into a value $g \in \Gamma_1$.

Also, no new group divides G. Since loop homomorphism have the same basic properties as group homomorphism, verifying this can be done in a classical way, by showing that the kernel of the homomorphism cannot be a proper subloop, or that none of the subloops of G can be normal.[2] In other words, all homomorphisms from G are trivial.

Finally, we show that every language recognized by H with $\perp$ in the accepting set F is also recognized by G. Let $A = \Gamma_0 \cup \{\epsilon\}$. It is sufficient to prove that for any $w = w_1 \cdots w_n \in A^+$, such that $\perp \in \eta_H(w)$, the set $\eta_G(w)$ must contain an element in $G - A$. Such a word can be expressed as $w = sut$, where u is the leftmost segment of w with $\perp \in \eta_H(u)$ and u contains no such proper segment. Then, we have $\eta_G(s) \subseteq A$, $\eta_G(u) \cap \Gamma_1 \neq \emptyset$. Hence, we only have to check that a word in $A\Gamma_1 A^*$ can always be evaluated into some element of $G - A$. This is done by an exhaustive analysis of which subsets Γ_i the suffix $t \in A^*$ can evaluate into.

Our construction actually preserves faithfulness:

Lemma 3. *Let $\mu : H \to M$ be a faithful semi-morphism from a halfloop H to an E-ordered monoid M, and let G be the loop obtained from H by the above construction. Then μ can be extended to $G - H$ by setting $\mu(g) = 0$ for all $g \in G - H$, and the resulting function $\mu : G \to M$ is a faithful semi-morphism. In other words, if H is faithful to M then so is G.*

5 Construction of Faithful Group-Free Halfloops

In this section, we sketch our method for building a faithful group-free halfloop from an aperiodic E-ordered monoid.

From now on, let $M = (M, \leq)$ be an aperiodic ordered monoid satisfying $x^\omega \leq 1$, where 1 is the identity. In practice, we will assume no further property for $\leq$. It can be easily proved that M has a zero element, which we denote by 0.

In an E-ordered monoid containing a zero element, we say that a $\mathcal{J}$-class C is *0-minimal* whenever $C \neq \{0\}$ and $C \cup \{0\}$ is an ideal.

We work by induction on the lattice of $\mathcal{J}$-classes of M: we start with $M_0 = \{0, 1\}$, and iteration $i + 1$ consists in creating the monoid M_{i+1} from M_i by inserting just above its minimal ideal a $\mathcal{J}$-class C of M, such that C is 0-minimal in M_{i+1}. Concretely, $M_{i+1} = M_i \cup C$, and the operation is defined on $s, t \in M_i$ by

- if $st = u$ in the monoid M with $u \in M_{i+1}$, then $st = u$ in the monoid M_{i+1};
- if $st = u$ in M with $u \notin M_{i+1}$, then $st = 0$ in M_{i+1}.

[2] A subloop K is normal if it satisfies the three conditions $gK = Kg$, $(gh)K = g(hK)$ and $K(gh) = (Kg)h$.

We embed the triple (H_i, μ_i, α_i), faithful to M_i by induction hypothesis, in $(H_{i+1}, \mu_i, \alpha_i)$, faithful to M_{i+1}, by adding to H_i a set $X = H_{i+1} \setminus H_i$ of new elements, and adjusting the operation, so that

- for all $a \in H_i$ and $m \in M_i$, $m \neq 0$, $\mu_{i+1}(a) = \mu_i(a)$ and $\alpha_{i+1}(m) = \alpha_i(m)$;
- for all $a \in X$, $\mu_{i+1}(a) \in C$;
- for all $m \in C$, $\alpha_{i+1}(m) \in X$;
- if $ab = c$ in H_i with $c \neq \perp$, then $ab = c$ in H_{i+1}.

Once the values of $\mu_i(a)$ and $\alpha_i(m)$ are determined, they remain the same until the end of the induction. Therefore, from now on we use the notations $\mu(a)$, $\alpha(m)$, and $H(m) = \mu^{-1}(m)$.

A graphic interpretation of the induction step consists in taking the table $\mathcal{T}(H_i)$ of the operation in H_i, adding the lines and columns corresponding to the new elements of X, defining the content of the newly defined cells in the table, and replacing the appropriate occurrences of $\perp$ in $\mathcal{T}(H_i)$ with values from X. The nonzero entries in $\mathcal{T}(H_i)$ remain unchanged. We will sketch the construction at the induction step for two of the four possible cases.

Induction basis. The monoid M_0 is isomorphic to the E-ordered monoid U_1 defined in example 1.

Definition 4. *Given an element $m \in M$, we define its* dominating set *by $\overline{m} = \{ n \in M \mid m \leq n \}$; we extend this notation to subsets of M: if $E \subset M$, then let $\overline{E} = \bigcup_{m \in E} \overline{m}$.*

With this notation, a surjective mapping φ from a halfloop H to M is a semimorphism iff $\varphi(1) = 1$ and $\mu(ab) \in \overline{\mu(a)\mu(b)}$ for all $a, b \in H$.

Induction step; the regular and trivial case. Here, the new $\mathcal{J}$-class is $C = \{e\}$, where $ee = e$. Our construction uses the following properties of dominating sets.

Lemma 4. *Let $e \in E(M)$. Then, $\overline{e}$ is a submonoid of M and it holds $\overline{e} = \{ x \in M \mid xe = e \} = \{ x \in M \mid ex = e \}$.*

Lemma 5. *If $m \leq_{\mathcal{J}} s$ and m belongs to a regular $\mathcal{J}$-class, then there exists $m' \mathrel{\mathcal{J}} m$ such that $m' \leq s$.*

	K	$H_i(\overline{e})$
K	(1)	(2)
$H_i(\overline{e})$	(3)	(4)

We partition $\mathcal{T}(H_i)$ into four regions; in the table above, $K = H_i \setminus H_i(\overline{e})$. By Lemmas 4 and 5, only region (4) contains positions $[a, b]$ such that $\mu(a)\mu(b) \in \overline{e}$. Moreover, every occurrence of $\perp$ in this region is at a position $[a, b]$ such that $\mu(a)\mu(b) = e$ in M. Therefore, region (4) is embedded in an aperiodic loop using the method of the previous section; this yields a loop $H_{i+1}(\overline{e}) = H(\overline{e})$; in total, the resulting loop is $H_{i+1} = H_i \cup H(\overline{e})$, and we set $H(e) = H(\overline{e}) - H_i(\overline{e})$ and $\alpha(e) = \gamma$ for some $\gamma \in H_{i+1} - H_i$.

Proposition 1 ensures that the resulting H_{i+1} is group-free. The fact that H_{i+1} is faithful to M_{i+1} follows from the induction hypothesis and Lemma 3.

	D_0	D_1	D_2	D_3	X
G_0	(1)	...	(2)	...	$\perp$
G_1	...	...	(4)	...	(10)
G_2	(3)	(5)	(6)	(7)	$\perp$
G_3	...	...	(8)	(9)	(12)
X	$\perp$	(11)	$\perp$	(13)	$\perp$

	$\varDelta$	$\varDelta'$	d	$\varTheta'$	X'
$\varDelta$	$\varDelta \cup X'$	X'	$\varDelta'$	X'	
$\varDelta'$	X'			X'	
d	$\varDelta'$				
$\varTheta'$	X'	X'		X'	
X'					

Fig. 3. Case 2: the table $\mathcal{T}(H_{i+1})$ (left) and the partial latin square $\mathcal{Q}$ (right)

Induction step; the trivial non-regular case. Here, $C = \{m\}$ with $m^2 = 0$. Let $X = H_{i+1} \setminus H_i$ be the set of the halfloop elements created at this induction step. Building the table $\mathcal{T}(H_{i+1})$ is done in three steps: first, we identify those regions in $\mathcal{T}(H_{i+1})$ where the entries will be elements of X; second, we build a specific latin square over in a subset of H_{i+1}; in the third step we fill the appropriate entries in $\mathcal{T}(H_{i+1})$, using the latin square as a template. For the first step, we create two partitions of H_i, in order to use the following.

Lemma 6. *With $s, t \in M_i$, if $sm = m$, then $m \leq t \Leftrightarrow m \leq st$ (dually: $mt = m$ and $m \leq s \Leftrightarrow m \leq st$.*

$$\begin{aligned}
G_0 &= H(\{\, s \in M_i \mid s \notin \overline{m} \wedge sm = 0 \,\}); & D_0 &= H(\{\, t \in M_i \mid t \notin \overline{m} \wedge mt = 0 \,\});\\
G_1 &= H(\{\, s \in M_i \mid s \notin \overline{m} \wedge sm = m \,\}); & D_1 &= H(\{\, t \in M_i \mid t \notin \overline{m} \wedge mt = m \,\});\\
G_2 &= H(\{\, s \in M_i \mid s \in \overline{m} \wedge sm = 0 \,\}); & D_2 &= H(\{\, t \in M_i \mid t \in \overline{m} \wedge mt = 0 \,\});\\
G_3 &= H(\{\, s \in M_i \mid s \in \overline{m} \wedge sm = m \,\}); & D_3 &= H(\{\, t \in M_i \mid t \in \overline{m} \wedge mt = m \,\}).
\end{aligned}$$

This enables us to divide table $\mathcal{T}(H_{i+1})$ into 25 zones, as depicted in Figure 3. In the five regions identified with $\perp$, all entries are null. Seven other zones, identified with "...", cannot contain entries from $H(\overline{m})$, either by their definition or as a consequence of Lemma 6. For example, if $s \in \mu(G_0)$ and $t \in \mu(D_1)$, then $st = m$ is not possible since otherwise we would have $m = st = mt \Rightarrow m = st^{\omega} \leq s$ which contradict the definition of G_0.

The remaining blocks carry a label running from (1) to (13).

In this table, the line of $a \in H_{i+1}$ consists of all positions of the form $[a, b]$; if $a \in G_1$, then all entries in region (10) must belong to $H(\overline{m})$ if we want the halfloop to be faithful, and we will want to have an element of X at every position of region (4) where a $\perp$ has to be replaced. This means more positions to fill than there are elements in X: for every such element placed in region (4), a position in region (10) must be filled with an element of $H_i \cap H(\overline{m})$. We show that there are indeed enough of them.

For the second step, let $\varDelta = H_i(\overline{m}) = G_2 \cup G_3 = D_2 \cup D_3$ and $\varTheta = H_i \setminus \varDelta$. We partition $X = H_{i+1} \setminus H_i$ into four subsets, $X = \{d\} \cup \varDelta' \cup \varTheta' \cup X'$, where $|\varDelta'| = |\varDelta| = \delta$, $|\varTheta'| = |\varTheta| = \theta$, and $|X'| > 2\delta + \theta + 1$. Let $|X| = \chi$.

The partially defined latin square $\mathcal{Q}$ of Figure 3 represents a modification of the $(\delta + \chi) \times (\delta + \chi)$ subtable of $\mathcal{T}(H_{i+1})$ whose lines and columns belongs to $\varDelta \cup X$: Some entries which previously were containing $\perp$ have been filled with elements from X.

Let $\mathcal{Q}[a, b]$ denote the entry at position $[a, b]$.

- in each position $[a, b]$ of the block labelled with $\Delta \cup X'$, either $\mathcal{T}(H_i)[a, b] \in \Delta$ and then $\mathcal{Q}[a, b] = \mathcal{T}(H_i)[a, b]$, or $\mathcal{T}(H_i)[a, b] = \perp$, and we set $\mathcal{Q}[a, b] \in X'$;
- In a block labelled with a set X' or Δ', all entries must belong to this set.
- Other blocks contain only $\perp$

The set X' is large enough to ensure that $\mathcal{Q}$ satisfies the condition for Ryser's theorem (Theorem 4): this yields a latin square $\mathcal{S}$ over the set $X \cup \Delta$.

Let $\mathcal{T}(H_{i+1})[a, *]$ denote the line of a in $\mathcal{T}(H_{i+1})$, (dually, $\mathcal{T}(H_{i+1})[*, a]$ for the column) and define similar notations for $\mathcal{S}$. The third step consists in doing the following.

- For each $a \in \Delta$, use the line $\mathcal{S}[a, *]$ to fill the appropriate positions on the line $\mathcal{T}(H_{i+1})[a, *]$; this involves the regions of $\mathcal{T}(H_{i+1})$ labelled with 6, 7, 8, 9 and 12. Dually, use the line $\mathcal{S}[*, a]$ to fill $\mathcal{T}(H_{i+1})[*, a]$ which involves the regions of $\mathcal{T}(H_{i+1})$ labelled with 6, 7, 8, 9 and 13.
- Define some bijection $\tau : \Theta \to \Theta'$. For each $a \in \Theta$, define $\mathcal{T}(H_{i+1})[a, *]$ using in $\mathcal{S}[\tau(a), *]$, which means the regions labelled with 2, 4 and 10. Dually, define $\mathcal{T}(H_{i+1})[*, a]$ using in $\mathcal{S}[*, \tau(a)]$, which means the regions labelled with 3, 5 and 11.
- The appropriate entries of region 1 are defined by setting $\mathcal{T}(H_{i+1})[a, b] = \mathcal{S}[\tau(a), \tau(b)]$.

Finally, we define $\alpha(m) = d$ and $H(m) = X$. Verifying faithfulness reduces to proving condition 3 of Definition 2; the other conditions are satisfied by construction.

6 Conclusion

If the apparent absence of structure in finite groupoids may have been seen initially as an absolute barrier to comprehension, the results of this paper together with related results obtained in the last decade demonstrate that several fascinating facts and questions can be obtained by concentrating on key special cases.

Thus, the relationship established in this paper and its predecessors, between finite loops and E-ordered monoids, offers a different perspective to the study of regular languages, and uncovers a number of new questions. For example, what languages are recognized by nilpotent or solvable loops?

Moreover, what can be said on the algebraic structure of loops that recognize, for example, the open piecewise-testable languages? These languages are recognized by ordered monoids that satisfy the equation $x \leq 1$.

Another question is related to the fact that any groupoid whose multiplication monoid belongs to the variety *DO* can only recognize regular languages [5]. Hence, to what extent the definition of faithful semi-morphisms introduced in this paper could also be applied to other kinds of groupoids and ordered monoids?

Finally, Proposition 1 still has to be extended to arbitrary E-ordered monoids and loops; yet the crop of new concepts gathered while dealing with the group-free case is already likely to give us powerful insight on the structural and language-theoretical properties of finite loops and groupoids.

References

1. Albert, A.A.: Quasigroups. I. Trans. Amer. Math. Soc. 54, 507–519 (1943)
2. Albert, A.A.: Quasigroups. II. Trans. Amer. Math. Soc. 55, 401–419 (1944)
3. Beaudry, M., Lemieux, F., Thérien, D.: Finite loops recognize exactly the regular open languages. In: Degano, P., Gorrieri, R., Marchetti-Spaccamela, A. (eds.) ICALP 1997. LNCS, vol. 1256, pp. 110–120. Springer, Heidelberg (1997)
4. Beaudry, M., Lemieux, F., Thérien, D.: Star-free open languages and aperiodic loops. In: Ferreira, A., Reichel, H. (eds.) STACS 2001. LNCS, vol. 2010, pp. 87–98. Springer, Heidelberg (2001)
5. Beaudry, M., Lemieux, F., Thérien, D.: Groupoids that recognize only regular languages. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 421–433. Springer, Heidelberg (2005)
6. Bruck, R.H.: Contributions to the Theory of Loops. Trans. Amer. Math. Soc. 60, 245–354 (1946)
7. Bruck, R.H.: A Survey of Binary Systems. Springer, Heidelberg (1966)
8. Chein, O., Pfugfelder, H.O., Smith, J.D.H.: Quasigroups and Loops: Theory and Applications. Helderman Verlag, Berlin (1990)
9. Denes, J., Keedwell, A.D.: Latin squares and their applications. Academic Press, London (1974)
10. Eilenberg, S.: Automata, Languages and Machines, vol. B. Academic Press, London (1976)
11. Evans, T.: Embedding incomplete latin squares. Amer. Math. Monthly 67, 958–961 (1960)
12. Hall, M.: An existence theorem for latin squares. Bull. Amer. Math. Soc. 51, 387–388 (1945)
13. Mezei, J., Wright, J.B.: Algebraic automata and context-free sets. Information and Control 11, 1–29 (1967)
14. Moufang, R.: Zur Struktur von Alternativkörpern. Mathematische Annalen 110, 416–430 (1935)
15. Pfugfelder, H.O.: Quasigroups and Loops: Introduction. Heldermann Verlag (1990)
16. Pin, J.-E.: Varieties of Formal Languages. Plenum Press, New York (1986)
17. Pin, J.-E.: A variety theorem without complementation. Russian Mathematics (Izvestija vuzov. Matematika) 39, 80–90 (1995)
18. Pin, J.-E.: Polynomial closure of group languages and open set of the Hall topology. Theoretical Computer Science 169, 185–200 (1996)
19. Pin, J.-É.: Syntactic semigroups. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of formal languages. Springer, Heidelberg (1997)
20. Ryser, H.J.: A combinatorial theorem with an application to latin rectangles. Proc. Amer. Math. Soc. 2, 550–552 (1951)
21. Schützenberger, M.P.: On Finite Monoids having only trivial subgroups. Information and Control 8, 190–194 (1965)

Boundedness of Monadic Second-Order Formulae over Finite Words

Achim Blumensath[1], Martin Otto[1], and Mark Weyer[2]

[1] Fachbereich Mathematik, Technische Universität Darmstadt
[2] Lehrstuhl Logik in der Informatik, Humboldt-Universität zu Berlin

Abstract. We prove that the boundedness problem for monadic second-order logic over the class of all finite words is decidable.

1 Introduction

In applications one frequently employs tailor-made logics to achieve a balance between expressive power and algorithmic manageability. Adding fixed-point operators to weak logics turned out to be a good way to achieve such a balance. Think, for example of the addition of transitive closure operators or more general fixed-point constructs to database query languages, or of various fixed-point defined reachability or recurrence assertions to logics used in verification. Fixed-point operators introduce a measure of relational recursion and typically boost expressiveness in the direction of more dynamic and less local properties, by iteration and recursion based on the expressiveness that is locally or statically available in the underlying fragment, say of first-order logic FO. We here exclusively consider monadic least fixed points, based on formulae $\varphi(X, x)$ that are monotone (positive) in the monadic recursion variable X. Any such φ induces a monotone operation $F_\varphi : P \mapsto \{\, a \in \mathfrak{A} \mid \mathfrak{A} \models \varphi(P, a) \,\}$ on monadic relations $P \subseteq A$. The least fixed point of this operation over $\mathfrak{A}$, denoted as $\varphi^\infty(\mathfrak{A})$, is also the least stationary point of the monotone iteration sequence of stages $\varphi^\alpha(\mathfrak{A})$ starting from $\varphi^0(\mathfrak{A}) := \emptyset$. The least α for which $\varphi^{\alpha+1}(\mathfrak{A}) = \varphi^\alpha(\mathfrak{A})$ is called the closure ordinal for this fixed-point iteration on $\mathfrak{A}$.

For a concrete fixed-point process it may be hard to tell whether the recursion employed is crucial or whether it is spurious and can be eliminated. Indeed this question comes in two versions: (a) one can ask whether a resulting fixed point is also uniformly definable in the base logic without fixed-point recursion (a pure expressiveness issue); (b) one may also be interested to know whether the given fixed-point iteration terminates within a uniformly finitely bounded number of iterations (an algorithmic issue, concerning the dynamics of the fixed-point recursion rather than its result).

The boundedness problem $\mathsf{Bdd}(\mathcal{F}, \mathcal{C})$ for a class of formulae $\mathcal{F}$ and a class of structures $\mathcal{C}$ concerns question (b): to decide, for $\varphi \in \mathcal{F}$, whether there is a finite upper bound on its closure ordinal, uniformly across all structures $\mathfrak{A} \in \mathcal{C}$ (we call such fixed-point iterations, or φ itself, *bounded over* $\mathcal{C}$).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 67–78, 2009.

Interestingly, for first-order logic, as well as for many natural fragments, the two questions concerning eliminability of least fixed points coincide at least over the class of all structures. By a classical theorem of Barwise and Moschovakis [1], the only way that the fixed point $\varphi^\infty(\mathfrak{A})$ can be first-order definable for every $\mathfrak{A}$, is that there is some finite α for which $\varphi^\infty(\mathfrak{A}) = \varphi^\alpha(\mathfrak{A})$ for all $\mathfrak{A}$. The converse is clear from the fact that the unfolding of the iteration to any fixed finite depth α is easily mimicked in FO.

In other cases – and even for FO over other, restricted classes of structures, e.g., in finite model theory – the two problems can indeed be distinct, and of quite independent interest.

We here deal with the boundedness issue. Boundedness (even classically, over the class of all structures, and for just monadic fixed points as considered above) is undecidable for most first-order fragments of interest, e.g., [6]. Notable exceptions are monadic boundedness for positive existential formulae (datalog) [3], for modal formulae [9], and for (a restricted class of) universal formulae without equality [10].

One common feature of these decidable cases of the boundedness problem is that the fragments concerned have a kind of tree model property (not just for satisfiability in the fragment itself, but also for the fixed points and for boundedness). This is obvious for the modal fragment [9], but clearly also true for positive existential FO (derivation trees for monadic datalog programs can be turned into models of bounded tree width), and similarly also for the restricted universal fragment in [10].

Motivated by this observation, [7] has made a first significant step in an attempt to analyse the boundedness problem from the opposite perspective, varying the class of structures rather than the class of formulae. The hope is that this approach could go beyond an ad-hoc exposition of the decidability of the boundedness problem for individual syntactic fragments, and offer a unified model theoretic explanation instead. [7] shows that boundedness is decidable for *all* monadic fixed points in FO over the class of all acyclic relational structures. Technically [7] expands on modal and locality based proof ideas and reductions to the MSO theory of trees from [9,10] that also rest on the availability of a Barwise–Moschovakis equivalence. These techniques do not seem to extend to either the class of all trees (where Barwise–Moschovakis fails) or to bounded tree width (where certain simple locality criteria fail).

The present investigation offers another step forward in the alternative approach to the boundedness problem, on a methodologically very different note, and – maybe the most important novel feature – in a setting where neither locality nor Barwise–Moschovakis are available. On the one hand, the class of formulae considered is extended from first-order logic FO to monadic second-order logic MSO – a leap which greatly increases the robustness of the results w.r.t. interpretations, and hence their model theoretic impact. On the other hand, automata are crucially used and, for the purposes of the present treatment, the underlying structures are restricted to just finite word structures. We expect that this restriction can be somewhat relaxed, though. Work in progress based on automata theoretic results recently obtained by Colcombet and Löding [2] shows that our approach generalises from finite words to the case of finite trees. This extension of the present results will, via MSO interpretability in trees, then reach up at least to the finite model-theory version of the following conjecture, which has been implicit as a potential keystone to this alternative approach to boundedness:

Conjecture 1. The boundedness problem for monadic second-order logic over the class of all trees (and hence over any MSO-definable class of finite tree width) is decidable.

2 Preliminaries

We assume that the reader is familiar with basic notions of logic (see, e.g., [4] for details). Throughout the paper we assume that all vocabularies are finite and that they contain only relation symbols and constant symbols, but no function symbols. We regard free variables as constant symbols.

Let φ and ψ be formulae over a vocabulary τ containing a unary relation symbol X and a constant symbol x. As usual, the formula $\varphi[c/x]$ is obtained from φ by replacing all free occurrences of x by c. The formula $\varphi[\psi(x)/X]$ is obtained from φ by replacing all free occurrences of X, say Xc with constant symbol c, by $\psi[c/x]$. For $\alpha < \omega$, we define the formula φ^α inductively as follows:

$$\varphi^0 := \bot \quad \text{and} \quad \varphi^{\alpha+1} := \varphi[\varphi^\alpha(x)/X]\,.$$

Note that the vocabulary of φ^α is $\tau \smallsetminus \{X\}$. Suppose that $\mathfrak{A}$ is a structure of vocabulary $\tau \smallsetminus \{X, x\}$. If φ is positive in X, then φ^α defines the α-th stage of the least fixed-point induction of φ on $\mathfrak{A}$. We denote this set by $\varphi^\alpha(\mathfrak{A})$. The corresponding fixed point is $\varphi^\infty(\mathfrak{A})$.

Definition 1. (a) *Let φ be a formula over τ, positive in X, and let $\alpha < \omega$. We say that φ is* bounded by α *over a class $\mathcal{C}$ if $\varphi^\alpha(\mathfrak{A}) = \varphi^{\alpha+1}(\mathfrak{A})$, for all $\mathfrak{A} \in \mathcal{C}$. We call φ* bounded *over $\mathcal{C}$ if it is bounded by some $\alpha < \omega$.*

(b) *The* boundedness problem *for a logic L over a class $\mathcal{C}$ is the problem to decide, given a formula $\varphi \in L$, whether φ is bounded over $\mathcal{C}$.*

Lemma 1. *Let L be a logic and $\mathcal{C}$ a class of structures such that equivalence of L-formulae over $\mathcal{C}$ is decidable. The boundedness problem for L over $\mathcal{C}$ is decidable if and only if there is a computable function $f : L \to \omega$ such that, if a formula $\varphi \in L$ is bounded over $\mathcal{C}$, then it is bounded by $f(\varphi)$ over $\mathcal{C}$.*

In this paper we consider the class of all finite words over some alphabet Σ. We encode such words as structures in the usual way. Let τ_Σ be the signature consisting of a binary relation $\leq$ and unary relations P_c, for every $c \in \Sigma$. We represent a finite word $w = a_0 \dots a_{n-1} \in \Sigma^*$ as the structure whose universe $[n] := \{0, \dots, n-1\}$ consists of all positions in w and where $\leq$ is interpreted by the usual order on integers, and P_c is interpreted by the set $\{\, i \in [n] \mid a_i = c \,\}$ of all positions carrying the letter c.

We denote the *concatenation* of two words $\mathfrak{A}$ and $\mathfrak{B}$ by $\mathfrak{A}+\mathfrak{B}$. For a structure $\mathfrak{A}$ and a set $U \subseteq A$, we denote by $\mathfrak{A}_U$ the substructure induced by U. (If $\mathfrak{A}$ contains constants with value outside of U then we drop them from the vocabulary when forming $\mathfrak{A}_U$.)

We will reduce the boundedness problem to a corresponding problem for automata. A *distance automaton* is a tuple $\mathcal{A} = (\Sigma, Q, \Delta_0, \Delta_1, I, F)$, where $\mathcal{A}' = (\Sigma, Q, \Delta, I, F)$ for $\Delta = \Delta_0 \dot{\cup} \Delta_1$ is a finite nondeterministic automaton in the usual sense with alphabet Σ, state space Q, transition relation $\Delta \subseteq Q \times \Sigma \times Q$, set of initial states $I \subseteq Q$, and set of final states $F \subseteq Q$. The language $L(\mathcal{A})$ of $\mathcal{A}$ is the language of $\mathcal{A}'$ in the usual sense, and for $w \in L(\mathcal{A})$, the distance $d_{\mathcal{A}}(w)$ is the minimal number of transitions from Δ_1, the minimum ranging over all accepting runs of $\mathcal{A}'$ on w. For $w \notin L(\mathcal{A})$, we set $d_{\mathcal{A}}(w) := \infty$. As usual, we set $d_{\mathcal{A}}(L) := \{ d_{\mathcal{A}}(w) \mid w \in L \}$, for sets $L \subseteq \Sigma^*$. This definition is a slightly modified version of the one in [5].

Theorem 1 (Hashiguchi [5,8]). *Let $\mathcal{A}$ be a distance automaton with state space Q. If $d_{\mathcal{A}}(L(\mathcal{A}))$ is bounded, then it is bounded by $2^{4|Q|^3}$:*

$$\sup d_{\mathcal{A}}(L(\mathcal{A})) < \infty \quad \textit{implies} \quad \sup d_{\mathcal{A}}(L(\mathcal{A})) \leq 2^{4|Q|^3}.$$

3 Positive Types

For a vocabulary τ, we denote by $\mathrm{MSO}^n[\tau]$ the set of all MSO-formulae over τ with quantifier rank at most n. If $X \in \tau$ is a unary predicate we write $\mathrm{MSO}^n_X[\tau]$ for the subset of all formulae where the predicate X occurs only positively. $\mathrm{MSO}^n_X[\tau]$ is finite up to logical equivalence, and we will silently assume that all formulae are canonised in some way. For example, for $\Phi \subseteq \mathrm{MSO}^n_X[\tau]$ the conjunction $\bigwedge \Phi$ is always a formula from $\mathrm{MSO}^n_X[\tau]$, and it will even happen that $\bigwedge \Phi \in \Phi$. The following result carries over from $\mathrm{MSO}^n[\tau]$ to $\mathrm{MSO}^n_X[\tau]$.

Fact 1. *There exists a computable function $f : \omega \to \omega$ such that, up to logical equivalence, we have*

$$|\mathrm{MSO}^n[\tau]| \leq f\big(n + |\tau| + \mathrm{ar}(\tau)\big) .$$

Definition 2. *Let τ be a vocabulary and $X \in \tau$. The X-*positive* n-*type* of a τ-structure $\mathfrak{A}$ is the set*

$$\mathrm{tp}^n_X(\mathfrak{A}) := \{ \varphi \in \mathrm{MSO}^n_X[\tau] \mid \mathfrak{A} \models \varphi \} .$$

We write $\mathrm{Tp}^n_X[\tau]$ for the set of all X-positive n-types of τ-structures.

Lemma 2. *Let $\mathfrak{A}$ be a structure and $P \subseteq P' \subseteq A$. Then*

$$\mathrm{tp}^n_X(\mathfrak{A}, P) \subseteq \mathrm{tp}^n_X(\mathfrak{A}, P') ,$$

where X is interpreted by P and P', respectively.

Fact 2. *Let $\mathfrak{B}$ be the τ-reduct of $\mathfrak{A}$. Then* $\mathrm{tp}^n_X(\mathfrak{B}) = \mathrm{tp}^n_X(\mathfrak{A}) \cap \mathrm{MSO}^n_X[\tau]$.

Lemma 3. *For every $n < \omega$, there is a binary function $\oplus_n$ such that*

$$\mathrm{tp}^n_X(\mathfrak{A} + \mathfrak{B}) = \mathrm{tp}^n_X(\mathfrak{A}) \oplus_n \mathrm{tp}^n_X(\mathfrak{B})\,, \quad \textit{for all words } \mathfrak{A} \textit{ and } \mathfrak{B}\,.$$

Furthermore, $\oplus_n$ is monotone:

$$s \subseteq s' \quad \textit{and} \quad t \subseteq t' \quad \textit{implies} \quad s \oplus_n t \subseteq s' \oplus_n t'\,.$$

Note that, being a homomorphic image of word concatenation $+$, the operation $\oplus_n$ is associative.

4 The Main Theorem

Let us temporarily fix a formula $\varphi \in \mathrm{MSO}^n_X[\tau]$ with vocabulary $\tau = \{x, X, \leq, P_a, P_b, \dots\}$ belonging to a word structure with one constant symbol x and one additional unary predicate X. Let

$$\pi : \mathrm{Tp}^n_X[\tau] \to \mathrm{Tp}^n_X[\tau \setminus \{x\}]$$

be the canonical projection defined by $\pi(t) := t \cap \mathrm{MSO}^n_X[\tau \setminus \{x\}]$.

Note that, by our assumption on τ, there are exactly two X-positive n-types of one-letter τ-words with letter a: the one not containing Xx and the one which does contain Xx. We will at times denote these by, respectively, 0_a and 1_a. Frequently, we will omit the index a if we do not want to specify the letter.

Given a word structure $\mathfrak{A}$ of vocabulary $\tau \setminus \{X, x\}$, we consider the fixed-point induction of φ. For every $\alpha < \omega$ and every position p of $\mathfrak{A}$ we consider the type $\mathrm{tp}(\mathfrak{A}, \varphi^\alpha(\mathfrak{A}), p)$. We annotate $\mathfrak{A}$ with all these types. At each position p we write down the list of these types for all stages α. These annotations can be used to determine the fixed-point rank of all elements of $\mathfrak{A}$. A position p enters the fixed point at stage α if the α-th entry of the list is the first one containing a type t with $Xx \in t$.

We can regard the annotation as consisting of several layers, one for each stage of the induction. At a position p each change between two consecutive layers is caused by some change at some other position in the previous step. In this way we can trace back changes of the types through the various layers.

In order to determine whether the fixed-point inductions of the formula are bounded, we construct a distance automaton that recognises (approximations of) such annotations. Furthermore, the distance computed by the automaton coincides with the longest path of changes in the annotation. It follows that the automaton is bounded if and only if the fixed-point induction is bounded. Consequently, we can solve the boundedness problem for φ with the help of Theorem 1.

Let us start by precisely defining the annotations we use. A local stage annotation at a position above a fixed letter of $\mathfrak{A}$ captures the flow of information that is relevant for stage updates in the fixed-point induction at this letter and at some stage.

Definition 3. (a) *A* local stage annotation *is a* 6*-tuple*

$$\gamma = \begin{pmatrix} {}^{<}t & t^{\wedge} & t^{>} \\ {}_{>}t & {}_{\wedge}t & t_{<} \end{pmatrix}$$

of types where

- ${}_{\wedge}t,\ t^{\wedge} \in \{0_a, 1_a\}$ *with* ${}_{\wedge}t \subseteq t^{\wedge}$*, for some letter* $a \in \Sigma$*,*
- ${}_{>}t,\ {}^{<}t,\ t^{>},\ t_{<} \in \mathrm{Tp}^n_X[\tau \smallsetminus \{x\}]$,
- ${}^{<}t = \pi({}_{\wedge}t \oplus t_{<})$ *and* $t^{>} = \pi({}_{>}t \oplus {}_{\wedge}t)$,
- $Xx \in t^{\wedge}$ *iff* $\varphi \in {}_{>}t \oplus {}_{\wedge}t \oplus t_{<}$.

We say that γ *is an annotation of* a*, for the letter* a *in the first clause.*

(b) *Let* $\mathfrak{A}$ *be the word structure corresponding to* $a_0 \dots a_{\ell-1} \in \Sigma^*$*. For* $\alpha < \omega$*, we denote the expansion of* $\mathfrak{A}$ *by the* α*-th stage of* φ *by* $\mathfrak{A}^\alpha := (\mathfrak{A}, \varphi^\alpha(\mathfrak{A}))$*.*

The annotated word $\mathrm{An}(\mathfrak{A})$ *is a word* $b_0 \dots b_{\ell-1}$ *where the* p*-th letter* b_p *is the sequence of local stage annotations of* a_p *obtained by the removal of duplicates from the sequence* $(\gamma^\alpha)_{\alpha<\omega}$ *with*

$$\gamma^\alpha := \begin{pmatrix} \mathrm{tp}^n_X(\mathfrak{A}^\alpha_{[p,\ell)}) & \mathrm{tp}^n_X(\mathfrak{A}^{\alpha+1}_{\{p\}}, p) & \mathrm{tp}^n_X(\mathfrak{A}^\alpha_{[0,p]}) \\ \mathrm{tp}^n_X(\mathfrak{A}^\alpha_{[0,p)}) & \mathrm{tp}^n_X(\mathfrak{A}^\alpha_{\{p\}}, p) & \mathrm{tp}^n_X(\mathfrak{A}^\alpha_{(p,\ell)}) \end{pmatrix} .$$

Here, $\mathfrak{A}^\alpha_U$ *denotes* $(\mathfrak{A}^\alpha)_U$*, not* $(\mathfrak{A}_U)^\alpha$*.*

The components of an annotation γ are called *incoming from the left, outgoing to the left,* and so on. They are denoted by ${}_{>}\gamma, {}^{<}\gamma, \dots$. We also speak of the ${}_{>}\bullet$-component of γ, etc.

Example 1. Consider the formula

$$\varphi(X, x) := \forall y[y < x \to Xy] \vee \forall y[y > x \to Xy] .$$

Figure 1 shows (the first 4 elements of) the real annotation of a word of length at least 9. Here,

- λ denotes the type of the empty word,
- 0 denotes any type not containing the formula $\exists y Xy$,
- 1 denotes any type containing the formula $\forall y Xy$, and
- 01 denotes any type containing $\exists y Xy$, but not $\forall y Xy$.

Below we will construct an automaton that, given a word $\mathfrak{A}$ guesses potential annotations for $\mathfrak{A}$ and computes bounds on the length of the fixed-point induction of φ on $\mathfrak{A}$. Unfortunately, the real annotations $\mathrm{An}(\mathfrak{A})$ cannot be recognised by automata. For instance, in the above example the real annotations of words of even length are of the form ux^ny^nv where y^nv is the 'mirror image' of ux^n. This language is not regular.

So we have to work with approximations. Let us see what such approximations look like.

		1 1 1 1 1 1	1 1 1 1 1 1	
	1 1 1 1 1 1	01 1 1 1 1 01	01 1 1 1 1 01	
1 1 1 λ 1 1	01 1 1 1 1 01	01 1 01 1 0 01	01 1 01 1 0 01	...
01 1 1 λ 1 01	01 1 01 1 0 01	01 0 01 01 0 01	01 0 01 01 0 01	
0 1 0 λ 0 0	0 0 0 0 0 0	0 0 0 0 0 0	0 0 0 0 0 0	

Fig. 1. Annotation for $\varphi(X, x) := \forall y[y < x \rightarrow Xy] \vee \forall y[y > x \rightarrow Xy]$

Definition 4. (a) *We extend the order $\subseteq$ on X-positive n-types to local stage annotations by requiring that $\subseteq$ holds component-wise. A* history *(at a) is a strictly increasing sequence $h = (h^0 \subsetneq \cdots \subsetneq h^m)$ of local stage annotations (at a) such that*

- $_\wedge(h^0) = 0_a$*,*
- $_\wedge(h^{i+1}) = (h^i)^\wedge$*, for all $i < m$, and*
- $(h^m)^\wedge = 1_a$ *implies* $_\wedge(h^m) = 1_a$.

Let Σ_τ denote the set of all histories with $a \in \Sigma$. An annotated word *is a word over Σ_τ.*

(b) *We say that an annotated word is* consistent, *if it satisfies the following conditions.*

(1) *If h_2 is the immediate successor of h_1, then the projections of h_1 to the components $\bullet^>$ and $\bullet_<$ coincide*[1] *with the projections of h_2 to the components $_>\bullet$ and $^<\bullet$, respectively.*
(2) *For the first letter: the $_>\bullet$ components in its history are all equal to* $\mathrm{tp}^n_X(\lambda)$*, where λ is the empty word.*
(3) *Similarly, for the last letter: the $\bullet_<$ components in its history are all equal to* $\mathrm{tp}^n_X(\lambda)$*.*

Clearly, $\mathrm{An}(\mathfrak{A})$ is a consistent annotated word. Furthermore, consistency of annotated words can be checked by an automaton since all conditions are strictly local. The main part of our work will consist in computing bounds on the real fixed-point rank of an element from such a word.

For an annotated word $\mathfrak{A}$, we index the individual type annotations by triples (p, i, j) where p is a position in $\mathfrak{A}$, i is an index for the history at position p, and j specifies the component in the local stage annotation. We denote the type specified in this way by $t_{p,i,j}$, or by $({}_>t)_{p,i}$, $({}^<t)_{p,i}$ for a concrete component $j = {}_>\bullet, {}^<\bullet$, etc.

[1] This is coincidence as a set, i.e., with duplicates removed.

When considering the annotated word encoding the fixed-point induction of φ, the indices of particular interest are those at which type changes occur. The fixed point is reached as soon as no such changes occur anymore.

Definition 5. *An index $I = (p, i, j)$ is* relevant *if either*

$$i > 0 \text{ and } t_{p,i,j} \neq t_{p,i-1,j}\,, \quad \text{or} \quad i = 0\,,\ j = \bullet^\wedge\,,\ \text{and } Xx \in t_{p,i,j}\,.$$

In the latter case, we call I initially relevant.

During the fixed-point induction changes at one index trigger changes at other indices in the next stage. The following definition formalises this dependency. We introduce three notions of dependency between indices. We have *direct dependencies,* where a change at one index immediately leads to a change at another one, and we have what we call *lower* and *upper dependencies,* intuitively associated with the temporal sequence of events. However, due to the lack of synchronisation between levels of adjacent histories (which in turn comes from the deletion of duplicates in each history), this temporal intuition is not directly available for dependencies linking adjacent histories. Some of the real stage dependencies can only be reconstructed globally, which will eventually give us the required bounds on ranks.

Definition 6. *Let $I = (p, i, j)$ and $I' = (p', i', j')$ be two relevant indices. We say that I* directly depends *on I' if I is not initially relevant and one of the following cases occurs:*

$$\begin{array}{llll}
j = {}_>\bullet\,, & j' = \bullet^>\,, & p' = p - 1\,, & \text{and} \quad t_I = t_{I'}\,;\\
j = \bullet_<\,, & j' = {}^<\bullet\,, & p' = p + 1\,, & \text{and} \quad t_I = t_{I'}\,;\\
j = {}_\wedge\bullet\,, & j' = \bullet^\wedge\,, & p' = p\,, & \text{and} \quad i' = i - 1\,;\\
j = \bullet^\wedge\,, & j' \in \{{}_>\bullet, {}_\wedge\bullet, \bullet_<\}\,, & p' = p\,, & \text{and} \quad i' = i\,;\\
j = {}^<\bullet\,, & j' \in \{\bullet_<, {}_\wedge\bullet\}\,, & p' = p\,, & \text{and} \quad i' = i\,;\\
j = \bullet^>\,, & j' \in \{{}_>\bullet, {}_\wedge\bullet\}\,, & p' = p\,, & \text{and} \quad i' = i\,.
\end{array}$$

A direct dependency of some index $(p, i, {}_\wedge\bullet)$ on $(p, i-1, \bullet^\wedge)$ is called a jump.

Relaxing the equality requirement $i' = i$ to either $i' \leq i$ or to $i' \geq i$ in each of the last three clauses (thus also allowing upward or downward steps within the same history in those cases), we obtain dependencies from below *or* from above.

Note that the last three forms of direct dependencies go from outgoing to incoming indices within the same local annotation.

Furthermore, I directly depends on I' if and only if it depends on I' both from below and from above.

Also note that the first two clauses of (direct) dependency are the only dependencies between distinct (namely adjacent) histories. In these there is no condition on i, i', corresponding to the lack of synchronisation discussed above.

Finally note that in the case of a jump, i.e., a (direct) dependency of $(p, i, {}_\wedge\bullet)$ on $(p, i-1, \bullet^\wedge)$, we have

$$({}_\wedge t)_{p,i} = (t^\wedge)_{p,i-1} = 1 \quad \text{and} \quad ({}_\wedge t)_{p,i-1} = 0\,.$$

In particular, at every position p there can be at most one jump.

Lemma 4. *Let I be a relevant index in a consistent annotated word. Either I is initially relevant, or there is some relevant index on which I depends directly.*

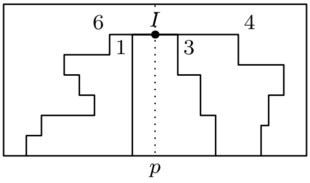
We can form a digraph consisting of all relevant indices where there is an edge from I to I' if I depends on I' from below. We call this digraph the *lower dependency graph.* Similarly, we can define the *upper dependency graph* by using dependencies from above.

Lemma 5. *The digraph of lower dependencies in a consistent annotated word is acyclic.*

A path in the dependency digraph is called *grounded* if it ends in an initially relevant index. The *rank* of a path is the number of jumps it contains.

Due to acyclicity and finiteness, all maximal paths in the lower dependency graph are grounded. The same is true also in the direct dependency graph, since every relevant index is either initial or it directly depends on some other relevant index by Lemma 4. For the upper dependency graph, which may have cycles, we can only say that every maximal cycle-free path must be grounded (and there are always such, namely in particular paths w.r.t. direct dependencies).

Let I be a relevant index and $\alpha < \omega$. We say that α is *a lower rank* of I if in the lower dependency graph there is some grounded path from I of rank α. Similarly, we define *upper ranks* of I as the ranks of grounded cycle-free paths in the upper dependency graph. Note that I can have several different lower and upper ranks, but at least one of each kind (due to the existence of grounded paths w.r.t. direct dependencies, Lemma 4).



We now fix a consistent annotated word $\mathfrak{A}$ over the underlying Σ-word $\mathfrak{B}$. Let ℓ be their length. As above, we write $\mathfrak{B}^\alpha := (\mathfrak{B}, \varphi^\alpha(\mathfrak{B}))$ for the expansion of $\mathfrak{B}$ by the α-th stage of the fixed-point induction.

We say that α *satisfies* an outgoing index $I = (p, i, j)$ if

$$\begin{array}{llll} & j = {}^{<}\bullet & \text{and} & \mathfrak{B}^\alpha_{[p,\ell)} \models t_I\,, \\ \text{or} & j = \bullet^{\wedge} & \text{and} & (\mathfrak{B}^{\alpha+1}_{\{p\}}, p) \models t_I\,, \\ \text{or} & j = \bullet^{>} & \text{and} & \mathfrak{B}^\alpha_{[0,p]} \models t_I\,. \end{array}$$

Note that $\mathfrak{B}^\alpha_{[p,\ell)} \models t_I$ just means that

$$t_I \subseteq \operatorname{tp}^n_X(\mathfrak{B}^\alpha_{[p,\ell)})\,.$$

Reverting the inclusion, we say that $I = (p, i, {}^{<}\bullet)$ *confines* α if

$$\mathrm{tp}^n_X(\mathfrak{B}^\alpha_{[p,\ell)}) \subseteq t_I \,.$$

For the other outgoing cases, we define confinement analogously.

The next lemma relates the real fixed-point induction of φ on $\mathfrak{B}$ to the given annotation $\mathfrak{A}$, through confinement. In particular, the top level of the annotation confines all stages of the real fixed point.

Lemma 6. *Let p be a position and m_p the length of the history at position p.*

(a) $\mathrm{tp}^n_X(\mathfrak{B}^0_{[0,p)}) = ({}_{>}t)_{p,0}$ *and* $\mathrm{tp}^n_X(\mathfrak{B}^0_{(p,\ell)}) = (t_{<})_{p,0}$.

(b) *For every $\alpha < \omega$, we have*

$$\begin{aligned} \mathrm{tp}^n_X(\mathfrak{B}^\alpha_{[0,p)}) &\subseteq ({}_{>}t)_{p,m_p} \,, \\ \mathrm{tp}^n_X(\mathfrak{B}^\alpha_{\{p\}}, p) &\subseteq ({}_{\wedge}t)_{p,m_p} \,, \\ \mathrm{tp}^n_X(\mathfrak{B}^\alpha_{(p,\ell)}) &\subseteq (t_{<})_{p,m_p} \,. \end{aligned}$$

We can use the preceding lemma to show that the lower and upper ranks provide bounds for the real rank of an element.

Lemma 7. *Let $I = (p, i, j)$ be a relevant outgoing index and $\alpha < \omega$.*

(a) *If $\alpha \geq \alpha'$ for all lower ranks α' of I, then α satisfies I.*

(b) *If I is not initially relevant and all upper ranks of I are larger than α, then $(p, i-1, j)$ confines α.*

We call a position p *active* if there is some i such that $(p, i, \bullet^\wedge)$ is relevant; in this case, the corresponding i is unique. We may thus define the set of *upper ranks* of an active position p as the set of upper ranks of the relevant index of the form $(p, i, \bullet^\wedge)$ at p. Recall that an upper rank of a relevant index is any rank of a grounded cycle-free upper dependency path.

Lemma 8. *Let p be a position.*

(a) *If $p \in \varphi^\infty(\mathfrak{B})$, then p is active.*

(b) *If p is active, then $p \in \varphi^\infty(\mathfrak{B})$.*

(c) *If $p \in \varphi^\alpha(\mathfrak{B})$ (and hence p is active by* (a)*), then some upper rank of p is at most α.*

A *proposal* is a pair $(\mathfrak{A}, p)$ where $\mathfrak{A}$ is a consistent annotated word and p is an active position in $\mathfrak{A}$. In order to treat proposals as words over some alphabet one can extend annotated letters with a mark for the special position p.

Lemma 9. *There exists a computable function $g : \omega \to \omega$ such that, for every formula φ, we can effectively construct a distance automaton $\mathcal{A}$ with at most $g(|\varphi|)$ states such that*

(a) *$L(\mathcal{A})$ is the set of proposals;*

(b) *if* $(\mathfrak{A}, p)$ *is a proposal then* $d_{\mathcal{A}}(\mathfrak{A}, p)$ *is the minimum over all upper ranks of* p*.*

Next, let us consider annotated words that arise from the actual fixed-point induction. Recall that $\mathrm{An}(\mathfrak{B})$ is the word whose p-th letter is the history (h^i) at position p. The removal of duplicates in the definition of a history induces a non-decreasing mapping $i_{\mathfrak{B},p} : \omega \to \omega$ from stages to history entries such that, for example, ${}_{>}(h^{i_{\mathfrak{B},p}(\alpha)}) = \mathrm{tp}^{\alpha}_{X}(\mathfrak{B}_{[0,p)})$. For $I = (p, i, j)$, we set

$$\alpha_{\mathfrak{B}}(I) := \alpha_{\mathfrak{B},p}(i) := \min \{ \alpha < \omega \mid i = i_{\mathfrak{B},p}(\alpha) \} .$$

Lemma 10. *Let* $\mathfrak{B}$ *be a word and let* I *be a relevant index in* $\mathrm{An}(\mathfrak{B})$*. Then each upper rank of* I *is bounded from below by* $\alpha_{\mathfrak{B}}(I)$*.*

Theorem 2. *The boundedness problem for* MSO *over the class of all finite words is decidable.*

Proof. Let $\varphi \in \mathrm{MSO}$ be positive in X and let g be the function from Lemma 9. We exploit Lemma 1 and claim that, over finite words, if φ is bounded then it is bounded by $N := 2^{4g(|\varphi|)^3} + 1$.

Assume that φ is bounded over finite words, say by N'. For every proposal $(\mathfrak{A}, p)$, it follows from Lemma 8(b), that $p \in \varphi^{\infty}(\mathfrak{A})$. Hence, $p \in \varphi^{N'}(\mathfrak{A})$. Lemma 8(c) then implies that some rank of p is at most N'. Let $\mathcal{A}$ be the distance automaton from Lemma 9. Then we have $d_{\mathcal{A}}(L(\mathcal{A})) \leq N' < \infty$. Therefore, Theorem 1 implies that $d_{\mathcal{A}}(L(\mathcal{A})) \leq N - 1$. Consequently, for all proposals $(\mathfrak{A}, p)$, some rank of p is at most $N - 1$. In particular, this holds if $\mathfrak{A} = \mathrm{An}(\mathfrak{B})$, for some word structure $\mathfrak{B}$. By Lemma 10, p enters the fixed-point induction not later than stage N. As $\mathfrak{B}$ and p were arbitrary, it follows that φ is bounded over words by N. □

5 Extensions

Having obtained the decidability for the boundedness problem over the class of all finite words we can use model theoretic interpretations to obtain further decidability results.

Theorem 3. *For all* k*, the boundedness problem for* MSO *over the class of all finite structures of path width at most* k *is decidable.*

Example 2. Let C_n be the class of all unranked trees (T, E, S) of height at most n where E is the successor relation and S is the next sibling relation. This class has path width at most $2n$. By the theorem, it follows that the boundedness problem for monadic second-order formulae over C_n is decidable.

Using similar techniques, one can extend the theorem to MSO-axiomatisable subclasses, to guarded second-order logic GSO, and to simultaneous fixed points. If we could show that the boundedness problem is also decidable for the class of all (finite) trees, then it would follow in the same way that the problem is decidable for every GSO-axiomatisable class of (finite) structures of bounded tree width.

References

1. Barwise, J., Moschovakis, Y.N.: Global inductive definability. The Journal of Symbolic Logic 43, 521–534 (1978)
2. Colcombet, T., Löding, C.: The nesting-depth of disjunctive μ-calculus for tree languages and the limitedness problem. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 416–430. Springer, Heidelberg (2008)
3. Cosmadakis, S.S., Gaifman, H., Kanellakis, P.C., Vardi, M.Y.: Decidable optimization problems for database logic programs. In: Proc. 20th Annual ACM Symposium on Theory of Computing, STOC 1988, pp. 477–490 (1988)
4. Ebbinghaus, H.-D., Flum, J.: Finite Model Theory. Springer, Heidelberg (1995)
5. Hashiguchi, K.: Improved limitedness theorems on finite automata with distance functions. Theoretical Computer Science 72, 72–78 (1990)
6. Hillebrand, G., Kanellakis, P., Mairson, H., Vardi, M.: Undecidable boundedness problems for datalog programs. The Journal of Logic Programming 25, 163–190 (1995)
7. Kreutzer, S., Otto, M., Schweikardt, N.: Boundedness of monadic FO over acyclic structures. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 571–582. Springer, Heidelberg (2007)
8. Leung, H., Podolskiy, V.: The limitedness problem on distance automata: Hashiguchi's method revisited. Theoretical Computer Science 310, 147–158 (2004)
9. Otto, M.: Eliminating recursion in the μ-calculus. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 531–540. Springer, Heidelberg (1999)
10. Otto, M.: The boundedness problem for monadic universal first-order logic. In: Proc. 21th IEEE Symposium on Logic in Computer Science, LICS 2006, pp. 37–46 (2006)

Semilinear Program Feasibility

Manuel Bodirsky[1], Peter Jonsson[2], and Timo von Oertzen[3]

[1] CNRS/LIX, École Polytechnique, 91128 Palaiseau, France
bodirsky@lix.polytechnique.fr

[2] Department of Computer and System Science, Linköpings Universitet
SE-581 83 Linköping, Sweden
petej@ida.liu.se

[3] Max-Planck-Institute for Human Development, Königin-Luise-Strasse 5
14195 Berlin
vonoertzen@mpib-berlin.mpg.de

Abstract. We study logical techniques for deciding the computational complexity of infinite-domain constraint satisfaction problems (CSPs). For the fundamental algebraic structure $\Gamma = (\mathbb{R}; L_1, L_2, \dots)$ where $\mathbb{R}$ are the real numbers and $L_1, L_2, \dots$ is an enumeration of all linear relations with rational coefficients, we prove that a semilinear relation R (i.e., a relation that is first-order definable with linear inequalities) either has a quantifier-free Horn definition in Γ or the CSP for $(\mathbb{R}; R, L_1, L_2, \dots)$ is NP-hard. The result implies a complexity dichotomy for all constraint languages that are first-order expansions of Γ: the corresponding CSPs are either in P or are NP-complete depending on the choice of allowed relations. We apply this result to two concrete examples (generalised linear programming and metric temporal reasoning) and obtain full complexity dichotomies in both cases.

1 Introduction

Let $\Gamma = (D; R_1, R_2, \dots)$ be a relational structure[1] over the set D. The *constraint satisfaction problem for* Γ (CSP(Γ) in short) is the computational problem to decide whether a given primitive positive sentence Φ involving relation symbols for the relations in Γ is true in Γ. A first-order formula is called *primitive positive* (pp) if it is of the form

$$\exists x_1, \dots, x_n . \psi_1 \wedge \dots \wedge \psi_m$$

where ψ_i are atomic formulas, i.e., formulas of the form $x = y$ or $R(x_{i_1}, \dots, x_{i_k})$ with R the relation symbol for a k-ary relation from Γ. We call such a formula a *pp-formula.* The conjuncts in a pp-formula Φ are also called the *constraints* of Φ, and to emphasise the connection between the structure Γ and the constraint satisfaction problem, we typically refer to Γ as a *constraint language.*

[1] Our terminology is standard; all notions that are not introduced in the paper can be found in standard text books, e.g., in [11].

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 79–90, 2009.

By choosing an appropriate constraint language Γ, many computational problems that have been studied in the literature can be formulated as CSP(Γ) (see e.g. [2, 4]). It often turns out that the structure Γ can be chosen to be ω-categorical, i.e., the set of all first-order sentences that is true in the structure has a unique countable model (up to isomorphism). The linear order of the rationals $(\mathbb{Q}, <)$ is a well-known example of an ω-categorical structure. Every finite domain CSP can be formulated within an ω-categorical structure. The condition of ω-categoricity is interesting for constraint satisfaction because the so-called *universal-algebraic approach*, which is intensively studied for finite constraint languages, applies—at least in principle—also for ω-categorical structures (see e.g. [3] for an application of the universal-algebraic approach to infinite-domain CSPs). However, many interesting and common structures are not ω-categorical. In this paper we demonstrate that methods based on logical definability can sometimes be used for studying the complexity of such languages.

Our prime example will be a class of constraint languages that contains the constraint language of *linear program feasibility*, i.e., the problem to decide whether a given system of linear inequalities with rational coefficients has a real solution. Let LI be an enumeration of the linear relations $L(x_1, \ldots, x_n)$ defined by inequalities of the form $c_1x_1 + \ldots + c_nx_n \leq c_0$ or equalities of the form $c_1x_1 + \ldots + c_nx_n = c_0$ where $n > 0$ and $c_0, c_1, \ldots, c_n \in \mathbb{Q}$.

A relation is called *semilinear* if it has a first-order definition in the structure $(\mathbb{R}; LI)$. We are mainly concerned with constraint languages that are expansions of $(\mathbb{R}; LI)$ by semilinear relations. The set of semilinear relations is a rich set. For example, every relation $R \subseteq X^k$ where X is a finite subset of $\mathbb{Q}$ is semilinear; thus, every finitary relation on a finite set can be viewed as a semilinear relation. Piecewise linear functions constitute another example: a function $f : \mathbb{R} \rightarrow \mathbb{R}$ is piecewise linear if $\mathbb{R}$ can be partitioned into finitely many subintervals $I_1, \ldots, I_n$ such that on each subinterval I_j, f is a linear function. A well-known concrete example is the absolute value function $| \cdot |$. If every interval $I_j = [a_j, b_j]$ satisfies $a_j, b_j \in \mathbb{Q}$, then relations like $\{(x, f(x)) \mid x \in \mathbb{R}\}$ and $\{(x, y, z) \in \mathbb{R}^3 \mid x = f(y - z)\}$ are semilinear.

We prove the following result.

Theorem 1. *Let $\Gamma = (\mathbb{R}; LI, R_1, R_2, \ldots)$ be a constraint language such that $R_1, R_2, \ldots$ are semilinear relations. Then, either each R_i has a quantifier-free Horn definition in $(\mathbb{R}; LI)$ and CSP(Γ) is in P, or CSP(Γ) is NP-complete.*

A first-order formula in conjunctive normal form is *Horn* if and only if each clause contains at most one positive literal. Throughout this paper, the negative literals are of the kind $\neg(c_1x_1 + \cdots + c_nx_n = c_0)$ and all other literals are considered to be positive. We typically write $e_1 \neq e_2$ instead of literals of the form $\neg(e_1 = e_2)$, and $e_1 > e_2$ instead of literals of the form $\neg(e_1 \leq e_2)$.

Thus, Theorem 1 tells us that the borderline between easy and hard problems can be precisely characterised in terms of logical definability. We use Theorem 1 for proving complete complexity dichotomies for two additional problems:

Generalised linear programming. Linear programming can be viewed as optimising a linear function over the feasible points of an instance of CSP($\mathbb{R}; LI$). This view suggests a generalisation: optimise a linear function over the feasible points of an instance of CSP(Γ) where Γ is a first-order expansion of ($\mathbb{R}; LI$). We completely classify the complexity of this problem and present an algorithm for the tractable cases.

Temporal reasoning with metric constraints. A *temporal constraint language* Γ is a structure $(\mathbb{R}; R_1, R_2, \ldots)$ with a first-order definition in $(\mathbb{R}; <)$. Many computational problems in artificial intelligence and scheduling can be modelled as constraint satisfaction problems for temporal constraint languages. Often, these languages are extended with some mechanism for expressing *metric* time, i.e., the ability to assign numerical values to variables and performing some kind of arithmetic calculations [6]. We study metric temporal languages Γ that satisfy the following restrictions: (1) All relations in Γ are semilinear, (2) Γ contains the relation $\leq$, (3) Γ contains addition $x = y + z$, and (4) Γ contains the constant 1. It has been observed that almost every polynomial-time solvable metric temporal reasoning problem is a subclass of the so-called Horn-DLR class [13]. Our result shows that this is not a coincidence: whenever a metric temporal language Γ satisfies conditions (1)-(4) and is not a subclass of Horn-DLR, then CSP(Γ) is NP-hard.

2 Preliminaries

We will be working with infinite constraint languages so it is necessary to decide how to represent relations. We represent semilinear relations by quantifier-free first-order formulas over LI in conjunctive normal form. The relations from LI are represented by their rational coefficients, which are quotients of integers written in binary. Lemma 2 will justify that we restrict ourselves to quantifier-free formulas. This representation allows us, for instance, to easily (and in polynomial time) check whether a given rational tuple is a member of a relation or not.

We say that a relation $R(x_1, \ldots, x_k)$ is *pp-definable* in Γ if there exists a quantifier pp-formula ϕ over Γ such that $(x_1, \ldots, x_n) \in R$ iff $\phi(x_1, \ldots, x_n)$ holds in Γ. The following simple but important result explains the importance of pp-definability for the constraint satisfaction problem.

Lemma 1 (Jeavons et al. [12]). *Let $\Gamma = (D; R_1, R_2, \ldots)$ be a relational structure, and let R be pp-definable over Γ. Then CSP$(D; R, R_1, R_2, \ldots)$ is polynomial-time equivalent to CSP(Γ).*

Lemma 1 will be used extensively in the sequel and we will not make explicit references to it. We say that a relation R is *quantifier-free Horn definable* over $(\mathbb{R}; LI)$ if there exists a quantifier-free Horn formula ϕ such that $R(x_1, \ldots, x_n) \equiv \phi(x_1, \ldots, x_n)$ in $(\mathbb{R}; LI)$. It is important to note that Lemma 1 does not hold in general if we replace 'primitive positive definable' with 'quantifier-free Horn definable'. The relations that are quantifier-free Horn definable in $(\mathbb{R}; LI)$ have been given various names in the literature; *Horn-DLRs* and *Horn constraints* are two examples [13,14].

Theorem 2. *Let Γ be a structure whose relations are quantifier-free Horn definable in $(\mathbb{R}; LI)$. Then CSP(Γ) is in P, and for every satisfiable instance Φ of CSP(Γ) over k variables, the solution set S to Φ satisfies*

$$S_* = \{x \in \mathbb{R}^k \mid Ax \geq \alpha, Bx \neq \beta\} \subseteq S \subseteq \{x \in \mathbb{R}^k \mid Ax \geq \alpha\} = S^*$$

where S_* and S^* have the same dimension in $\mathbb{R}^k$. Furthermore, the matrix A and the vector α can be computed in polynomial time.

Proof. The complexity result is proved in, for instance, [5,13,14]. The result concerning the solution set follows from the algorithm presented in [13]; furthermore, this algorithm explicitly constructs A and α. □

A structure Γ admits *quantifier elimination* if every first-order formula is over Γ equivalent to a quantifier free formula; see [15] for an introduction to this concept. The following result has been proved by, for instance, Ferrante and Rackoff [7].

Lemma 2. $(\mathbb{R}; LI)$ *admits quantifier elimination.*

3 First-Order Expansions of Linear Program Feasibility

We will now prove Theorem 1. Let $R(x_1, \ldots, x_k)$ be a relation over $\mathbb{R}$. The relation R is *convex* if it defines a convex subset of $\mathbb{R}^k$, and R *excludes an interval* if there are $p, q \in R$ and reals $0 < \delta_1 < \delta_2 < 1$ such that $p + (q - p)y \notin R$ whenever $\delta_1 \leq y \leq \delta_2$. Note that we can assume that δ_1, δ_2 are rational numbers.

If a unary relation is 'far' from being convex in the sense that it excludes an interval, then it is useful for proving NP-hardness via reductions from CSP($\{0,1\}$, $R_{1/3}$) where $R_{1/3} = \{(1,0,0), (0,1,0), (0,0,1)\}$. This problem is equivalent to the NP-complete variant of One-In-Three 3Sat where no clause contains a negated literal [9, LO4].

Lemma 3. *Let $T \subseteq \mathbb{R}$ be a unary relation. If T excludes an interval, then CSP$(\mathbb{R}; LI, T)$ is NP-hard.*

Proof. We know that there are points $p, q \in T$ and rational numbers $0 < \delta_1 < \delta_2 < 1$ such that $p + (q - p)y \notin T$ whenever $\delta_1 \leq y \leq \delta_2$. Assume without loss of generality that $\delta_1 = 1/5$, $\delta_2 = 4/5$ (otherwise, choose $p, q \in T$ appropriately). Define

$$U(y) \quad \equiv \quad \exists z.\, z = p + (q - p)y \;\wedge\; T(z) \;\wedge\; 0 \leq y \leq 1.$$

Observe that U is pp-definable in $(\mathbb{R}; LI, T)$. We can now show NP-hardness by a polynomial-time reduction from CSP($\{0,1\}, R_{1/3}$). Let ϕ be an arbitrary instance of this problem. For each variable v appearing in ϕ, introduce the constraint $U(v)$. For each constraint $R_{1/3}(v_i, v_j, v_k)$ in ϕ, introduce the constraints $v_i + v_j + v_k \geq 3/5, v_i + v_j + v_k \leq 8/5$. One can see that the resulting instance has a solution if and only if ϕ has a solution. □

We continue by presenting a well-known result in convex geometry. It is one of many versions of the so-called *Hahn-Banach separation theorem* which is a consequence of the Hahn-Banach theorem in functional analysis [10,1]. Given two non-empty subsets A, B of $\mathbb{R}^k$, we say that a hyperplane H *separates* A and B if A is in one and B is in the other of the two closed halfspaces determined by H.

Theorem 3. *Let A and B be nonempty, convex, and disjoint subsets of $\mathbb{R}^k$. Then, there is a hyperplane separating A and B.*

The *dimension* of a subset S of $\mathbb{R}^n$ is the largest integer d such that S contains a d-dimensional sphere (of some radius $\epsilon > 0$).

Corollary 1. *Let A and B be nonempty, convex, and disjoint subsets of $\mathbb{R}^k$. If A is open and has dimension k, then there exists a separating hyperplane H such that $A \cap H = \emptyset$.*

Proof. By Theorem 3, there exists a hyperplane H that separates A and B. Since A has full dimension, it cannot be the case that $A \subseteq H$. If $A \cap H \neq \emptyset$, then H contains some interior points of A but not all of them. The fact that A is open implies that none of the closed halfspaces determined by H contains all of A. This contradicts that H separates A and B. □

We can now prove the main lemma.

Lemma 4. *Let R be a semilinear relation of arity k. Then, either*

1. *a unary relation U that excludes an interval can be pp-defined in $(\mathbb{R}; LI, R)$; or*
2. *R is quantifier-free Horn definable in $(\mathbb{R}; LI)$.*

Proof. Suppose that R cannot be quantifier-free Horn defined in $(\mathbb{R}; LI)$. Choose a quantifier-free CNF formula $\phi = C_1 \wedge \ldots \wedge C_n$ that defines R (such a formula exists by Lemma 2). Without loss of generality, choose ϕ such that

1. ϕ contains no literals of the type $a^T x < b$ or $a^T x = b$. These can be removed by using the following equivalences:

$$(a^T x < b \vee C_1 \vee \ldots \vee C_m) \Leftrightarrow$$

$$(a^T x \leq b \vee C_1 \vee \ldots \vee C_m) \wedge (a^T x \neq b \vee C_1 \vee \ldots \vee C_m)$$

and

$$(a^T x = b \vee C_1 \vee \ldots \vee C_m) \Leftrightarrow$$

$$(a^T x \leq b \vee C_1 \vee \ldots \vee C_m) \wedge (a^T x \geq b \vee C_1 \vee \ldots \vee C_m)$$

2. ϕ contains the minimum number of non-Horn clauses (among the formulas satisfying 1.)

Now, consider a non-Horn clause

$$C = (P_1 \vee \ldots \vee P_m \vee N_1 \vee \ldots \vee N_p)$$

in ϕ where $P_1, \ldots, P_m$ are positive literals and $N_1, \ldots, N_p$ are negative literals. Let $P = \{x \in \mathbb{R}^k \mid x \text{ satisfies } (P_1 \vee \ldots \vee P_m)\}$ and note that P is a closed k-dimensional subset of $\mathbb{R}^k$. We can without loss of generality assume that $P \neq \mathbb{R}^k$ since, otherwise, we could remove the clause and obtain an equivalent formula with a smaller number of non-Horn clauses.

Define the formula $\psi = \phi \setminus C$, let S' be the set of points that satisfy ψ, and let $S = S' \cap P$. For any set $X \subseteq \mathbb{R}^k$, we define the *complement* X^c such that $X^c = \mathbb{R}^k \setminus X$. Note that P^c is a nonempty open set and it is convex (since it is an intersection of convex sets). Furthermore, the dimension of P^c is k since $P^c = (\bigcup_{i=1}^m P_i)^c = \bigcap_{i=1}^m P_i^c$ where P_i^c, $1 \leq i \leq m$, is an open halfspace with dimension k. By combining these three properties (openness, full dimension, and convexity), the following is easy to see:

(*) Let $p, q \in S$ and let L be the line $\{py + q(1-y) \mid 0 \leq y \leq 1\}$. If $L \cap P^c \neq \emptyset$, then the line $L \cap P^c$ has non-zero length.

We consider three different cases:

1. $S = \emptyset$. Then, every positive literal can be removed from C which leads to a contradiction since the number of non-Horn clauses decreases.
2. $S \neq \emptyset$ and there are points $p = (p_1, \ldots, p_k), q = (q_1, \ldots, q_k) \in S$ such that the line $L = \{py + q(1-y) \mid 0 \leq y \leq 1\}$ satisfies $L \cap P^c \neq \emptyset$. Define the unary relation U such that

$$\begin{aligned} U(y) \equiv \exists z_1, \ldots, z_k. \quad & \bigwedge_{i=1}^{k} z_i = p_i y + q_i(1-y) \quad \wedge \\ & \bigwedge_{l \in \{N_1, \ldots, N_p\}} \neg l \quad \wedge \quad R(z_1, \ldots, z_k) \quad \wedge \quad 0 \leq y \leq 1 \end{aligned}$$

   Note that U is pp-definable in $(\mathbb{R}; LI, R)$. We see that U is nonempty by the choice of p and q. By (*), $L \cap P^c$ has non-zero length so we can chose $0 < a < b < 1$ such that $py + q(1-y) \in L \cap P^c$ whenever $y \in [a, b]$. Arbitrarily choose d such that $a \leq d \leq b$ and consider $U(d)$. The existentially quantified vector $z = (z_1, \ldots, z_k)$ is forced to equal $pd + q(1-d)$ and this point is in $L \cap P^c$. Now, $z \in L \cap P^c$ implies that $z \notin P$, so z does not satisfy the formula $\bigwedge_{l \in \{N_1, \ldots, N_p\}} \neg l \quad \wedge \quad R(z_1, \ldots, z_k)$ (recall that P corresponds to the points satisfying $(P_1 \vee \ldots \vee P_m)$). Thus, $d \notin U$ and U excludes an interval since d can be arbitrarily chosen in the interval $[a, b]$.
3. $S \neq \emptyset$ and for every pair of points $p, q \in S$, the line L from p to q satisfies $L \cap P^c = \emptyset$. Define $T = \{sy + s'(1-y) \mid s, s' \in S \text{ and } 0 \leq y \leq 1\}$. This set is convex, $S \subseteq T$, and $T \cap P^c = \emptyset$. Apply Corollary 1 on the convex sets P^c and T: there exists a separating hyperplane $H = \{x \in R^k \mid e^T x = f\}$

such that $P^c \cap H = \emptyset$. Consequently, $T = T \cap \{x \in R^k \mid e^T x \leq f\}$ or $T = T \cap \{x \in R^k \mid e^T x \geq f\}$; let $\widehat{H}$ denote the closed halfspace that gives equality.

We claim that $S' \cap P = S' \cap \widehat{H}$; this implies that the positive literals $P_1, \ldots, P_m$ in C can be replaced by a single literal l that equals either $e^T x \leq f$ or $e^T x \geq f$. This is a contradiction since the number of non-Horn clauses decreases.

$\underline{S' \cap P \subseteq S' \cap \widehat{H}}$: $T = T \cap \widehat{H}$ and $S \subseteq T$ so $S = S \cap \widehat{H}$, too. Since $S \subseteq S'$, it follows that $S' \cap P = S = S \cap \widehat{H} \subseteq S' \cap \widehat{H}$.

$\underline{S' \cap \widehat{H} \subseteq S' \cap P}$: Suppose that $\widehat{H} \not\subseteq P$. Then, there exists an $x \in P^c$ such that $x \in \widehat{H}$. We know that P^c and T are separated by H and that $H \cap P^c = \emptyset$. Thus, no point in P^c is in $\widehat{H}$ by the very choice of $\widehat{H}$. Hence, $\widehat{H} \subseteq P$ and $S' \cap \widehat{H} \subseteq S' \cap P$.

These three cases complete the proof. □

The proof of Theorem 1 leads to equivalent characterizations of semilinear relations that are quantifier-free Horn definable.

Definition 1. *We say that $R \subseteq \mathbb{R}^n$ is* essentially convex *if for all $p, q \in R$ there are only finitely many points on the line between p and q that are not in R.*

Corollary 2. *Let R be a semilinear relation. Then the following are equivalent.*

1. *R is Horn-definable;*
2. *R is essentially convex;*
3. *R does not exclude an interval.*

Proof. For the implication from 1 to 2, assume that R is Horn-definable, and let $p, q \in R$. For each clause C from a Horn definition of R with k disequality literals, at most k points on the line between p and q do not satisfy C (since if some point does not satisfy an inequality from C, then also p or q do not satisfy the inequality). Hence, the number of points on the line between p and q that are not contained in R is bounded by the total number of disequality literals in a Horn definition of R, and finite.

The implication from 2 to 3 is immediate. In the previous Theorem we have shown that 3 implies 1, which proves the statement. □

Proof (of Theorem 1). If all relations of Γ are Horn-definable, then CSP(Γ) can be solved in polynomial time (Theorem 2). Otherwise, if there is a relation R from Γ that is not Horn definable, then Lemma 4 shows that R excludes an interval, and NP-hardness of CSP(Γ) follows by Lemma 3.

So we only have to show that CSP(Γ) is in NP. Let Φ be an arbitrary instance of CSP(Γ). Recall that we represent semilinear relations in the input by quantifier-free conjunctive normal form formulas over LI. One can now nondeterministically guess one literal from each clause and verify – in polynomial-time by Theorem 2 – that all the selected literals are simultaneously satisfiable. □

4 Applications

4.1 Generalised Linear Programming

The optimisation problem *linear programming* (LP) is defined in the following way: $\max\{c^T x \mid Ax \leq b\}$ where x is a variable vector taking values from $\mathbb{R}^k$ and A, b, c are matrices and vectors (of suitable dimensions) with entries in $\mathbb{Q}$.

We generalise LP as follows: let Γ be a set of semilinear relations and consider the problem $\max\{c^T x \mid \Phi(x)\}$ where Φ is a quantifier-free pp-formula over Γ. Denote this problem GLP(Γ). It is easy to see that LP is exactly the problem GLP$(\mathbb{R}; LI)$. These optimisation problems may have unbounded solutions and we let the objective value be ∞ if this is the case. If a given optimisation instance has no solution, i.e., if $\{c^T x \mid Ax \leq b\} = \emptyset$ or $\{c^T x \mid \Phi(x)\} = \emptyset$, then the objective value is defined to be $-\infty$.

Theorem 4. *Let $\Gamma = (\mathbb{R}; LI, R_1, R_2, \ldots\}$ be a constraint language such that $R_1, R_2, \ldots$ are semilinear. Then, either each R_i has a quantifier-free Horn definition in $(\mathbb{R}; LI)$ and GLP(Γ) is in P, or GLP(Γ) is NP-hard.*

Proof. If there is an R_i that does not have a quantifier-free Horn definition in $(\mathbb{R}; LI)$, then CSP$(\mathbb{R}; LI, R_i)$ is NP-hard by Theorem 1 and GLP(Γ) is NP-hard, too.

Assume instead that each R_i is quantifier-free Horn definable in $(\mathbb{R}; LI)$. Let (c, Φ) denote an arbitrary instance of GLP(Γ) over variables $x_1, \ldots, x_k$. First check whether the instance $\exists \overline{x}.\Phi(\overline{x})$ of CSP(Γ) is true or not; if not, then return $-\infty$ and stop. Otherwise, we know from Theorem 2 that the solution set S to Φ satisfies

$$S_* = \{x \in \mathbb{R}^k \mid Ax \leq \alpha, Bx \neq \beta\} \subseteq S \subseteq \{x \in \mathbb{R}^k \mid Ax \leq \alpha\} = S^*$$

where S_* and S^* have the same dimension in $\mathbb{R}^k$ and A, α can be computed in polynomial time. Now maximise $c^T x$ over $Ax \leq \alpha$ and let M denote the optimal value; this is a linear program so it can be solved in polynomial time. If $M = \infty$, we return ∞. Otherwise, it is known that the optimum is a rational number. Check whether the instance $\Phi \wedge c^T x = M$ has a solution. If it has, then the optimal value of (c, Φ) is M. Otherwise, S equals $\{x \in \mathbb{R}^k \mid Ax \leq \alpha\}$ with a finite number of subsets of hyperplanes being removed, and the removed points include the points where $c^T x$ attains its maximum. In this case, there exists a $\delta > 0$ such that for every $0 < \epsilon < \delta$, (c, Φ) has a solution x such that $c^T x = M - \epsilon$. □

4.2 Temporal Constraint Satisfaction

A *temporal constraint language* Γ is a structure $(\mathbb{R}; R_1, R_2, \ldots)$ with a first-order definition in $(\mathbb{R}; <)$. The complexity of every temporal constraint language has been determined by Bodirsky & Kara [3]. Fundamental and well-known temporal constraint lanugages are $PA = \{\leq, <, \neq, =\}$ (which is often called the *point algebra* in the literature [6]) and Ord-Horn [16] (which is a subset of the quantifier-free Horn definable relations over $(\mathbb{R};\leq)$).

Temporal constraint languages are often extended with some mechanism for expressing *metric* time, i.e. the ability to assign numerical values to variables and performing restricted arithmetic calculations. It is fair to say that the main bulk of languages for metric temporal reasoning are semilinear, cf. [6,13]. Inspired by this fact, we will completely classify the complexity of metric temporal constraint languages Γ that

1. are semilinear,
2. contain the temporal relation $\leq$,
3. allow basic addition $x+y=z$, and
4. contain the singleton relation $\{1\}$.

We first note that the equality relations in LI are pp-definable in $(\mathbb{R};+,1)$ with small pp-formulas; the proof is straightforward but tedious, using the fact that by iterated doubling we can produce a primitive positive definition of a number n that has logarithmic size in n.

Lemma 5. *The relation $\{(x_1,\ldots,x_l) \mid n_1x_1+\ldots+n_lx_l=n_0\}$ is pp-definable with a formula of length polynomial in the representation size of $n_0,n_1,\ldots,n_l$ in $(\mathbb{R};\{(x,y,z) \mid x+y=z\},1)$ for arbitrary $n_0,\ldots,n_l\in\mathbb{Q}$.*

Proof. Observe that we can assume that $n_0,\ldots,n_l$ are integers, because we can multiply the equality $n_1x_1+\cdots+n_lx_l=n_0$ by the least common multiple of the denominators of $n_0,n_1,\ldots,n_l$ and obtain an equivalent equation. Also note that $x=0$ is pp-definable by $x+x=x$, and we therefore freely use the terms 0 and 1 in pp-definitions. Similarly, $x=-1$ is pp-definable by $x+1=0$.

The proof is by induction on l. We first show how to express equations of the form $n_1x_1+n_2x_2=x_3$. By setting x_2 to -1 and x_3 to 0, this will solve the case $l=1$. We later complete the induction. For positive n_1,n_2, this formula is equivalent to

$$\begin{aligned}\exists u_1,\ldots,u_{n_1},v_1,\ldots,v_{n_2}.\quad & u_1=x_1\wedge\bigwedge_{i=1}^{n_1-1}x_1+u_i=u_{i+1}\\ &\wedge v_1=x_2\wedge\bigwedge_{i=1}^{n_2-1}x_2+v_i=v_{i+1}\\ &\wedge u_{n_1}+v_{n_2}\leq x_3\ .\end{aligned}$$

However, this formula is exponential in the representation size of n_1 and n_2, and hence cannot be used in polynomial-time reductions.

Let $bit(n,i)$ denote the i-th lowest bit in the binary representation of an integer n and $1\leq i\leq\lfloor\log n\rfloor+1$. Since $x=0$ can be pp-defined by $x+x=x$, the formula $x=bit(n,i)$ is (for fixed n,i) pp-definable as well, and we will use the term 0 and $bit(n,i)$ freely in other pp-definitions.

The following formula is equivalent to the previous one

$$\begin{aligned}\exists \bar{a}, \bar{b}, \bar{c}, \bar{d}. \quad & a_1 = x_1 \wedge \bigwedge_{i=1}^{\lfloor \log n_1 \rfloor} a_i + a_i = a_{i+1} \ \wedge \\ & b_1 = x_2 \wedge \bigwedge_{i=1}^{\lfloor \log n_2 \rfloor} b_i + b_i = b_{i+1} \ \wedge \\ & c_1 = bit(n_1, 1) \wedge \bigwedge_{i=1}^{\lfloor \log n_1 \rfloor} bit(n_1, i+1) a_i + c_i = c_{i+1} \ \wedge \\ & d_1 = bit(n_2, 1) \wedge \bigwedge_{i=1}^{\lfloor \log n_2 \rfloor} bit(n_2, i+1) b_i + d_i = d_{i+1} \ \wedge \\ & c_{\lfloor \log n_1 \rfloor + 1} + d_{\lfloor \log n_2 \rfloor + 1} \leq x_3 \ .\end{aligned}$$

and has polynomial length in the representation size of n_1 and n_2. If $l = 2$, and $n_1 = 0$ or $n_2 = 0$, then the proof is similar. If n_1 and n_2 have different signs, we replace the conjunct $c_{\lfloor \log n_1 \rfloor + 1} + d_{\lfloor \log n_0 \rfloor + 1} = e$ in the formula above appropriately by $c_{\lfloor \log n_1 \rfloor + 1} + e = d_{\lfloor \log n_0 \rfloor + 1}$ or $d_{\lfloor \log n_0 \rfloor + 1} + x_3 = c_{\lfloor \log n_1 \rfloor + 1}$. If both n_1 and n_2 are negative, then we use the pp-definition $\exists x_3'. -n_1 x_1 - n_2 x_2 = x_3' \wedge x_3' + x_3 = 0$.

Equalities of the form $n_1 x_1 + n_2 x_2 = n_0$ can now be defined by $\exists x_3. n_1 x_1 + n_2 x_2 = x_3 \wedge x_3 = n_0$. Now suppose that $l > 2$. By the inductive assumption, there is a pp-definition ϕ_1 for $n_1 x_1 + u = n_0$ and a pp-definition ϕ_2 for $n_2 x_2 + \ldots n_l x_l = u$. Then $\exists u. \phi_1 \wedge \phi_2$ is a pp-definition for $n_1 x_1 + \cdots + n_l x_l = n_0$. It is straightforward to verify that the pp-definition has polynomial size in the representation size of this equality. □

Given a linear equality $L(x_1, \ldots, x_k) \equiv c_1 x_1 + \ldots + c_l x_l = c_0$, let $\phi_{L(x_1, \ldots, x_k)}$ denote the pp-definition of $L(x_1, \ldots, x_k)$ in $(\mathbb{R}; \{(x,y,z) \mid x+y=z\}, 1)$ obtained in Lemma 5. The complete classification is now a consequence of Theorem 1.

Theorem 5. *Let Γ be a constraint language whose relations are semilinear and that contains the relations $\{(x,y,z) \mid x+y=z\}$, $\{1\}$, and $\leq$. Then, either each relation R from Γ has a quantifier-free Horn definition in $(\mathbb{R}; LI)$ and CSP(Γ) is in P, or CSP(Γ) is NP-hard.*

Proof. Assume that a relation R from Γ is not quantifier-free Horn definable in $(\mathbb{R}; LI)$; this implies that CSP$(\mathbb{R}; LI, R)$ is NP-complete by Theorem 1. We show that CSP(Γ) is NP-complete, too. Let Φ be an arbitrary instance of CSP$(\mathbb{R}; LI, R)$. Construct an instance Ψ of CSP(Γ) by replacing each occurrence of a linear equality constraint L by ϕ_L, and each occurrence of a linear inequality constraint $L = (c_1 x_1 + \ldots c_l x_l \leq c_0)$ in ϕ by a $\phi_{c_1 x_1 + \cdots + c_l x_l - y = 0} \wedge y \leq c$; use fresh variables for y and all existentially quantified variables introduced by ϕ_L. The resulting formula Ψ can be re-written as a primitive positive sentence over Γ without increasing its length and, by Lemma 5, the length of Ψ is polynomial in the length of Φ. Since Φ is satisfiable if and only if Ψ is satisfiable, CSP(Γ) is NP-hard. Clearly, the problem is in NP. If every $R \in \Gamma$ is quantifier-free Horn definable in $(\mathbb{R}; LI)$, then CSP(Γ) is in P by Theorem 2. □

We have seen that $(\mathbb{R}; LI)$ is primitive positive definable in $(\mathbb{R}; +, 1, \leq)$. This shows that linear program feasibility can, up to primitive positive definability, be cast as a CSP with a finite constraint language; this is interesting because for such constraint languages, the computational complexity of the CSP does not depend on the representation of the relations in the input.

5 Conclusions

We have presented complexity classifications for certain constraint satisfaction problems, and the results are to a large extent based on logical methods. We feel that the results and ideas presented in this paper can be extended in many different directions. Hence, it seems worthwhile to provide some concrete suggestions for future work. An interesting example is the class of structures that are definable in *Presburger arithmetics* [17], i.e., structures that are first-order definable over the integers with addition $(\mathbb{Z}; +)$. Presburger arithmetics is important for many different reasons: one is, of course, that $(\mathbb{Z}; +)$ is a very natural and fundamental class with a long history and many links to other branches of mathematics. Another reason is that the the problem to decide whether a given first-order sentence is true in $(\mathbb{Z}; +)$ is superexponential [8] (and thus provably worse than NP-complete) which makes the search for computationally easy fragments worthwhile. One may object that Presburger arithmetics do not admit quantifier elimination and this might make the application of our methods more difficult. However, slightly expanded structures do have quantifier elimination: one example is $(\mathbb{Z}; +, \{P_i \mid i > 1\}, 0, \pm 1, \pm 2, ...)$ where $P_n = \{x \in \mathbb{Z} \mid x \text{ is divisible by } n\}$ [15].

Adapting the universal-algebraic approach to cover the results of the paper would be another interesting line of research. We have already mentioned that the structures studied in this paper are not ω-categorical so this adaptation is nontrivial. Such an algebraic approach may be used for studying other structures that are not ω-categorical: two families of structures that come to mind are torsion-free divisible abelian groups (such as $(\mathbb{R}; +)$) and algebraically closed fields (such as $(\mathbb{C}; +, *)$). They share the property of being categorical in all uncountable cardinals (cf. [15] and [18]) which possibly simplifies the study of them.

Acknowledgements

Peter Jonsson is partially supported by the *Center for Industrial Information Technology* (CENIIT) under grant 04.01 and by the *Swedish Research Council* (VR) under grant 2006-4532.

References

1. Banach, S.: Sur les fonctionelles linéaires. Studia Math. 1, 211–216, 223–229 (1929)
2. Bodirsky, M.: Constraint satisfaction problems with infinite templates. In: Creignou, N., Kolaitis, P., Vollmer, H. (eds.) Complexity of Constraints. LNCS, vol. 5250. Springer, Heidelberg (2008)
3. Bodirsky, M., Kára, J.: The complexity of temporal constraint satisfaction problems. In: Proceedings of the 40th ACM Symposium on Theory of Computing (STOC-2008), pp. 29–38 (2008)
4. Bulatov, A., Jeavons, P., Krokhin, A.: Classifying the computational complexity of constraints using finte algebras. SIAM J. Comput. 34(3), 720–742 (2005)
5. Cohen, D., Jeavons, P., Jonsson, P., Koubarakis, M.: Building tractable disjunctive constraints. J. ACM 47(5), 826–853 (2001)
6. Drakengren, T., Jonsson, P.: Computational complexity of temporal constraint problems. In: Fisher, M., Gabbay, D., Vila, L. (eds.) Handbook of Temporal Reasoning in Artificial Intelligence, pp. 197–218. Elsevier, Amsterdam (2005)
7. Ferrante, J., Rackoff, C.: A decision procedure for the first order theory of real addition with order. SIAM J. Comput. 4(1), 69–76 (1975)
8. Fischer, M., Rabin, M.: Super-exponential complexity of Presburger arithmetic. In: Karp, R.M. (ed.) Complexity of Computation, pp. 56–64. SIAM, Philadelphia (1974); Proceedings of the SIAM-AMS symposium in applied mathematics
9. Garey, M., Johnson, D.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman, New York (1979)
10. Hahn, H.: Über linearer Gleichungssysteme in linearer Räumen. J. Reine Angew. Math. 157, 214–229 (1927)
11. Hodges, W.: A Shorter Model Theory. Cambridge University Press, New York (1997)
12. Jeavons, P., Cohen, D., Gyssens, M.: Closure properties of constraints. J. ACM 44(4), 527–548 (1997)
13. Jonsson, P., Bäckström, C.: A unifying approach to temporal constraint reasoning. Artif. Intell. 102(1), 143–155 (1998)
14. Koubarakis, M.: Tractable disjunctions of linear constraints: Basic results and applications to temporal reasoning. Theor. Comput. Sci. 266(1–2), 311–339 (2001)
15. Marker, D.: Model Theory: An Introduction. Springer, Heidelberg (2002)
16. Nebel, B., Bürckert, H.-J.: Reasoning about temporal relations: a maximal tractable subclass of Allen's interval algebra. J. ACM 42(1), 43–66 (1995)
17. Presburger, M.: Über die Vollständigkeit eines gewissen Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In: Comptes Rendus du I congrès de Mathématiciens des Pays Slaves, pp. 92–101 (1929)
18. Steinitz, E.: Algebraischen Theorie der Körper. J. Reine. Angew. Math. 137, 167–309 (1910)

Floats and Ropes: A Case Study for Formal Numerical Program Verification[*]

Sylvie Boldo

INRIA Saclay - Île-de-France, ProVal, Orsay, F-91893
LRI, Univ Paris-Sud, CNRS, Orsay, F-91405

Abstract. We present a case study of a formal verification of a numerical program that computes the discretization of a simple partial differential equation. Bounding the rounding error was tricky as the usual idea, that is to bound the absolute value of the error at each step, fails. Our idea is to find out a precise analytical expression that cancels with itself at the next step, and to formally prove the correctness of this approach.

1 Introduction

Given a program using floating-point arithmetic, it is pretty hard to know the final rounding error of the result. We are interested in proving numerical analysis programs with a very high level of guarantee. We present here a simple example of scientific computation. The basic property is that each floating-point result is a correct rounding of the exact real value: using the default rounding mode, the result is the floating-point number that is closest to the real value. This property is defined in the IEEE-754 standard [1] and all modern processors comply with it.

Nevertheless, even if each computation is correct, *i.e.* the best possible, there is no guarantee that the final result after many such computations is still accurate. There exist several methods for bounding the final error of a program, including interval arithmetic, forward and backward analysis [2,3]. These well-known methods may or may not give useful results. However, when they state a bad rounding error, it does not always imply the error is huge. It is a known fact that floating-point errors may cancel [3] but it is very difficult to handle. We use here a method that displays these cancellations and takes advantage of them. This idea of exhibiting floating-point errors cancellation has been used by Even, Seidel and Ferguson in [4]. This article's technique is also linked to static analysis [5], and provides more precision and more readability at the cost of less genericity. This technique is also linked to expansions [6] as the error is somewhat "computed" and used.

To increase the trust in our results, we use deductive formal methods: we machine-check all proofs using the Coq proof checker [7]. We use a high-level formalization of floating-point numbers [8,9]. We use the Why platform for verification of C programs, that includes the Caduceus tool [10,11]. The Caduceus

[*] This work was funded by the French national research organization (ANR), by the CerPAN (ANR-05-BLAN-0281-04) and F∮st (ANR-08-BLAN-0246-01) projects.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 91–102, 2009.

tool allows the user to precisely specify a C program. Each function is annotated with pre-conditions (what the function `requires` from the inputs) and post-conditions (what the function `ensures` at its end). The annotations and requirements (pointer dereferencing for example) are then transformed into proof obligations that have to be solved by proof assistants or decision procedures.

The Caduceus tool has floating-point annotations [12] that allow to specify numerical programs. More precisely, each floating-point number has a ghost value called exact which does not suffer from rounding. This real value is then computed with the same operations as the float value except that the ghost operation is exact. The macro `round_error`(f) is then used for $|f - \text{exact}(f)|$. More, each floating-point number has another ghost value called model that the user may set and which does not suffer from rounding. It is used to represent the ideal result of the function (computed with infinite sums, no discretization...). For example, to compute a naive exponential by the polynomial evaluation of `1+x+x*x/2`, the corresponding exact value is $1+x+\frac{x^2}{2}$ (with mathematical exact operations) while the model value is $\exp(x)$.

Inside the annotations, all computations are exact. For example, this program takes x as input and multiplies it by 2.

```
/*@ requires |x| < 2^^(1022)
  @ ensures  \result=2*x
  @     && \round_error(\result)=2*\round_error(x) */

double multiply2(double x) { return 2*x; }
```

This function requires x to be small enough so that the multiplication does not overflow. It ensures that the result, denoted by the macro `\result` is equal to the *mathematical* multiplication of x by 2. This is correct as the radix is 2. More, the rounding error of the result is twice the rounding error of the input.

Section 2 describes a first simplified example. Section 3 tackles the bounding of the rounding error of the discretization of the spread of acoustic waves on a rope. Section 4 gives conclusive remarks and perspectives.

2 First Example: Linear Recurrence of Order 2

2.1 The Problem

We first present a linear recurrence of order 2. For some initial values u_0 and u_1, we compute the following sequence:

$$u_{n+1} = 2 \times u_n - u_{n-1}.$$

This may seem a silly idea as this sequence can be solved. Indeed, we know that, mathematically, $u_n = u_0 + (u_1 - u_0) \times n$. Nevertheless, this example is representative of the analytical error idea and corresponds rather nicely to our real problem (Section 3 with $\mathtt{a} = 0$).

To compute u_N, we assume that (u_i) is bounded by 1: for all $i \leq N$, $|u_i| \leq 1$. This requires u_0 and u_1 to be small and close enough one to another. This noteworthy limitation still corresponds to similar properties in our real program. Without this property, the error bound would be multiplied by $\max_{0 \leq i \leq N} |u_i|$.

We use this C program that we will later annotate:

```
double comput_seq(double u0, double u1, int N) {
  int i;
  double uprev, ucur, tmp;
  uprev=u0; ucur =u1;

  for (i=2; i<=N; i++) {
    tmp   = 2*ucur-uprev;
    uprev = ucur;
    ucur  = tmp;
  }
  return ucur;
}
```

We use IEEE double precision floating-point numbers [1] with a 53-bits long mantissa. So, if f is the rounding in `double` of a real x, then $|f - x| \leq 2^{-53}|f|$.

As the exact value of $|u_n|$ is bounded by 1, if the errors of u_{n-1} and u_{n-2} are not too big, then the error in the subtraction is less than 2^{-53}. A natural error analysis gives: let $E_i = u_i - \text{exact}(u_i)$, then $|E_{i+1}| \leq 2 \times |E_i| + |E_{i-1}| + 2^{-53}$. Assuming u_0 and u_1 are error-free, this gives us that $|E_N|$ is roughly equal to $2^N \times 2^{-53}$. This error is very pessimistic and should be improved upon.

2.2 The Analytical Error

The idea is that the error should be signed. Taking its absolute value can only lead to an exponential error. By keeping its sign, we have that:

$$E_{i+1} = 2 \times E_i - E_{i-1} + \varepsilon_{i+1} \quad \text{with} \quad |\varepsilon_{i+1}| \leq 2^{-53}.$$

The key point is that we have now a subtraction between $2 \times E_i$ and E_{i-1}. At each step, the error will only be added a small value, therefore E_{i-1} is close to E_i, so that $2 \times E_i - E_{i-1} \approx E_i$. This allows us to get rid of the exponential in the error bound. We now assume that u_0 and u_1 are not exact anymore. Of course, the initial errors must not be too big: we assume the computed u_i do not exceed 2. More precisely, the annotations of the C function are given in Figure 1.

Theorem 1. *If the pre-conditions of Figure 1 are satisfied, then the post-conditions of Figure 1 hold.*

Proof. This proof deeply relies on the definition of the predicate *mkp* which is a loop invariant inductively proved correct at each iteration update. The idea is to keep track of both the exact value and the exact floating-point error of u_p and u_c in order to bound the final error.

```
/*@ requires 2 <= N <= 2^^25-1 &&
  @      \round_error(u0) + \round_error(u1) <= 1./(6*N) &&
  @      \forall int k; 0 <= k <= N =>
  @             |\exact(u0)+k*(\exact(u1)-\exact(u0))| <= 1
  @ ensures
  @   \exact(\result)==\exact(u0)+N*(\exact(u1)-\exact(u0))
  @    && \round_error(\result) <= N*(N+1)/2.*2^^(-53)
  @                + N*(\round_error(u0)+\round_error(u1))
  @*/

double comput_seq(double u0, double u1, int N) { ...
```

Fig. 1. Annotated C program for computing the linear recurrence of order 2

The predicate *mkp* is a property linking the inputs u_0 and u_1 and the state of the program: the number of iterations n and the current $u_n = \texttt{ucur} = u_c$ and $u_{n-1} = \texttt{uprev} = u_p$. For a given float f, let us denote by $\delta(f) = f - \mathrm{exact}(f)$. We have $\texttt{round\_error}(f) = |\delta(f)|$. We define the predicate *mkp* by:

$$\begin{aligned} mkp(u_0, u_1, u_c, u_p, n) = {} & \exists \varepsilon : \mathbb{N} \to \mathbb{R}, \\ & \forall i \in \mathbb{N}, i \le n \Rightarrow |\varepsilon_i| \le 2^{-53} \\ & \wedge\ \delta(u_p) = \textstyle\sum_{j=0}^{n-1} (n-j)\ \varepsilon_j + (1-n)\ \delta(u_0) + n\ \delta(u_1) \\ & \wedge\ \delta(u_c) = \textstyle\sum_{j=0}^{n} (n+1-j)\ \varepsilon_j + (-n)\ \delta(u_0) + (n+1)\ \delta(u_1) \end{aligned} .$$

As soon as *mkp* is defined, the proof is rather easy. The exact value of u_i is computed by recurrence. The *mkp* property is proved the same way. At stage i of the iteration, we define a new ε_i which is the signed rounding error committed during this iteration. As the multiplication is exact, $\varepsilon_i = u_i - (2 \times u_{i-1} - u_{i-2})$. This value can easily be bounded as this is the error of one single subtraction such that the result is smaller than 2. Therefore $|\varepsilon_i| \le 2^{-53}$ and

$$|E_N| \le \frac{N(N+1)}{2} 2^{-53} + N \times (|\delta(u_0)| + |\delta(u_1)|).$$

There is left to guarantee that $|u_i| \le 2$, we first know, according to our assumptions, that $|\mathrm{exact}(u_i)| \le 1$. So there is left to prove that $|E_i| \le 1$. And the analytical expression of the error shows that the floating-point error is smaller than $i(i+1) \times 2^{-54} + i\,(|\delta(u_0)| + |\delta(u_1)|)$. We therefore need to bound N by about 2^{25} so that $i(i+1)2^{-54}$ is bounded enough. Moreover, we have to bound $|\delta(u_0)| + |\delta(u_1)| = \mathrm{round\_error}(u_0) + \mathrm{round\_error}(u_1)$ by $1/(6N)$. These values are sufficient to guarantee that the error is bounded: $|E_i| \le 1$. □

3 Second Example: Rope

3.1 The Problem

The piece of code that is studied here is extracted from a numerical code by F. Clément about acoustic waves [13]: given a rope attached at its two ends, we create a wave by applying a force (initializations). The rope then undulates, depending on some mathematical equations that can be discretized and computed.

The mathematical point of view is that look for u from $\mathbb{R}^2$ to $\mathbb{R}$, solution of the differential equation, knowing initial values of u and its derivative for $t = 0$:

$$\frac{\partial^2 u(x,t)}{\partial t^2} - c^2 \frac{\partial^2 u(x,t)}{\partial x^2} = 0.$$

The value $u(x, t)$ gives the position of the rope at the abscissa x and the time t. It is discretized both in space and time with steps $(\Delta x, \Delta t)$. The result is a matrix p where $p[i][k] = p_i^k$ is the position of the rope at the abscissa $i \times \Delta x$ and the time $k \times \Delta t$. The matrix p is computed by the following piece of code:

```
for (k=1; k<nk; k++) {
  p[0][k+1] = 0.;
  for (i=1; i<ni; i++) {
    dp = p[i+1][k] - 2.*p[i][k] + p[i-1][k];
    p[i][k+1] = 2.*p[i][k] - p[i][k-1] + a*dp;
   }
  p[ni][k+1] = 0.;
}
```

This is the main iteration of the program. Before that, $p[\ldots][0]$ and $p[\ldots][1]$ are set. The value $\mathtt{a}$ is a parameter computed previously. It is assumed that $0 < \mathtt{a} \lessapprox 1$. Typically, $\mathtt{a}$ can be the rounding of 0.9 or 0.99. The value i is bounded by the ends of the rope 0 and $\mathtt{ni}$. We compute the position of the rope between the initial time 0 and a maximum time $\mathtt{nk}$.

We assume that (p_i^k) is bounded. The reason is that (p_i^k) represents values that are supposed to be smaller than 1 (as the rope cannot fly away). These model p_i^k cannot be computed as they would need infinite sums or absence of discretizations. Nevertheless, the $\mathrm{exact}(p_i^k)$ are near these model values so we may assume they are smaller that 1.5. We assume $\mathtt{nk}$ is small enough to guarantee that the floating-point values $|p_i^k|$ are smaller than 2. As the error will be proved proportional to k^2, this roughly corresponds to $\mathtt{nk} \le 2^{22}$.

3.2 The Pyramids

Let ε_i^k be the (signed) floating-point error made in the two lines computing p_i^k. From the preceding program, we set

$$\varepsilon_i^{k+1} = p_i^{k+1} - (2p_i^k - p_i^{k-1} + \mathrm{exact}(\mathtt{a}) \times (p_{i+1}^k - 2p_i^k + p_{i-1}^k)).$$

As the $|p_i^k|$ are assumed to be smaller than 2, this value can be bounded. To prove that, we use simple interval arithmetic: the idea is to bound each step of the proof. We formally prove that $|\varepsilon_i^{k+1}| \le 85 \times 2^{-52}$ for a reasonable error bound for $\mathtt{a}$, that is to say $|\mathtt{a} - \mathrm{exact}(\mathtt{a})| \le 2^{-49}$.

Note that the floating-point error $E_i^k = p_i^k - \mathrm{exact}(p_i^k)$ is much bigger than ε_i^k as it contains all the preceding rounding errors. More precisely, given the definition of p_i^k and the preceding discussion on the dependencies, E_i^k depends and only depends on the following ε_j^l:

$$\begin{array}{ccccccccc}
 & & & & \varepsilon_i^k & & & & \\
 & & & \varepsilon_{i-1}^{k-1} & \varepsilon_i^{k-1} & \varepsilon_{i+1}^{k-1} & & & \\
 & & \varepsilon_{i-2}^{k-2} & \varepsilon_{i-1}^{k-2} & \varepsilon_i^{k-2} & \varepsilon_{i+1}^{k-2} & \varepsilon_{i+2}^{k-2} & & \\
 & \iddots & & & \vdots & & & \ddots & \\
\varepsilon_{i-k}^0 & & \cdots & & \varepsilon_i^0 & & \cdots & & \varepsilon_{i+k}^0
\end{array}$$

This unfortunately means both that the error is at least proportional to k^2, and that the analytical error is a pyramidal double summation.

This is quite harder than the previous expression, as a double summation is more difficult to handle, both on the paper and using a proof assistant. Moreover, the error is not the sum of all the ε_j^l of the pyramid. They have to be multiplied by a well-chosen constant depending on their place in the pyramid. This constant is far from trivial and is mathematically defined in the next subsection.

3.3 The α Sequence

Each ε_i^k is to be multiplied by its own constant that gives the significance of each rounding error in the final error. Therefore, let us introduce $\alpha : (\mathbb{Z} \times \mathbb{Z}) \to \mathbb{R}$ defined by

$$\alpha_0^0 = 1 \qquad \forall i \neq 0,\ \alpha_i^0 = 0$$

$$\alpha_{-1}^1 = \alpha_1^1 = \breve{a} \qquad \alpha_0^1 = 2(1-\breve{a}) \qquad \forall i \notin \{-1,0,1\},\ \alpha_i^1 = 0$$

$$\alpha_i^k = \breve{a} \times (\alpha_{i-1}^{k-1} + \alpha_{i+1}^{k-1}) + 2(1-\breve{a}) \times \alpha_i^{k-1} - \alpha_i^{k-2}$$

where $\breve{a}$ is the exact value of the floating-point value a of the preceding subsection (so $\breve{a}$ is typically 0.9 or 0.99). The non-zero terms fit in a pyramid looking like this for $\breve{a} = 0.9$, where lines are indexed by k and columns by i:

$$\begin{array}{ccccccccc}
 & & & & 1 & & & & \\
 & & & 0.9 & 0.2 & 0.9 & & & \\
 & & 0.81 & 0.36 & 0.66 & 0.36 & 0.81 & & \\
 & 0.729 & 0.486 & 0.495 & 0.58 & 0.495 & 0.486 & 0.729 & \\
 & \iddots & & & & & & \ddots & \\
0.9^k & & & & \cdots & & & & 0.9^k
\end{array}$$

It is easy to prove that $\alpha_i^k = \alpha_{-i}^k$, that $\alpha_k^k = \breve{a}^k$ and that if $i < -k$ or $i > k$, then $\alpha_i^k = 0$. More interesting, the sum "by line" has a surprisingly simple expression:

Lemma 1.

$$\sum_{i=-\infty}^{+\infty} \alpha_i^k = k + 1.$$

Proof. We have:

$$\sum_{i=-\infty}^{+\infty} \alpha_i^{k+1} = 2\breve{a} \sum_{i=-\infty}^{+\infty} \alpha_i^k + 2(1-\breve{a}) \sum_{i=-\infty}^{+\infty} \alpha_i^k - \sum_{i=-\infty}^{+\infty} \alpha_i^{k-1} = 2 \sum_{i=-\infty}^{+\infty} \alpha_i^k - \sum_{i=-\infty}^{+\infty} \alpha_i^{k-1}.$$

The sum by line verifies the linear recurrence of Section 2. As $\sum_{i=-\infty}^{+\infty} \alpha_i^0 = 1$ and $\sum_{i=-\infty}^{+\infty} \alpha_i^1 = 2$, we have $\sum_{i=-\infty}^{+\infty} \alpha_i^k = k + 1$. □

Lemma 2. $\alpha_i^k \geq 0$

Proof. The demonstration was found out by M. Kauers and V. Pillwein.

If we denote by P the Jacobi polynomial, we have

$$\alpha_n^j = \sum_{k=j}^{n} \binom{2k}{j+k} \binom{n+k+1}{2k+1} (-1)^{j+k} a^k = a^j \sum_{k=0}^{n-j} P_k^{(2j,0)}(1-2a)$$

Now the conjecture follows directly from the inequality of Askey and Gasper[14], which asserts that $\sum_{k=0}^{n} P_k^{(r,0)}(x) > 0$ for $r > -1$ and $-1 < x \leq 1$ (see Theorem 7.4.2 in The Red Book [15]). □

This assertion is not formally proved as it involves both many complex computations and very high level mathematics. Moreover, this lemma can be ignored at the price of a less tight bound (see Section 3.7).

3.4 The Analytical Error

Now, we claim that we can express the exact floating-point error of p_i^k in an analytical way:

Theorem 2.

$$E_i^k = p_i^k - \mathit{exact}(p_i^k) = \sum_{l=0}^{k} \sum_{j=-l}^{l} \alpha_j^l \; \varepsilon_{i+j}^{k-l}$$

Proof. The analytical expression exactly fits the computation of p_i^{k+1} and the sequence α_i^k is defined so that they cancel at the right time.

We prove the expression of E_i^k by induction on k. We assume the initializations fulfill this requirement by choosing wisely ε_i^0 and ε_i^1 so that this expression is correct for $k = 0$ and $k = 1$. We now assume the expression is correct for $k - 1$ and k and we prove it for $k + 1$:

$$\begin{aligned} E_i^{k+1} &= p_i^{k+1} - (2p_i^k - p_i^{k-1} + \breve{a} \times (p_{i+1}^k - 2p_i^k + p_{i-1}^k)) \\ &\quad +2(1-\breve{a})(p_i^k - \mathrm{exact}(p_i^k)) \\ &\quad +\breve{a}(p_{i+1}^k - \mathrm{exact}(p_{i+1}^k) + p_{i-1}^k - \mathrm{exact}(p_{i-1}^k)) \\ &\quad -(p_i^{k-1} - \mathrm{exact}(p_i^{k-1})) \\ &= \varepsilon_i^{k+1} + 2(1-\breve{a})E_i^k + \breve{a}(E_{i+1}^k + E_{i-1}^k) - E_i^{k-1} \end{aligned}$$

After a lot of tiring but stupid computation (mostly summation games), we have the equality:

$$\begin{aligned} E_i^{k+1} = {} & \varepsilon_i^{k+1} + 2(1-\check{a})\varepsilon_i^k + \check{a}\varepsilon_{i+1}^k + \check{a}\varepsilon_{i-1}^k + \sum_{l=0}^{k-1} \Bigg(2(1-\check{a})\alpha_{-l-1}^{l+1}\varepsilon_{i-l-1}^{k-1-l} \\ & +2(1-\check{a})\alpha_{l+1}^{l+1}\varepsilon_{i+l+1}^{k-1-l} + \check{a}\alpha_l^{l+1}\ \varepsilon_{i+l+1}^{k-1-l} + \check{a}\alpha_{l+1}^{l+1}\ \varepsilon_{i+l+2}^{k-1-l} \\ & +\check{a}\alpha_{-l}^{l+1}\ \varepsilon_{i-l-1}^{k-1-l} + \check{a}\alpha_{-l-1}^{l+1}\ \varepsilon_{i-l-2}^{k-1-l} + \sum_{j=-l}^{l} \alpha_j^{l+2}\varepsilon_{i+j}^{k-1-l} \Bigg) \end{aligned}$$

We also go the other way:

$$\begin{aligned} \sum_{l=0}^{k+1}\sum_{j=-l}^{l} \alpha_j^l\ \varepsilon_{i+j}^{k+1-l} = {} & \varepsilon_i^{k+1} + 2(1-\check{a})\varepsilon_i^k + \check{a}\varepsilon_{i+1}^k + \check{a}\varepsilon_{i-1}^k \\ & + \sum_{l=0}^{k-1} \Bigg(\alpha_{-l-2}^{l+2}\ \varepsilon_{i-l-2}^{k-1-l} + \alpha_{-l-1}^{l+2}\ \varepsilon_{i-l-1}^{k-1-l} + \alpha_{l+1}^{l+2}\ \varepsilon_{i+l+1}^{k-1-l} \\ & + \alpha_{l+2}^{l+2}\ \varepsilon_{i+l+2}^{k-1-l} + \sum_{j=-l}^{l} \alpha_j^{l+2}\ \varepsilon_{i+j}^{k-1-l} \Bigg) \end{aligned}$$

Let us compute $\Delta = E_i^{k+1} - \sum_{l=0}^{k+1}\sum_{j=-l}^{l} \alpha_j^l\ \varepsilon_{i+j}^{k+1-l}$ to prove this value is 0. We use the facts that $\alpha_i^i = \check{a}^i$ and that $\alpha_{-l-1}^l = \alpha_{l+1}^l = 0$ and $\alpha_{-l-2}^{l+1} = \alpha_{l+2}^{l+1} = 0$.

$$\begin{aligned} \Delta = {} & \sum_{l=0}^{k-1} \big(\varepsilon_{i+l+2}^{k-1-l}(\check{a}\alpha_{l+1}^{l+1} - \alpha_{l+2}^{l+2}) + \varepsilon_{i-l-2}^{k-1-l}(\check{a}\alpha_{-l-1}^{l+1} - \alpha_{-l-2}^{l+2}) \\ & +\varepsilon_{i-l-1}^{k-1-l}(2(1-\check{a})\alpha_{-l-1}^{l+1} + \check{a}\alpha_{-l}^{l+1} - \alpha_{-l-1}^{l+2}) \\ & + \varepsilon_{i+l+1}^{k-1-l}(2(1-\check{a})\alpha_{l+1}^{l+1} + \check{a}\alpha_l^{l+1} - \alpha_{l+1}^{l+2})\big) \\ = {} & \sum_{l=0}^{k-1} \varepsilon_{i-l-1}^{k-1-l}(2(1-\check{a})\alpha_{-l-1}^{l+1} + \check{a}\alpha_{-l}^{l+1} \\ & -(2(1-\check{a})\alpha_{-l-1}^{l+1} + \check{a}\alpha_{-l}^{l+1} + \check{a}\alpha_{-l-2}^{l+1} - \alpha_{-l-1}^{l})) \\ & +\varepsilon_{i+l+1}^{k-1-l}(2(1-\check{a})\alpha_{l+1}^{l+1} + \check{a}\alpha_l^{l+1} \\ & -(2(1-\check{a})\alpha_{l+1}^{l+1} + \check{a}\alpha_l^{l+1} + \check{a}\alpha_{l+2}^{l+1} - \alpha_{l+1}^{l})) \\ = {} & \sum_{l=0}^{k-1} \big(\varepsilon_{i-l-1}^{k-1-l}(-\check{a}\alpha_{-l-2}^{l+1} + \alpha_{-l-1}^{l})) + \varepsilon_{i+l+1}^{k-1-l}(-\check{a}\alpha_{l+2}^{l+1} + \alpha_{l+1}^{l}))\big) = 0 \end{aligned}$$

This rather complicated expression is proved correct. We can express the precise floating-point error with this double summation. □

3.5 ε Tossing

The previous proof assumes that the double summation is correct for all (i', k') such that $k' < k$. This would be correct if there was an infinite set of i where p_i^k is computed. The i such that p_i^k is computed are the integers between 0 and `ni`. At the ends of the range, p_i^k is exact because set to 0.

This is quite unpleasant because this means our fine recurrence fails because of these extrema. The reason is that $E_0^k = 0$, so E_0^k is *a priori* not equal to the expected double summation, except if we define ε out of the $[0; \mathtt{ni}]$ range so that $E_0^k = \sum_{l=0}^{k} \sum_{j=-l}^{l} \alpha_j^l \ \varepsilon_j^{k-l} = 0$ and $E_{\mathtt{ni}}^k = \sum_{l=0}^{k} \sum_{j=-l}^{l} \alpha_j^l \ \varepsilon_{\mathtt{ni}+j}^{k-l} = 0$.

We defined ε_i^k in the preceding Section for all k and for $0 < i < \mathtt{ni}$ as in Figure 2. We define $\varepsilon_0^k = 0$. We then define ε_i^k for all i, k in the following way:

- If $i \geq 0$, then
  - if $(i \div \mathtt{ni}) \mod 2 = 0$, then $\varepsilon_i^k = \varepsilon_{i \mod \mathtt{ni}}^k$,
  - else $\varepsilon_i^k = -\varepsilon_{(\mathtt{ni}-i) \mod \mathtt{ni}}^k$,
- else $\varepsilon_i^k = -\varepsilon_{-i}^k$.

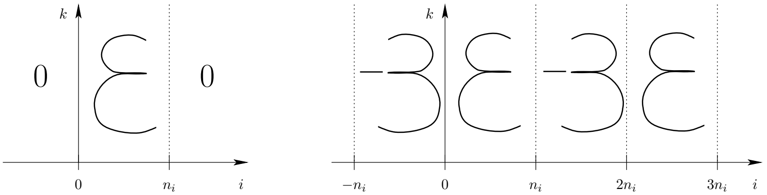


Fig. 2. Initial ε and tossed ε

On the ranges $[k \times \mathtt{ni}; (k+1) \times \mathtt{ni}]$, either we have ε (for even k), or we have a negated mirrored ε (for odd k). Figure 2 illustrates the way ε is defined on the whole range. This weird definition allows us to guarantee that the double summation is exactly zero for $i = 0$ and $i = \mathtt{ni}$. Indeed, by symmetry of α and by antisymmetry of ε:

$$E_0^k = \sum_{l=0}^{k} \sum_{j=-l}^{l} \alpha_j^l \ \varepsilon_j^{k-l} = \sum_{l=0}^{k} \left(\sum_{j=-l}^{-1} \alpha_j^l \ \varepsilon_j^{k-l} + \alpha_0^l \ \varepsilon_0^{k-l} + \sum_{j=1}^{l} \alpha_j^l \ \varepsilon_j^{k-l} \right)$$
$$= \sum_{l=0}^{k} \left(\sum_{j=1}^{l} -\alpha_j^l \ \varepsilon_j^{k-l} + 0 + \sum_{j=1}^{l} \alpha_j^l \ \varepsilon_j^{k-l} \right) = 0$$

The same kind of proof holds for $E_{\mathtt{ni}}^k = 0$. This proof trick is here only to pretend that E_0 and $E_{\mathtt{ni}}$ are equal to 0. They do not imply anything on the values of the E_i between these bounds.

3.6 Formal Proof

All the proofs described have been done and machined-checked using Coq. This allows us to formally verify the annotations of the loop invariant and the final error bound. The unproved assumptions are solved using axioms.

All the annotated programs and the corresponding Coq developments described in this article are available at `http://www.lri.fr/~sboldo/gallery.html`. They require libraries about floating-point arithmetic[1], and its formalization in the Why platform [2].

The difficulty did not lie in the floating-point part. The only floating-point proof is the 85×2^{-52} one, which is basic interval arithmetic. Nevertheless, it was not automatized and was 735 lines long. We have since developed automations that cut down this proof to 10 lines and improved the result to 80×2^{-52} [16].

The hard part (apart from finding out the analytical expression) is handling the double summation expressions. For example, we handle expressions such as $\displaystyle\sum_{l=1}^{k} \sum_{j=-l+1}^{l+1} \alpha_{j-1}^{l}\ \varepsilon_{i+j}^{k-l}$ with the following expression:

```
(sum_f_z (fun l : Z => sum_f_z (fun j : Z
   => alpha a (j - 1) (Zabs_nat l) * eps (i + j) (k - l))
    (- l + 1) (l + 1)) 1 k)%R.
```

This is rather cumbersome to handle, even if it is just following the pen and paper proof. Note that some automations and nice notations on big operators have been developed in [17]; they would have helped the proof and its readability. There is indeed nothing very technical or tricky in this formal proof, except that the loop invariant must be defined using higher order logic.

3.7 Final Error

The analytical expression of the error is in itself interesting (it may lead to know which error is dominating the others). Nevertheless, the main interest is to get a not-too-overestimated final bound for the rounding error.

Theorem 3.

$$\left|\mathbf{E_i^k}\right| = \left|\mathbf{p_i^k} - \mathbf{exact}\left(\mathbf{p_i^k}\right)\right| \leq \mathbf{85} \times \mathbf{2}^{-\mathbf{53}} \times (\mathbf{k}+\mathbf{1}) \times (\mathbf{k}+\mathbf{2})$$

Proof. Let us bound the rounding error of p_i^k, that is $|E_i^k| = \left|\sum_{l=0}^{k} \sum_{j=-l}^{l} \alpha_j^l\ \varepsilon_{i+j}^{k-l}\right|$. We know that for all j and l, $|\varepsilon_j^l| \leq 85 \times 2^{-52}$ and that $\sum_{i=-\infty}^{+\infty} \alpha_i^l = l+1$ by Lemma [1]. As the α_i^k are nonnegative, then the error is easily bounded by $85 \times 2^{-52} \times \sum_{l=0}^{k} l+1$. □

[1] Available at `http://lipforge.ens-lyon.fr/www/pff/`.

[2] Available at `http://why.lri.fr/`.

As the proof of the non-negativity of the α_i^k is nontrivial and not formally proved, the validity of this result may be put in question. Nevertheless, the α_i^k are the discretization of the differential equation with different initializations (and no rounding error). Therefore, as the $|p_i^k|$ are bounded, the $|\alpha_i^k|$ are bounded the same way. Therefore we may assume that $|\alpha_i^k| \leq 1.5$, and this permits to prove a bound on $|E_i^k|$:

$$\begin{aligned}|E_i^k| = \left|\sum_{l=0}^{k}\sum_{j=-l}^{l}\alpha_j^l \varepsilon_{i+j}^{k-l}\right| &\leq \sum_{l=0}^{k}\sum_{j=-l}^{l}|\alpha_j^l \varepsilon_{i+j}^{k-l}| \leq 85 \times 2^{-52} \times \sum_{l=0}^{k}\sum_{j=-l}^{l}|\alpha_j^l| \\ &\leq 85 \times 2^{-52} \times \frac{3}{2}\sum_{l=0}^{k}\sum_{j=-l}^{l} 1 = 255 \times 2^{-53}\sum_{l=0}^{k} 2\,l + 1 < 2^{-45} \times (k+1)^2\end{aligned}$$

This is a slightly worse bound than the previous one, but it only relies on the study of the partial differential equation.

4 Conclusion

In order to prove the program entirely, the only difficulty left is the boundedness of the discretized solution of the partial differential equation: we assumed that $|p_i^k| \leq 2$. As we bound the floating-point error, if $\texttt{nk} \leq 2^{22}$ or so, it is indeed enough to bound $|\text{exact}(p_i^k)|$ by 1.5. This last fact is due to the consistency of the numerical scheme. This is a part of the global proof of the program, that demonstrates that this programs computes an approximation of the spread of acoustic waves on a rope. This mostly tackles Coq formalization of mathematical knowledge and requires many new definitions and lemmas about scalar product, symmetrical operators, Taylor series, $f = O(g)$, O of functions of two variables... and is therefore out of the scope of this paper.

This technique of the analytical error and precise floating-point error cancellation coming with its formal proof is new. The reason is that it requires very generic specifications as the loop invariant needs to be logically defined: it states there exists a function ε that has such and such property. And Caduceus allows us to express such a high-level property on a C program. We then use Coq as a back-end to formally check the specifications. This genericity is an advantage compared to automatic methods that cannot express our loop invariant.

We have shown how the analytical error technique increases the quality of the final floating-point error on two examples. Instead of the exponential error we obtain by usual methods, we get a quadratic error: this is indeed a terrific improvement. But the price is rather high as the user has to find out the exact expression of the analytical error before proving it. The analytical expression of the second example took us a few months to be worked out. We did not find any method to infer the analytical error from the program. We plan to develop methods to find out automatically analytical expressions or hints towards analytical expressions in order to spread the use of this technique.

This technique cannot provide an error bound to all floating-point programs as it requires a readable expression for the analytical error simple enough to handle in proofs. We also intend to study this class of programs.

Acknowledgements

We thank the participants of the CerPAN and F∮st projects, namely M. Mayero, J.-C. Filliâtre and especially F. Clément for providing the C program, documentation and many explanations. We also thank B. Salvy, M. Kauers and V. Pillwein for their help (and proof!) on the non-negativity of the α_i^k.

References

1. IEEE: IEEE Standard for Floating-Point Arithmetic. IEEE Std. 754-2008 (2008)
2. Wilkinson, J.H.: Rounding Errors in Algebraic Processes. Prentice-Hall, Upper Saddle River (1963)
3. Higham, N.J.: Accuracy and stability of numerical algorithms. SIAM, Philadelphia (2002)
4. Even, G., Seidel, P., Ferguson, W.E.: A Parametric Error Analysis of Goldschmidt's Division Algorithm. In: 16th IEEE Symposium on Computer Arithmetic (2003)
5. Goubault, E., Putot, S.: Static analysis of numerical algorithms. In: Yi, K. (ed.) SAS 2006. LNCS, vol. 4134, pp. 18–34. Springer, Heidelberg (2006)
6. Dekker, T.J.: A floating point technique for extending the available precision. Numerische Mathematik 18(3), 224–242 (1971)
7. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions. Springer, Heidelberg (2004)
8. Daumas, M., Rideau, L., Théry, L.: A generic library of floating-point numbers and its application to exact computing. In: 14th International Conference on Theorem Proving in Higher Order Logics, Edinburgh, Scotland, pp. 169–184 (2001)
9. Boldo, S.: Preuves formelles en arithmétiques á virgule flottante. PhD thesis, École Normale Supérieure de Lyon (2004)
10. Filliâtre, J.C., Marché, C.: Multi-Prover Verification of C Programs. In: Sixth International Conference on Formal Engineering Methods, pp. 15–29. Springer, Heidelberg (2004)
11. Filliâtre, J.-C., Marché, C.: The Why/Krakatoa/Caduceus platform for deductive program verification. In: Damm, W., Hermanns, H. (eds.) CAV 2007. LNCS, vol. 4590, pp. 173–177. Springer, Heidelberg (2007)
12. Boldo, S., Filliâtre, J.-C.: Formal verification of floating-point programs. In: 18th IEEE Symposium on Computer Arithmetic, Montpellier, France, pp. 187–194 (2007)
13. Bécache, E.: Étude de schémas numériques pour la résolution de léquation des ondes. In: ENSTA (2003)
14. Askey, R., Gasper, G.: Certain rational functions whose power series have positive coefficients. The American Mathematical Monthly 79, 327–341 (1972)
15. Andrews, G.E., Askey, R., Roy, R.: Special functions. Cambridge University Press, Cambridge (1999)
16. Boldo, S., Filliâtre, J.-C., Melquiond, G.: Combining Coq and Gappa for Certifying Floating-Point Programs. In: 16th Symposium on the Integration of Symbolic Computation and Mechanised Reasoning (2009)
17. Bertot, Y., Gonthier, G., Biha, S.O., Pasca, I.: Canonical big operators. In: Mohamed, O.A., Muñoz, C., Tahar, S. (eds.) TPHOLs 2008. LNCS, vol. 5170, pp. 86–101. Springer, Heidelberg (2008)

Reachability in Stochastic Timed Games

Patricia Bouyer[1,⋆] and Vojtěch Forejt[2,⋆⋆]

[1] LSV, CNRS & ENS Cachan, France
bouyer@lsv.ens-cachan.fr
[2] Masaryk University, Brno, Czech Republic
forejt@fi.muni.cz

Abstract. We define stochastic timed games, which extend two-player timed games with probabilities (following a recent approach by Baier *et al*), and which extend in a natural way continuous-time Markov decision processes. We focus on the reachability problem for these games, and ask whether one of the players has a strategy to ensure that the probability of reaching a fixed set of states is equal to (or below, resp. above) a certain number r, whatever the second player does. We show that the problem is undecidable in general, but that it becomes decidable if we restrict to single-clock $1\frac{1}{2}$-player games and ask whether the player can ensure that the probability of reaching the set is $=1$ (or >0, $=0$).

1 Introduction

Timed systems. Timed automata [1] are a well-established formalism for the modelling and analysis of timed systems. A timed automaton is roughly a finite-state automaton enriched with clocks and clock constraints. This model has been extensively studied, and several verification tools have been developed. To represent interactive or open systems, the model of timed games has been proposed [2], where the system interacts with its environment, and the aim is to build a controller that will guide the system, so that it never violates its specification, whatever are the actions of the environment.

Adding probabilities to timed automata. In [3,4], a purely probabilistic semantics has been given to timed automata, in which both delays and discrete choices are randomized. The initial motivation of the previous works was not to define a model with real-time and probabilistic features, but rather to propose an alternative semantics to timed automata, following the long-running implementability and robustness paradigm [9,12,17]. The idea is that unlikely behaviours should not interfere with the validity of a formula in a timed automaton, and the probabilistic semantics has been proposed to provide a way of measuring the 'size' of sets of runs in a timed automaton. In this context, natural model-checking questions have been considered: (i) 'Does the automaton *almost-surely* satisfy a given ω-regular property?', and (ii) 'Does the automaton satisfy

[⋆] This author was partly supported by the French project DOTS (ANR-06-SETI-003) and by the European project QUASIMODO (FP7-ICT-STREP- 214755).

[⋆⋆] This author was partly supported by the Czech Science Foundation, Grant No. 102/09/H042 and by the research centre Institute for Theoretical Computer Science (ITI), project No. 1M0545.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 103–114, 2009.

a given ω-regular property with probability at least p?'. The first problem is decidable for single-clock timed automata [3], but it is open for general timed automata. The second problem is decidable for a subclass of single-clock timed automata [6].

If we consider probabilities no more as a way of providing an alternative semantics to timed automata but rather as part of the model itself, the purely stochastic model defined in [3] can be viewed as an extension of continuous-time Markov chains (CTMCs in short), which have been extensively studied, both by mathematicians [11] and by computer scientists for their role in verification [5,13].

Stochastic timed games. In real-life systems, pure stochastic models might not be sufficient, and non-determinism and even interaction with an environment might be crucial features (we might think of communication protocols, where messages can be lost, and response delays are probabilistic). In the same way continuous-time Markov decision processes extend CTMCs, we can extend the purely stochastic model of [3] with non-determinism, and even with interaction.

In this paper, we propose the model of *stochastic timed games*, which somehow extend classical timed games with probabilities. In this model, some locations are probabilistic (in some context we could say they represent the nature), and the other locations belong either to player $\Diamond$ or to player $\Box$. We call these locations respectively $\bigcirc$-locations, $\Diamond$-locations, and $\Box$-locations. Following classical terminology in stochastic finite games [8] where the nature is viewed as half a player, those games will be called $2\frac{1}{2}$*-player timed games*, and stochastic timed games with no $\Box$-locations will be called $1\frac{1}{2}$*-player timed games*. Finally, the purely stochastic model of [3] can then be called the $\frac{1}{2}$*-player game model* (there are no $\Diamond$-locations nor $\Box$-locations).

We assume a stochastic timed game is given, and we play the game as follows. At $\Diamond$-locations, player $\Diamond$ chooses the next move (delay and transitions to be taken), at $\Box$-locations, player $\Box$ chooses the next move, and at $\bigcirc$-locations, the environment is purely probabilistic (and the probability laws on delays and transitions are given in the description of the model). Moves for the two players are given by (deterministic) strategies, and given two strategies $\lambda_\Diamond$ (for player $\Diamond$) and $\lambda_\Box$ (for player $\Box$), the play of the game is a probability distribution over the set of runs of the timed automaton. Some natural questions can then be posed:

Qualitative questions: given $r \in \{0, 1\}$, is there a strategy for player $\Diamond$ such that for every strategy for player $\Box$, the probability (under those strategies) of satisfying some reachability property is equal to (resp. less than, resp. more than...) r?

Quantitative questions: given $r \in (0, 1)$, is there a strategy for player $\Diamond$ such that for every strategy for player $\Box$, the probability (under those strategies) of satisfying some reachability property is equal to (resp. no less than, resp. no more than...) r?

On that model, only restricted results have been proven so far, and they only concern the $\frac{1}{2}$-player case: all qualitative questions can be decided (in NLOGSPACE) if we restrict to single-clock models [3], and under a further restriction on the way probabilities are assigned to delays, all quantitative questions can be decided [6].

Our contribution. In this paper, we show the two following results:

- For $1\frac{1}{2}$-player games *with a single clock*, the qualitative questions 'equal to 0' or 'equal to 1' can be solved in PTIME, matching the known PTIME-hardness in

classical Markov decision processes [16], and the qualitative question 'larger than 0' can be solved in NLOGSPACE, matching the NLOGSPACE-hardness of the reachability in finite graphs;
- For $2\frac{1}{2}$-player games, the quantitative questions are undecidable. We will make precise in the core of the paper the classes of models for which this result holds.

2 Definitions

Timed automata. We assume the classical notions of clocks, clock valuations and guards are familiar to the reader [1]. We write $\mathcal{G}(X)$ for the set of diagonal-free guards over set of clocks X. A *timed automaton* is a tuple $\mathcal{A} = (L, X, E, \mathcal{I})$ such that: (i) L is a finite set of locations, (ii) X is a finite set of clocks, (iii) $E \subseteq L \times \mathcal{G}(X) \times 2^X \times L$ is a finite set of edges, and (iv) $\mathcal{I} : L \to \mathcal{G}(X)$ assigns an invariant to each location. A *state* s of such a timed automaton is a pair $(\ell, v) \in L \times (\mathbb{R}_+)^{|X|}$ (where v is a clock valuation). If $s = (\ell, v)$ is a state and $t \in \mathbb{R}_+$, we write $s + t$ for the state $(\ell, v + t)$. We say that there is a *transition* (t, e) from state $s = (\ell, v)$ to state $s' = (\ell', v')$, we then write $s \xrightarrow{t,e} s'$, if $e = (\ell, g, Y, \ell') \in E$ is such that (i) $v + t \models g$, (ii) for every $0 \le t' \le t$, $v + t' \models \mathcal{I}(\ell)$, (iii) $v' = [Y \leftarrow 0](v + t)$, and (iv) $v' \models \mathcal{I}(\ell')$. A *run* in $\mathcal{A}$ is a finite or infinite sequence $\varrho = s_0 \xrightarrow{t_1,e_1} s_1 \xrightarrow{t_2,e_2} s_2 \cdots$ of states and transitions. An edge e is *enabled* in state s whenever there is a state s' such that $s \xrightarrow{0,e} s'$. Given s a state of $\mathcal{A}$ and e an edge, we define $I(s, e) = \{t \in \mathbb{R}_+ \mid s \xrightarrow{t,e} s' \textit{ for some } s'\}$ and $I(s) = \bigcup_{e \in E} I(s, e)$. The automaton $\mathcal{A}$ is *non-blocking* if for all states s, $I(s) \neq \emptyset$. For the sake of simplicity, we assume that timed automata are non-blocking.

Stochastic timed games. A *stochastic timed game* is a tuple $\mathcal{G} = (\mathcal{A}, (L_\Diamond, L_\Box, L_\bigcirc), w, \mu)$ where $\mathcal{A} = (L, X, E, \mathcal{I})$ is a timed automaton, $(L_\Diamond, L_\Box, L_\bigcirc)$ is a partition of L into the locations controlled by player $\Diamond$, $\Box$ and $\bigcirc$, respectively, w is a function that assigns to each edge leaving a location in $L_\bigcirc$ a positive (integral) *weight*, and μ is a function that assigns to each state $s \in L_\bigcirc \times (\mathbb{R}_+)^{|X|}$ a measure over $I(s)$, such that for all such s, $\mu(s)$ satisfies the following requirements:

1. $\mu(s)(I(s)) = 1$;
2. We write χ for the Lebesgue measure. If $\chi(I(s)) > 0$, $\mu(s)$ is equivalent[1] to χ. Furthermore, the choice of the measures should not be too erratic and those measures should evolve smoothly when moving states. We thus require that for every $a < b$, for every s, there is some $\varepsilon > 0$ such that $\mu(s + \delta)((a - \delta, b - \delta))$ is lower-bounded by ε on the set $\{\delta \in \mathbb{R}_+ \mid (a - \delta, b - \delta) \subseteq I(s + \delta)\}$. If $\chi(I(s)) = 0$, the set $I(s)$ is finite, and $\mu(s)$ must be equivalent to the uniform distribution over points of $I(s)$.

Note that these conditions are required, see [3], but can be easily satisfied. For instance, a timed automaton with uniform distributions on bounded sets and with exponential distributions on unbounded intervals (with a smoothly varying rate, see [10]) satisfies these conditions. Also note that we impose no requirements on the representation of the measures. All our results hold regardless of the representation.

[1] Measures χ_1 and χ_2 are *equivalent* if for all measurable sets A, $\chi_1(A) = 0 \Leftrightarrow \chi_2(A) = 0$.

In the following, we will say that a timed automaton is *equipped with uniform distributions over delays* if for every state s, $I(s)$ is bounded, and $\mu(s)$ is the uniform distribution over $I(s)$. We will say that the automaton is *equipped with exponential distributions over delays* whenever, for every s, either $I(s)$ has zero Lebesgue measure, or $I(s) = \mathbb{R}_+$ and for every location ℓ, there is a positive rational number α_ℓ such that $\mu(s)(I) = \displaystyle\int_{t \in I} \alpha_\ell \cdot e^{-\alpha_\ell t}\,\mathrm{d}t$.

Intuitively, in a stochastic game, locations in $L_\Diamond$ (resp. $L_\Box$) are controlled by player $\Diamond$ (resp. player $\Box$), whereas locations in $L_\bigcirc$ belong to the environment and behaviours from those locations are governed by probabilistic laws. Indeed, in these locations, both delays and discrete moves will be chosen probabilistically: from s, a delay t is chosen following the *probability distribution over delays* $\mu(s)$; Then, from state $s + t$ an enabled edge is selected following a discrete probability distribution that is given in a usual way with the weight function w: in state $s + t$, the probability of edge e (if enabled) is $w(e)/\left(\sum_{e'}\{w(e') \mid e' \text{ enabled in } s+t\}\right)$. This way of probabilizing behaviours in timed automata has been presented in [4,3], where all locations were supposed to be probabilistic. We now formalize the *stochastic process* that is defined by a stochastic game, when fixing strategies for the two players.

A *strategy* for player $\Diamond$ (resp. player $\Box$) is a function that assigns to every finite run $\varrho = (\ell_0, v_0) \xrightarrow{t_1,e_1} \ldots \xrightarrow{t_n,e_n} (\ell_n, v_n)$ with $\ell_n \in L_\Diamond$ (resp. $\ell_n \in L_\Box$) a pair $(t, e) \in \mathbb{R}_+ \times E$ such that $(\ell_n, v_n) \xrightarrow{t,e} (\ell, v)$ for some (ℓ, v). In order to later be able to measure probabilities of certain sets of runs, we impose the following additional measurability condition on the strategy λ: for every finite sequence of edges $e_1, \ldots, e_n$ and every state s, the function $\kappa : (t_1, \ldots, t_n) \mapsto (t, e)$ such that $\kappa(t_1, \ldots, t_n) = (t, e)$ iff $\lambda(s \xrightarrow{t_1,e_1} s_1 \ldots \xrightarrow{t_n,e_n} s_n) = (t, e)$ is measurable.[2]

A *strategy profile* is a pair $\Lambda = (\lambda_\Diamond, \lambda_\Box)$ where $\lambda_\Diamond$ and $\lambda_\Box$ are strategies for players $\Diamond$ and $\Box$ respectively. Given a stochastic timed game $\mathcal{G}$, a finite run ϱ ending in a state s_0 and a strategy profile $\Lambda = (\lambda_\Diamond, \lambda_\Box)$, we define $\mathit{Run}(\mathcal{G}, \varrho, \Lambda)$ (resp. $\mathit{Run}^\omega(\mathcal{G}, \varrho, \Lambda)$) to be the set of all finite (resp. infinite) runs generated by $\lambda_\Diamond$ and $\lambda_\Box$ after prefix ϱ, *i.e.*, the set of all runs $s_0 \xrightarrow{t_1,e_1} s_1 \xrightarrow{t_2,e_2} \cdots$ in the underlying automaton satisfying the following condition: if $s_i = (\ell, v)$ and $\ell \in L_\Diamond$ (resp. $\ell \in L_\Box$), then $\lambda_\Diamond$ (resp. $\lambda_\Box$) returns (t_{i+1}, e_{i+1}) when applied to $\varrho \xrightarrow{t_1,e_1} s_1 \xrightarrow{t_2,e_2} \ldots \xrightarrow{t_i,e_i} s_i$. Moreover, given a finite sequence of edges $e_1, \ldots, e_n$, we define the *symbolic path* $\pi_\Lambda(\varrho, e_1 \ldots e_n)$ by

$$\pi_\Lambda(\varrho, e_1 \ldots e_n) = \{\varrho' \in \mathit{Run}(\mathcal{G}, \varrho, \Lambda) \mid \varrho' = s_0 \xrightarrow{t_1,e_1} \cdots \xrightarrow{t_n,e_n} s_n,\ t_i \in \mathbb{R}_+\}$$

When Λ is clear from the context, we simply write $\pi(\varrho, e_1 \ldots e_n)$.

We extend the definitions of [3] to stochastic games, and define, given a strategy profile $\Lambda = (\lambda_\Diamond, \lambda_\Box)$ and a finite run ϱ ending in $s = (\ell, v)$, a measure $\mathcal{P}_\Lambda$ over the set $\mathit{Run}(\mathcal{G}, \varrho, \Lambda)$. To that aim, we define $\mathcal{P}_\Lambda$ on symbolic paths initiated in ϱ by $\mathcal{P}_\Lambda(\pi(\varrho)) = 1$ and then inductively as follows:

[2] For the purpose of this definition, we define the measurable space on the domain (and codomain) as a product space of measurable spaces of its components (where for real numbers and edges we take the σ-algebra generated by intervals and by set of edges, respectively).

- If $\ell \in L_\Diamond$ (resp. $\ell \in L_\Box$) and $\lambda_\Diamond(\varrho) = (t, e)$ (resp. $\lambda_\Box(\varrho) = (t, e)$), we set

$$\mathcal{P}_\Lambda(\pi(\varrho, e_1 \dots e_n)) = \begin{cases} 0 & \text{if } e_1 \neq e \\ \mathcal{P}_\Lambda(\pi(\varrho \xrightarrow{t,e} s', e_2 \dots e_n)) & \text{otherwise (where } s \xrightarrow{t,e} s'\text{)} \end{cases}$$

- If $\ell \in L_\bigcirc$, we define

$$\mathcal{P}_\Lambda(\pi(\varrho, e_1 \dots e_n)) = \int_{t \in I(s,e_1)} p(s+t)(e_1) \cdot \mathcal{P}_\Lambda(\pi(\varrho \xrightarrow{t,e_1} s^{t,e_1}, e_2 \dots e_n))\, \mathrm{d}\mu(s)(t)$$

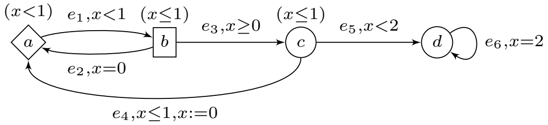
  where $s \xrightarrow{t,e_1} s^{t,e_1}$ for every $t \in I(s, e_1)$.

These integrals are properly defined thanks to the measurability condition we impose on strategies, and thanks to Fubini's Theorem [19].

Following [3], it is not difficult to see that, given a measurable constraint $\mathcal{C}$ of $\mathbb{R}_+^n$, we can extend this definition to *constrained symbolic paths* $\pi_\Lambda^\mathcal{C}(\varrho, e_1 \dots e_n)$, where $\pi_\Lambda^\mathcal{C}(\varrho, e_1 \dots e_n) = \{\varrho' \in Run(\mathcal{G}, \varrho, \Lambda) \mid \varrho' = s_0 \xrightarrow{t_1,e_1} \cdots \xrightarrow{t_n,e_n} s_n \text{ and } (t_1, \dots, t_n) \models \mathcal{C}\}$. We now consider the *cylinder* generated by a constrained symbolic path: an infinite run ϱ'' is in the cylinder generated by $\pi_\Lambda^\mathcal{C}(\varrho, e_1 \dots e_n)$, denoted $\mathsf{Cyl}(\pi_\Lambda^\mathcal{C}(\varrho, e_1 \dots e_n))$, if $\varrho \in Run^\omega(\mathcal{G}, \varrho, \Lambda)$ and there exists $\varrho' \in \pi_\Lambda^\mathcal{C}(\varrho, e_1 \dots e_n)$ which is a prefix of ϱ''. We extend $\mathcal{P}_\Lambda$ to those cylinders in a natural way: $\mathcal{P}_\Lambda(\mathsf{Cyl}(\pi_\Lambda^\mathcal{C}(\varrho, e_1 \dots e_n))) = \mathcal{P}_\Lambda(\pi_\Lambda^\mathcal{C}(\varrho, e_1 \dots e_n))$, and then in a unique way to the σ-algebra Ω_Λ^ϱ generated by those cylinders. Following [3], we can prove the following correctness lemma.

Lemma 1. *Let $\mathcal{G}$ be a stochastic timed game. For every strategy profile Λ, for every finite run ϱ, $\mathcal{P}_\Lambda$ is a probability measure over $(Run^\omega(\mathcal{G}, \varrho, \Lambda), \Omega_\Lambda^\varrho)$.*

Example 2. Consider the following game $\mathcal{G}$:



Suppose the game is equipped with uniform distributions over delays and over edges, and consider a strategy profile $\Lambda = (\lambda_\Diamond, \lambda_\Box)$ such that strategy $\lambda_\Diamond$ assigns $(0.5, e_1)$ to each run ϱ ending in state (a, v) if $v \leq 0.5$ and $(0, e_1)$ otherwise, and such that strategy $\lambda_\Box$ assigns $(0, e_3)$ to each run ending in (b, v). If $\varrho = (a, 0) \xrightarrow{0.5,e_1} (b, 0.5) \xrightarrow{0,e_3} (c, 0.5)$, $\mathcal{P}_\Lambda(\pi(\varrho, e_4 e_1 e_3 e_4)) = \frac{1}{36}$. ⌟

The reachability problem. In this paper we study *the reachability problem* for stochastic games, which is stated as follows. Given a game $\mathcal{G}$, an initial state s, a set of locations A, a comparison operator $\sim \in \{<, \leq, =, \geq, >\}$ and a rational number $r \in [0, 1]$, decide whether there is a strategy $\lambda_\Diamond$ for player $\Diamond$, such that for every strategy $\lambda_\Box$ for player $\Box$, if $\Lambda = (\lambda_\Diamond, \lambda_\Box)$, $\mathcal{P}_\Lambda\{\varrho \in Run(\mathcal{G}, s, \Lambda) \mid \varrho \text{ visits } A\} \sim r$. In that case, we say that $\lambda_\Diamond$ is a *winning strategy* from s for the *reachability objective* $\mathsf{Reach}_{\sim r}(A)$.

A special case is when $r \in \{0, 1\}$, and the problem is called the *qualitative reachability problem*. In all other cases, we speak of the *quantitative reachability problem*.

Example 3. Consider again the stochastic timed game $\mathcal{G}$ of Example 2 together with the qualitative reachability objective $\mathsf{Reach}_{=1}(\{d\})$. Player $\Diamond$ has a winning strategy $\lambda_\Diamond$ for that objective from state $(a, 0)$, which is defined as follows: $\lambda_\Diamond(\varrho) = (0.5, e_1)$ for all runs ϱ ending in state $(a, 0)$. On the other hand, player $\Diamond$ has no winning strategy from $(a, 0)$ for the quantitative objective $\mathsf{Reach}_{<0.9}(\{c\})$. ⌟

The region automaton abstraction. The well-known region automaton construction is a finite abstraction of timed automata which can be used for verifying many properties like ω-regular untimed properties [1]. In this paper, we will only use this abstraction in the context of single-clock timed automata, where the original abstraction can be slightly improved [14]. Furthermore, we will still interpret this abstraction as a timed automaton, as it is done in [3].

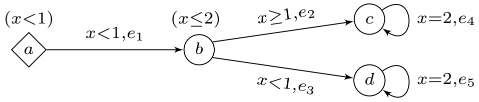
Let $\mathcal{A}$ be a single-clock timed automaton, and $\Gamma = \{0 = \gamma_0 < \gamma_1 < \cdots < \gamma_p\}$ be the set of constants that appear in $\mathcal{A}$ (plus the constant 0). We define the set $R_\mathcal{A}$ of regions in $\mathcal{A}$ as the set of intervals of the form $[\gamma_i; \gamma_i]$ (with $0 \leq i \leq p$), or $(\gamma_{i-1}; \gamma_i)$ (with $1 \leq i \leq p$) or $(\gamma_p; +\infty)$. Assuming $\mathcal{A} = (L, \{x\}, E, \mathcal{I})$, the *region automaton* of $\mathcal{A}$ is the timed automaton $\mathcal{R}(\mathcal{A}) = (Q, \{x\}, T, \kappa)$ such that $Q = L \times R_\mathcal{A}$, $\kappa((\ell, r)) = \mathcal{I}(\ell)$, and $T \subseteq (Q \times R_\mathcal{A} \times E \times 2^X \times Q)$ is such that $(\ell, r) \xrightarrow{r'', e, Y} (\ell', r')$ is in T iff there exists $e \in E$, $v \in r$, $\tau \in \mathbb{R}_+$ such that $v + \tau \in r''$, $(\ell, v) \xrightarrow{\tau, e} (\ell', v')$, and $v' \in r'$.

In the case of single-clock timed automata, the above automaton $\mathcal{R}(\mathcal{A})$ has size polynomial in the size of $\mathcal{A}$ (the number of regions is polynomial), and the reachability problem is indeed NLOGSPACE-complete in single-clock timed automata [14]. In the following, we will assume w.l.o.g. that timed automata are given in their region automaton form. Hence, to every location of this automaton will be associated a single region in which the valuation will be when arriving in that location.

3 Qualitative Reachability in Single-Clock $1\frac{1}{2}$-Player Games

In this section we restrict to single-clock $1\frac{1}{2}$-player games, *i.e.*, stochastic games with a single clock, and with no locations for player $\Box$. Furthermore, we focus on the qualitative reachability problems.

Optimal strategies may not exist. Indeed, it may be the case that for every $\varepsilon > 0$, there is a strategy achieving the (reachability) objective with probability at least $1 - \varepsilon$ (resp. at most ε), while there is no strategy achieving the objective with probability 1 (resp. 0). In this case, we speak about *ε-optimal strategies*. For instance, consider the following game, where we assume uniform distributions over delays.



Assuming that the objective is to reach location c (resp. d) from $(a, 0)$, one can check that by taking the edge e_1 close enough to time 1, the probability of reaching c (resp. d) can be arbitrary close to 1 (resp. 0), while there is no strategy that could ensure reaching c (resp. d) with probability 1 (resp. 0).

In this paper we will ask whether there are strategies that precisely achieve a qualitative objective (like equal to 1, or equal to 0), and we leave for future work the interesting but difficult question whether we can approximate arbitrarily these objectives or not.

Decidability of the existence of optimal strategies. We now turn to one of the two main theorems of this paper, whose proof will be developed.

Theorem 4. *Given a single-clock $1\frac{1}{2}$-player timed game $\mathcal{G}$, $s = (\ell, 0)$ a state of $\mathcal{G}$, and A a set of locations of $\mathcal{G}$, we can decide in PTIME whether there is a strategy achieving the objective* $\mathsf{Reach}_{=1}(A)$ *(or* $\mathsf{Reach}_{=0}(A)$*). We can decide in NLOGSPACE whether there is a strategy achieving the objective* $\mathsf{Reach}_{>0}(A)$. *These complexity upper bounds are furthermore optimal.*

For the rest of the section, we assume that $\mathcal{G}$ is a single-clock $1\frac{1}{2}$-player timed game with the underlying automaton being a region automaton. We also fix a set A of locations. Computing winning states for the objectives $\mathsf{Reach}_{=0}(A)$ and $\mathsf{Reach}_{>0}(A)$ can be performed using a simple fixpoint algorihm (in fact, the problem can be reduced to the similar problem for discrete-time Markov decision processes [18]). [3]

Proposition 5. *We can compute in PTIME (resp. NLOGSPACE) the set of states from which player $\Diamond$ has a strategy to achieve the objective* $\mathsf{Reach}_{=0}(A)$ *(resp.* $\mathsf{Reach}_{>0}(A)$. *Furthermore, this set of states is closed by region (i.e., if (ℓ, v) is winning, then for every v' in the same region as v, (ℓ, v') is winning).*

The case of the objective $\mathsf{Reach}_{=1}(A)$ requires more involved developments, but a proposition identical to the previous one can however be stated.

Proposition 6. *We can compute in PTIME the set of states from which player $\Diamond$ has a strategy to achieve the objective* $\mathsf{Reach}_{=1}(A)$. *Furthermore, this set of states is closed by region.*

The restriction to single-clock games yields the following important property: resetting the unique clock somehow resets the history of the game, the target state is then uniquely determined by the target location. Hence, we will first focus on games where the clock is never reset, and then decompose the global game w.r.t. the resetting transitions and solve the different non-resetting parts separately and glue everything together.

We first focus on games without any resets, and consider a more complex objective: given two sets of locations A and B such that $B \subseteq A$, we say that the strategy λ achieves the objective $\mathsf{ExtReach}(A, B)$ if it achieves both $\mathsf{Reach}_{=1}(A)$ and $\mathsf{Reach}_{>0}(B)$. We can prove (using another fixpoint algorithm):

[3] All propositions in this section make use of the fact that we can remove w.l.o.g. certain "negligible" edges from the game effectively. An edge e of $\mathcal{G}$ is said to be *negligible* if it starts from some $\bigcirc$-location ℓ and if it is constrained by some punctual constraint, whereas there is another edge leaving ℓ labelled with a non-punctual constraint.

Lemma 7. *We assume that the clock is never reset in $\mathcal{G}$. Let A and $B \subseteq A$ be two sets of locations of $\mathcal{G}$. We can compute in PTIME the set of states from which player $\Diamond$ has a strategy to achieve the objective* $\mathsf{ExtReach}(A, B)$. *Furthermore, this set of states is closed by region.*

Now we show how we can use Lemma 7 to solve the games for the objective $\mathsf{Reach}_{=1}(A)$. This lemma heavily relies on the specific properties of single-clock timed automata that we have mentioned earlier. Somehow to solve the objective $\mathsf{Reach}_{=1}(A)$, we will enforce moving from one resetting transition to another one, always progressing towards A. This is formalized as follows.

Lemma 8. *If ℓ_{in} is a location of $\mathcal{G}$, the following two statements are equivalent:*

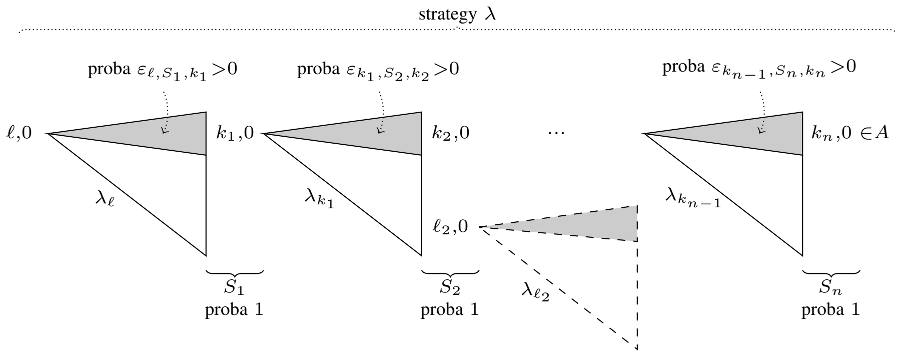
1. *There is a strategy λ from $(\ell_{in}, 0)$ that achieves the objective* $\mathsf{Reach}_{=1}(A)$.
2. *Writing L_0 for the set of locations which are targets of resetting transitions (w.l.o.g. we assume $\ell_{in} \in L_0$ and $A \subseteq L_0$), there is a set $R \subseteq L_0 \times 2^{L_0} \times L_0$ such that:*
   - *(a) There is $(\ell_{in}, S, k) \in R$ for some $S \subseteq L_0$ and $k \in S$;*
   - *(b) Whenever $\ell \in S \setminus A$ for some $(\ell', S, k) \in R$, then $(\ell, S', k') \in R$ for some $S' \subseteq L_0$ and $k' \in S'$;*
   - *(c) For each $(\ell, S, k) \in R$, there is a strategy that achieves* $\mathsf{ExtReach}(S, \{k\})$ *from* $(\ell, 0)$ *without resetting the clock (except for the last move to S);*
   - *(d) For each $(\ell, S, k) \in R$, there is a sequence $k_1 k_2 \ldots k_n$ such that $k_1 = \ell$, $k_n \in A$, and for every $1 \leq i < n$, there exist $S_{i+1} \subseteq L_0$ and $k_{i+1} \in S_{i+1}$ such that $(k_i, S_{i+1}, k_{i+1}) \in R$.*

Moreover, the set R has polynomial size and can be computed in polynomial time.

We define some vocabulary before turning to the proof. If such an above relation R exists, we write $L_R = \{\ell \in L_0 \mid \exists S \subseteq L_0 \text{ and } k \in S \text{ s.t. } (\ell, S, k) \in R\}$. For every $\ell \in L_R$, we call the *distance to A* from ℓ the smallest integer n such that there is a chain leading to A, as in condition 2d. For every $\ell \in L_R$, the distance to A is a natural number. Furthermore, for every $\ell \in L_R$, there is $(\ell, S, k) \in R$ such that the distance to A from k is (strictly) smaller than the distance to A from ℓ.

Proof (sketch). We only justify the implication *2.* ⇒ *1.*, which gives a good intuition for the construction. We start by fixing some $(\ell, S, k) \in R$ for every $\ell \in L_R$ such that the distance to A from k is (strictly) smaller than the distance to A from ℓ. Let λ_ℓ be a (fixed) strategy that achieves the objective $\mathsf{ExtReach}(S, \{k\})$ from state $(\ell, 0)$. From these strategies we construct a strategy λ that achieves $\mathsf{Reach}_{=1}(A)$ from $(\ell_{in}, 0)$ as follows.

We let $\varrho = s_0 \xrightarrow{t_1, e_1} s_1 \xrightarrow{t_2, e_2} \cdots \xrightarrow{t_{n-1}, e_{n-1}} s_n$ be a finite run in $\mathcal{G}$, and set ϱ' as the longest suffix of ϱ which does not reset the clock. ϱ' starts in some state $s_i = (\ell_i, 0)$. If $\ell_i \in L_R$, we define $\lambda(\varrho)$ as $\lambda_{\ell_i, S_i, k_i}(\varrho')$, and otherwise we define it in an arbitrary manner (but the set of runs for which we will need to define the strategy in an arbitrary manner has probability 0). The intuitive meaning of the definition is depicted in the following picture.



It can be proven using standard tools of probability theory that λ achieves the objective. Indeed, if $\ell \in L_R$ and (ℓ, S, k) has been selected, from state $(\ell, 0)$, runs generated by λ almost-surely resets the clock, reaching states $(\ell', 0)$ with $\ell' \in S$, and with positive probability, say $\varepsilon_{\ell,S,k} > 0$, reach state $(k, 0)$. Furthermore, the distance to A from k is smaller than that from ℓ. Now, due to condition 2d, the probability to reach A from $(\ell, 0)$ is at least the positive product $\varepsilon_{\ell,S_1,k_1} \cdot \varepsilon_{k_1,S_2,k_2} \cdots \varepsilon_{k_{n-1},S_n,k_n}$. Hence, there exists $\varepsilon > 0$ such that the probability to reach A from any $(\ell, 0)$ (such that there is some $(\ell, S, k) \in R$) is at least ε. We can now conclude, by saying that from any $(\ell, 0)$ with $\ell \in L_R$, the probability to reach $\{(\ell', 0) \mid \ell' \in L_R\}$ is 1. Hence, with probability 1 we reach A under strategy λ. ⌟

4 Quantitative Reachability in $2\frac{1}{2}$-Player Games

In this section we present the second main theorem of this paper.

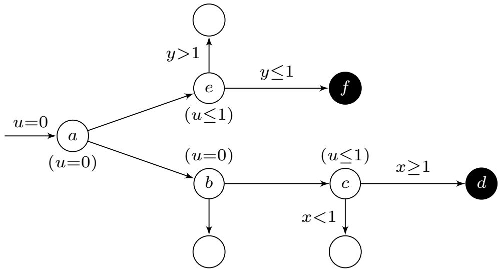
Theorem 9. *Given a $2\frac{1}{2}$-player timed game $\mathcal{G}$, $s = (\ell, 0)$ a state of $\mathcal{G}$, and A a set of locations of $\mathcal{G}$, we cannot decide whether there is a winning strategy from s for achieving the objective* $\mathsf{Reach}_{\sim\frac{1}{2}}(A)$ *(for $\sim \in \{<, \leq, =, \geq, >\}$). This result holds for games with three clocks.*

We will prove this theorem by reduction from the halting problem for two-counter machines. The reduction has been inspired by recent developments in weighted timed systems [7]. A *two-counter machine* $\mathcal{M}$ is a finite set of labeled instructions $1{:}\mathsf{inst}_1, \ldots,$ $n{-}1{:}\mathsf{inst}_{n-1}$, $n{:}\mathsf{stop}$ where each inst_i is of the form "$c_j := c_j + 1$; *goto* k" or "*if* $c_j = 0$ *then goto* k; *else* $c_j := c_j - 1$; *goto* ℓ". Here, $j \in \{1, 2\}$ and $k, \ell \in \{1, \ldots, n\}$). A configuration of $\mathcal{M}$ is a triple $[\mathsf{inst}_i, d_1, d_2]$ where inst_i is the instruction to be executed and $d_1, d_2 \in \mathbb{N}$ are the current values of the counters c_1 and c_2. A computational step $\rightsquigarrow$ between configurations is defined in the expected way. A computation is a (finite or infinite) sequence $\alpha_1, \alpha_2, \ldots$ where $\alpha_1 = [\mathsf{inst}_1, 0, 0]$ and for all i, $\alpha_i \rightsquigarrow \alpha_{i+1}$. A halting computation is a finite computation that ends in the instruction stop. The halting problem asks, given a two counter machine, whether there is a halting computation. This problem is known to be undecidable [15].

Let $\mathcal{M}$ be a two-counter machine. We construct a timed game $\mathcal{G}$ with three clocks and uniform distributions over delays, and a set of (black) locations A such that player $\Diamond$ has a strategy to reach A with probability $\frac{1}{2}$ iff $\mathcal{M}$ has a halting computation. In $\mathcal{G}$, player $\Diamond$ will simulate a computation of $\mathcal{M}$ and the values of both counters will be represented as the value of one clock. More precisely, if the values of the two counters are p_1 and p_2 respectively, the correct representation will be $\frac{1}{2^{p_1} \cdot 3^{p_2}}$. At the same time, player $\Box$ will be allowed to check that the representation is correct and "faithful" (*i.e.* that if an instruction of $\mathcal{M}$ is simulated, then the value of the counter is changed appropriately). Due to the lack of space, we are unable to provide the whole construction here, we only give a brief insight by defining some gadgets that we use in the construction.

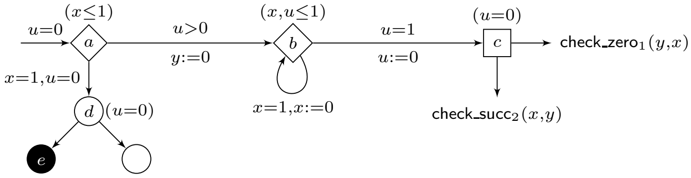
Note that in the gadgets, the letters x and y are clock variables. Later on, we will build up a game by instantiating the clock variables with real clocks. In all the gadgets, unless specified, the weight of each edge is 1 (so that when two edges are concurrently enabled, they are equally probable). From all states, the set of possible delays is bounded, hence we assume uniform distribution over delays everywhere. Finally, v_0 denotes the valuation assigning x_0 (resp. y_0, 0) to x (resp. y, u).

First, we define gadgets $\mathsf{check\_succ}_1(x, y)$ and $\mathsf{check\_succ}_2(x, y)$. These gadgets are used for testing whether the values of clocks x and y are of the form α and $\frac{\alpha}{2}$ for some $\alpha \in [0, 1]$ (in the case of $\mathsf{check\_succ}_1(x, y)$), or α and $\frac{\alpha}{3}$ for some $\alpha \in [0, 1]$ (in the case of $\mathsf{check\_succ}_2(x, y)$). We will later use these gadgets to check that an increment or a decrement of the counters has been faithfully made. The gadget $\mathsf{check\_succ}_1(x, y)$ has only probabilistic locations and has the following structure:



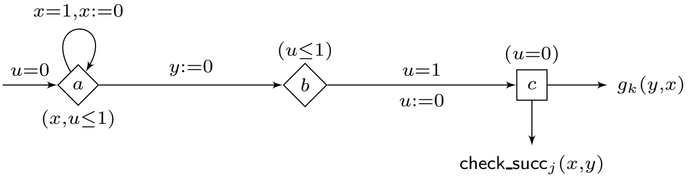
We claim that in gadget $\mathsf{check\_succ}_1(x, y)$, the probability of reaching the black locations from (a, v_0) is $\frac{1}{2}$ iff $x_0 = 2y_0$, because the probability of reaching one of the black locations is $\frac{1}{2}(1 - y_0) + \frac{1}{4}x_0$. The gadget $\mathsf{check\_succ}_2(x, y)$ can be created from $\mathsf{check\_succ}_1(x, y)$ by changing the weights of the edges.

Next, we define gadgets $\mathsf{check\_zero}_1(x, y)$ and $\mathsf{check\_zero}_2(x, y)$ that are used for testing that the value stored in clock x is $\frac{1}{3^p}$ for some $p \geq 0$ in the case of $\mathsf{check\_zero}_1$ (x, y), or $\frac{1}{2^p}$ for some $p \geq 0$ in the case of $\mathsf{check\_zero}_2(x, y)$. These gadgets will later be used to check that the value of the first or second counter is zero. The gadget $\mathsf{check\_zero}_1(x, y)$ has the following structure:



We claim that in the gadget $\mathsf{check\_zero}_1(x, y)$, player $\Diamond$ has a strategy from (a, v_0) for reaching the black locations with probability $\frac{1}{2}$ iff there is some integer $p \geq 0$ such that $x_0 = \frac{1}{3^p}$. The idea is that the value x_0 is of the required form iff it is possible to iteratively multiply its value by 2 until we eventually reach the value 1 (in which case we can take the edge down to d from a). The fact that we multiply by 2 is ensured by the gadget $\mathsf{check\_succ}_2(x, y)$. The gadget $\mathsf{check\_zero}_2(x, y)$ can be defined similarly and the precise definition is omitted here.

Finally for each instruction inst_i we create a gadget $g_i(x, y)$, that will simulate the instruction, with the encoding of the counters given earlier. For instance, if inst_i is of the form '$c_j := c_j + 1;$ goto k', then $g_i(x, y)$ is of the form:



Player $\Diamond$ chooses at which time the transition from a to b is taken, and it should be such that the value of y when entering c is properly linked with the value of x so that the incrementation of counter c_j has been simulated. The gadget for a decrementation follows similar ideas, but is a bit more technical, that is why we omit it here.

Remark 10. The above reduction is for a reachability objective of the form $\mathsf{Reach}_{=\frac{1}{2}}(A)$. However, we can twist the construction and have the reachability objective $\mathsf{Reach}_{\sim\frac{1}{2}}(A)$ (for any $\sim \in \{<, \leq, \geq, >\}$). Also, the construction can be twisted to get the following further undecidability results:

1. The value $\frac{1}{2}$ in the previous construction was arbitrary, and the construction could be modified so that it would work for any rational number $r \in (0, 1)$.
2. Instead of assuming uniform distributions over delays, one can assume unbounded intervals and exponential distributions over delays: it only requires one extra clock in the reduction.

5 Conclusion

In this paper, we have defined stochastic timed games, an extension of two-player timed games with stochastic aspects. This $2\frac{1}{2}$-player model can also be viewed as an extension of continuous-time Markov decision processes, and also of the purely stochastic model proposed in [3]. On that model, we have considered the qualitative and quantitative reachability problems, and have proven that the qualitative reachability problem can be decided in single-clock $1\frac{1}{2}$-player model, whereas the quantitative reachability problem is undecidable in the (multi-clock) $2\frac{1}{2}$-player model. This leaves a wide range of open problems. Another challenge is the computation of approximate almost-surely winning strategies (that is for every $\varepsilon > 0$, a strategy for player $\Diamond$ which ensures the reachability objective with probability larger than $1 - \varepsilon$). Finally, more involved objectives (like ω-regular or parity objectives) should be explored in the context of $1\frac{1}{2}$-player timed games.

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
2. Asarin, E., Maler, O., Pnueli, A., Sifakis, J.: Controller synthesis for timed automata. In: Proc. IFAC Symposium on System Structure and Control, pp. 469–474. Elsevier, Amsterdam (1998)
3. Baier, C., Bertrand, N., Bouyer, P., Brihaye, T., Größer, M.: Almost-sure model checking of infinite paths in one-clock timed automata. In: Proc. 23rd Annual Symposium on Logic in Computer Science (LICS 2008), pp. 217–226. IEEE Computer Society Press, Los Alamitos (2008)
4. Baier, C., Bertrand, N., Bouyer, P., Brihaye, T., Größer, M.: Probabilistic and topological semantics for timed automata. In: Arvind, V., Prasad, S. (eds.) FSTTCS 2007. LNCS, vol. 4855, pp. 179–191. Springer, Heidelberg (2007)
5. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.-P.: Model-checking algorithms for continuous-time Markov chains. IEEE Transactions on Software Engineering 29(7), 524–541 (2003)
6. Bertrand, N., Bouyer, P., Brihaye, T., Markey, N.: Quantitative model-checking of one-clock timed automata under probabilistic semantics. In: Proc. 5th International Conference on Quantitative Evaluation of Systems (QEST 2008). IEEE Computer Society Press, Los Alamitos (2008)
7. Bouyer, P., Brihaye, T., Markey, N.: Improved undecidability results on weighted timed automata. Information Processing Letters 98(5), 188–194 (2006)
8. Chatterjee, K., Jurdziński, M., Henzinger, T.A.: Simple stochastic parity games. In: Baaz, M., Makowsky, J.A. (eds.) CSL 2003. LNCS, vol. 2803, pp. 100–113. Springer, Heidelberg (2003)
9. De Wulf, M., Doyen, L., Markey, N., Raskin, J.-F.: Robust safety of timed automata. Formal Methods in System Design (to appear) (2008)
10. Desharnais, J., Panangaden, P.: Continuous stochastic logic characterizes bisimulation of continuous-time Markov processes. Journal of Logic and Algebraic Programming 56, 99–115 (2003)
11. Feller, W.: An Introduction to Probability Theory and Its Applications. John Wiley, Chichester (1968)
12. Gupta, V., Henzinger, T.A., Jagadeesan, R.: Robust timed automata. In: Maler, O. (ed.) HART 1997, vol. 1201, pp. 331–345. Springer, Heidelberg (1997)
13. Hermanns, H., Katoen, J.-P., Meyer-Kayser, J., Siegle, M.: A tool for model-checking Markov chains. International Journal on Software Tools for Technology Transfer 4, 153–172 (2003)
14. Laroussinie, F., Markey, N., Schnoebelen, P.: Model checking timed automata with one or two clocks. In: Gardner, P., Yoshida, N. (eds.) CONCUR 2004, vol. 3170, pp. 387–401. Springer, Heidelberg (2004)
15. Minsky, M.: Computation: Finite and Infinite Machines. Prentice Hall International, Englewood Cliffs (1967)
16. Papadimitriou, C.H., Tsitsiklis, J.N.: On stochastic scheduling with in-tree precedence constraints. SIAM Journal on Computing 16(1), 1–6 (1987)
17. Puri, A.: Dynamical properties of timed automata. In: Ravn, A.P., Rischel, H. (eds.) FTRTFT 1998. LNCS, vol. 1486, pp. 210–227. Springer, Heidelberg (1998)
18. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons, Chichester (1994)
19. Rudin, W.: Real and Complex Analysis. McGraw-Hill, New York (1966)

Equations Defining the Polynomial Closure of a Lattice of Regular Languages

Mário J.J. Branco[1] and Jean-Éric Pin[2,⋆]

[1] CAUL and Dep. Matemática da Faculdade de Ciências, Universidade de Lisboa Av. Prof. Gama Pinto, 2, 1649-003 Lisboa, Portugal

[2] LIAFA, Université Paris 7 and CNRS, Case 7014, 75205 Paris Cedex 13, France

The *polynomial closure* $\mathrm{Pol}(\mathcal{L})$ of a class of languages $\mathcal{L}$ of A^* is the set of languages that are finite unions of marked products of the form $L_0a_1L_1\cdots a_nL_n$, where the a_i are letters and the L_i are elements of $\mathcal{L}$.

The main result of this paper gives an equational description of $\mathrm{Pol}(\mathcal{L})$, given an equational description of $\mathcal{L}$, when $\mathcal{L}$ is a lattice of regular languages closed under quotients, or a *quotienting algebra of languages*, as we call it in the sequel. The term “equational description” refers to a recent paper [5], where it was shown that any lattice of regular languages can be defined by a set of profinite equations. More formally, our main result can be stated as follows:

> *If $\mathcal{L}$ is a quotienting algebra of languages, then $Pol(\mathcal{L})$ is defined by the set of equations of the form $x^\omega y x^\omega \leqslant x^\omega$, where x, y are profinite words such that the equations $x = x^2$ and $y \leqslant x$ are satisfied by $\mathcal{L}$.*

As an application of this result, we establish a set of profinite equations defining the class of languages of the form $L_0a_1L_1\cdots a_nL_n$, where each language L_i is either of the form u^* (where u is a word) or A^* (where A is the alphabet) and we prove that this class is decidable. Let us now give the motivations of our work and a brief survey of the previously known results.

Motivations. The polynomial closure occurs in several difficult problems on regular languages. For instance, a language has *star-height one* if and only if it belongs to the polynomial closure of the set of languages of the form F or F^*, where F is a finite language. Although this class is known to be decidable, it is still an open problem to find profinite equations for this class. Such a result could serve, in turn, to discover a language of generalized star-height > 1, a widely open problem.

The polynomial closure is also one of the two operations appearing in the definition of the *concatenation hierarchy* over a given set $\mathcal{L}$ of regular languages, defined by induction on n as follows. The level 0 is $\mathcal{L}$ and, for each $n \geqslant 0$, the level $2n+1$ is the polynomial closure of the level $2n$ and the level $2n+2$ is the Boolean closure of the level $2n+1$. The simplest hierarchy is built on the initial set $\mathcal{L} = \{\emptyset, A^*\}$. A nice result of Thomas [15] shows that a regular language is of level $2n+1$ in this hierarchy if and only if it is definable by a Σ_{n+1}-sentence of

[⋆] The authors acknowledge support from the AutoMathA programme of the European Science Foundation and the projects ISFL-1-143 and PTDC/MAT/69514/2006 of CAUL, financed by FCT and FEDER.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 115–126, 2009.

first order logic in the signature $\{<, (\mathbf{a})_{a \in A}\}$, where $\mathbf{a}$ is a predicate giving the positions of the letter a. Similar logical interpretations hold for other hierarchies, but unfortunately, only the very low levels of such hierarchies are known to be decidable and in general, this type of decidability problems is considered to be difficult. Our result certainly does not solve the problem in general, but it gives an algebraic approach that can be successful in some particular cases, like the one considered in Section 3, which does not follow from the results of [9].

Known results. A similar result was known when $\mathcal{L}$ is a variety of languages, that is, a class of regular languages closed under Boolean operations, quotients and inverse of morphisms, but depended on the conjunction of two theorems. The first theorem [11] relied on Eilenberg's theory of varieties, which gives a bijective correspondence between varieties of languages and varieties of finite monoids. It stated, in essence, that the polynomial closure corresponds, on the monoid level, to a certain Mal'cev product of varieties. The second result [10] gave identities for the Mal'cev product of two varieties of finite monoids. These results have been extended in [7] to positive varieties of languages and in [9] to quotienting algebras closed under inverse of length-preserving morphisms. However, all these proofs relied on the original proof of [11] and required the use of Mal'cev products and relational morphisms.

In summary, our new result is more general than all the previously known results. Further, our new proof combines various ideas from the above-mentioned papers, but avoids the use of Mal'cev products, a major difference with the original proof, although the experienced reader will still recognize their ghost in this paper. This could be a decisive advantage for potential extensions to other structures, like words over linear orders or finite trees.

1 Definitions and Background

1.1 Languages, Monoids and Syntactic Order

Let A be a finite alphabet. A *lattice of languages* is a set of regular languages of A^* containing the empty language, the full language A^* and closed under finite intersection and finite union. We denote by L^c the complement of a language L of A^*.

Let L be a language of A^* and let u be a word. The *left quotient* of L by u is the language $u^{-1}L = \{v \in A^* \mid uv \in L\}$. The *right quotient* Lu^{-1} is defined in a symmetrical way. A *quotienting algebra of languages* is a lattice of languages closed under the operations $L \mapsto u^{-1}L$ and $L \mapsto Lu^{-1}$, for any word u.

An *ordered monoid* is a monoid M equipped with a partial order $\leqslant$ compatible with the product on M: for all $x, y, z \in M$, if $x \leqslant y$ then $zx \leqslant zy$ and $xz \leqslant yz$. For each $x \in M$, we set $\downarrow x = \{y \in M \mid y \leqslant x\}$. A *morphism of ordered monoids* is an order-preserving monoid morphism.

The *syntactic congruence* of a language L of A^* is the equivalence relation on A^* defined by $u \sim_L v$ if and only if, for every $x, y \in A^*$,

$$xvy \in L \iff xuy \in L$$

The monoid $M = A^*/{\sim_L}$ is the *syntactic monoid* of L and the natural morphism $\eta : A^* \to M$ is called the *syntactic morphism* of L. It is a well-known fact that a language is regular if and only if its syntactic monoid is finite.

The *syntactic preorder* of a language L is the relation $\leqslant_L$ over A^* defined by $u \leqslant_L v$ if and only if, for every $x, y \in A^*$, $xvy \in L$ implies $xuy \in L$. The associated equivalence relation is the syntactic congruence $\sim_L$. Further, $\leqslant_L$ induces a partial order on the syntactic monoid M of L. This partial order $\leqslant$ is compatible with the product and can also be defined directly on M as follows: given $u, v \in M$, one has $u \leqslant v$ if and only if, for all $x, y \in M$, $xvy \in \eta(L)$ implies $xuy \in \eta(L)$. The ordered monoid $(M, \leqslant)$ is called the *syntactic ordered monoid* of L.

1.2 Factorization Forests

We review in this section an important combinatorial result of I. Simon on finite semigroups. A *factorization forest* is a function F that associates with every word x of A^2A^* a factorization $F(x) = (x_1, \ldots, x_n)$ of x such that $n \geqslant 2$ and $x_1, \ldots, x_n \in A^+$. The integer n is the *degree* of the factorization $F(x)$. Given a factorization forest F, the *height function* of F is the function $h : A^* \to \mathbb{N}$ defined recursively by

$$h(x) = \begin{cases} 0 & \text{if } |x| \leqslant 1 \\ 1 + \max\{h(x_i) \mid 1 \leqslant i \leqslant n\} & \text{if } F(x) = (x_1, \ldots, x_n) \end{cases}$$

The *height* of F is the least upper bound of the heights of the words of A^*.

Let M be a finite monoid and let $\varphi : A^* \to M$ be a morphism. A factorization forest F is *Ramseyan modulo* φ if, for every word x of A^2A^*, $F(x)$ is either of degree 2 or there exists an idempotent e of M such that $F(x) = (x_1, \ldots, x_n)$ and $\varphi(x_1) = \varphi(x_2) = \cdots = \varphi(x_n) = e$ for $1 \leqslant i \leqslant n$. The factorization forest theorem was first proved by I. Simon in [12,13,14] and later improved in [2,3,4,6]:

Theorem 1.1. *Let φ be a morphism from A^* into a finite monoid M. There exists a factorization forest of height $\leqslant 3|M| - 1$ which is Ramseyan modulo φ.*

1.3 Profinite Monoids and Equations

We briefly recall the definition of a free profinite monoid. More details can be found in [1,8]. A finite monoid M *separates* two words u and v of A^* if there is a morphism $\varphi : A^* \to M$ such that $\varphi(u) \neq \varphi(v)$. We set

$$r(u, v) = \min\bigl\{\mathrm{Card}(M) \mid \ M \text{ is a finite monoid that separates } u \text{ and } v\ \bigr\}$$

and $d(u, v) = 2^{-r(u,v)}$, with the usual conventions $\min \emptyset = +\infty$ and $2^{-\infty} = 0$. Then d is a *metric* on A^* and the completion of A^* for this metric is denoted by $\widehat{A^*}$. The product on A^* can be extended by continuity to $\widehat{A^*}$. This extended product makes $\widehat{A^*}$ a compact topological monoid, called the *free profinite monoid.* Its elements are called *profinite words.*

Every finite monoid M can be considered as a discrete metric space for the discrete metric d, defined by $d(x,y) = 0$ if $x = y$, and $d(x,y) = 1$ otherwise. Now, every morphism φ from A^* into a finite monoid is uniformly continuous and therefore can be extended (in a unique way) into a uniformly continuous morphism $\hat{\varphi}$ from $\widehat{A^*}$ to M.

Since A^* embeds naturally in $\widehat{A^*}$, every finite word is a profinite word. We shall also use the operator $x \mapsto x^\omega$ in $\widehat{A^*}$, which is formally defined by the formula $x^\omega = \lim_{n\to\infty} x^{n!}$ and is justified by the fact that the sequence $(x^{n!})_{n\geqslant 0}$ is a Cauchy sequence in $\widehat{A^*}$ and hence has a limit in $\widehat{A^*}$. Let φ be a morphism from A^* onto a finite monoid M and let $s = \hat{\varphi}(x)$. Then the sequence $(s^{n!})_{n\geqslant 0}$ is ultimately equal to s^ω, where ω is the least integer k such that for all $t \in M$, t^k is idempotent. Consequently, we obtain the formula $\hat{\varphi}(x^\omega) = \hat{\varphi}(x)^\omega$, which gives ground to the notation x^ω.

Let L be a regular language of A^*, let $(M, \leqslant)$ be its syntactic ordered monoid and let $\eta : A^* \to M$ its syntactic morphism. Given two profinite words $u, v \in \widehat{A^*}$, we say that L *satisfies the (profinite) equation* $u \leqslant v$ (resp. $u = v$) if $\hat{\eta}(u) \leqslant \hat{\eta}(v)$ (resp. $\hat{\eta}(u) = \hat{\eta}(v)$). By extension, we say that a set of languages $\mathcal{L}$ *satisfies a set of equations* Σ if every language of $\mathcal{L}$ satisfies every equation of Σ.

2 Polynomial Closure of Lattices of Languages

Let $\mathcal{L}$ be a set of languages of A^*. An $\mathcal{L}$-*monomial of degree* n is a language of the form $L_0a_1L_1 \cdots a_nL_n$, where each a_i is a letter of A and each L_i is a language of $\mathcal{L}$. An $\mathcal{L}$-*polynomial* is a finite union of $\mathcal{L}$-monomials. Its *degree* is the maximum of the degrees of its monomials. The *polynomial closure* of $\mathcal{L}$, denoted by $\mathrm{Pol}(\mathcal{L})$, is the set of all $\mathcal{L}$-polynomials.

Our main result gives an equational description of $\mathrm{Pol}(\mathcal{L})$, given an equational description of $\mathcal{L}$, when $\mathcal{L}$ is a quotienting algebra of languages. To state this result in a concise way, let us introduce a convenient notation. Given a set $\mathcal{R}$ of regular languages, denote by $\Sigma(\mathcal{R})$ the set of equations of the form $x^\omega y x^\omega \leqslant x^\omega$, where x, y are profinite words of $\widehat{A^*}$ such that the equations $x = x^2$ and $y \leqslant x$ are satisfied by $\mathcal{R}$. Note that the function mapping $\mathcal{R}$ to the class of languages satisfying $\Sigma(\mathcal{R})$ is monotonic for the inclusion. We can now state our main result:

Theorem 2.1. *If $\mathcal{L}$ is a quotienting algebra of languages, then Pol$(\mathcal{L})$ is defined by the set of equations $\Sigma(\mathcal{L})$.*

The proof is divided into several parts. We first prove in Proposition 2.2 that $\mathrm{Pol}(\mathcal{L})$ satisfies the equations of $\Sigma(\mathcal{L})$. To establish the converse of this property, we consider a language K satisfying all the equations of $\Sigma(\mathcal{L})$. We convert this property into a topological property (Proposition 2.4) and then use a compactness argument to show that K satisfies the equations of $\Sigma(\mathcal{F})$, where $\mathcal{F}$ is a finite sublattice of $\mathcal{L}$ (Proposition 2.5). The final part of the proof consists in proving that K belongs to $\mathrm{Pol}(\mathcal{F})$. This is where the factorization forest theorem arises, but a series of lemmas (Lemmas 2.6 to 2.11) are still necessary to find explicitly a polynomial expression for K.

Proposition 2.2. *If $\mathcal{L}$ is a lattice of languages, then $Pol(\mathcal{L})$ satisfies the equations of $\Sigma(\mathcal{L})$.*

Proof. Since, by [5, Theorem 7.2] the set of languages satisfying $\Sigma(\mathcal{L})$ is a lattice of languages, it suffices to prove the result for any $\mathcal{L}$-monomial. Let $L = L_0 a_1 L_1 \cdots a_n L_n$ be an $\mathcal{L}$-monomial and let $\eta\colon A^* \to M$ be its syntactic morphism. Let, for $0 \leqslant i \leqslant n$, $\eta_i\colon A^* \to M_i$ be the syntactic morphism of L_i. Let x and y be two profinite words such that each L_i satisfies the two equations $x = x^2$ and $y \leqslant x$.

Since A^* is dense in $\widehat{A^*}$, one can find a word $x' \in A^*$ such that $r(x', x) > \max\{|M_0|, \ldots, |M_n|, |M|\}$. It follows that $\eta(x') = \hat{\eta}(x)$ and, for $0 \leqslant i \leqslant n$, $\eta_i(x') = \hat{\eta}_i(x)$. Similarly, one can associate with y a word $y' \in A^*$ such that $\eta(y') = \hat{\eta}(y)$ and, for $0 \leqslant i \leqslant n$, $\eta_i(y') = \hat{\eta}_i(y)$. It follows that each L_i satisfies the equations $x' = {x'}^2$ and $y' \leqslant x'$ and that L satisfies the equation $x^\omega y x^\omega \leqslant x^\omega$ if and only if it satisfies the equations ${x'}^\omega y' {x'}^\omega \leqslant {x'}^\omega$. In other terms, it suffices to prove the result when x and y are words.

We need to establish the relation $(*) : \hat{\eta}(x^\omega y x^\omega) \leqslant \hat{\eta}(x^\omega)$. Let k be an integer such that $k > n$ and $\hat{\eta}(x^\omega) = \eta(x^k)$. Since $\hat{\eta}(x^\omega y x^\omega) = \eta(x^k y x^k)$, proving $(*)$ amounts to showing that $x^k y x^k \leqslant_L x^k$. Let $u, v \in A^*$ and suppose that $u x^k v \in L$. Thus $u x^k v = u_0 a_1 u_1 \cdots a_n u_n$, where, for $0 \leqslant i \leqslant n$, $u_i \in L_i$. Since $k > n$, one can find $h \in \{0, \ldots, n\}$, $j \in \{1, \ldots, k\}$ and $u'_h, u''_h \in A^*$ such that $u_h = u'_h x u''_h$, $u x^{j-1} = u_0 a_1 u_1 \cdots a_h u'_h$ and $x^{k-j} v = u''_h a_{h+1} u_{h+1} \cdots a_n u_n$. Since $u_h \in L_h$ and L_h satisfies the equations $x = x^2$ and $y \leqslant x$, one has $u'_h x^{k-j+1} y x^j u''_h \in L_h$, and since $u x^k y x^k v = u_0 a_1 u_1 \cdots a_h (u'_h x^{k-j+1} y x^j u''_h) a_{h+1} u_{h+1} \cdots a_n u_n$, one gets $u x^k y x^k v \in L$. Thus $x^k y x^k \leqslant_L x^k$, which completes the proof. □

The rest of this section is devoted to showing the converse implication in Theorem 2.1. Let us introduce, for each regular language L of A^*, the sets

$$E_L = \Big\{(x,y) \in \widehat{A^*} \times \widehat{A^*} \mid L \text{ satisfies } x = x^2 \text{ and } y \leqslant x\Big\}$$
$$F_L = \Big\{(x,y) \in \widehat{A^*} \times \widehat{A^*} \mid L \text{ satisfies } x^\omega y x^\omega \leqslant x^\omega\Big\}.$$

Recall that a subset of a topological space is *clopen* if it is both open and closed.

Lemma 2.3. *For each regular language L of A^*, the sets E_L and F_L are clopen in $\widehat{A^*} \times \widehat{A^*}$.*

Proof. Let $\eta\colon A^* \to M$ be the syntactic morphism of L. The formula $\alpha(x,y) = \big(\hat{\eta}(x), \hat{\eta}(x^2), \hat{\eta}(y)\big)$ defines a continuous map α from $\widehat{A^*} \times \widehat{A^*}$ into M^3, considered as a discrete space. Setting $\varDelta = \{(s,t,u) \in M^3 \mid s = t \text{ and } u \leqslant s\}$, we get

$$E_L = \Big\{(x,y) \in \widehat{A^*} \times \widehat{A^*} \mid \hat{\eta}(x) = \hat{\eta}(x^2) \text{ and } \hat{\eta}(y) \leqslant \hat{\eta}(x)\Big\} = \alpha^{-1}(\varDelta)$$

Now, since M is a discrete topological space, $\varDelta$ is clopen in M^3 and thus E_L is a clopen subset of $\widehat{A^*} \times \widehat{A^*}$.

A similar argument, using the continuous map $\beta : \widehat{A^*} \times \widehat{A^*} \to M^2$ defined by $\beta(x,y) = \big(\hat{\eta}(x^\omega y x^\omega), \hat{\eta}(x^\omega)\big)$, would show that F_L is clopen. □

We now convert our equational conditions into a topological property. Recall that a *cover* [*open cover*] of a topological space X is a collection of subsets [open subsets] of X whose union is X.

Proposition 2.4. *Let $\mathcal{F}$ be a set of regular languages of A^* and let K be a regular language of A^*. The following conditions are equivalent:*

(1) *K satisfies the profinite equations of $\Sigma(\mathcal{F})$,*

(2) *the set $\{F_K\} \cup \{E_L^c \mid L \in \mathcal{F}\}$ is an open cover of $\widehat{A^*} \times \widehat{A^*}$.*

Proof. Indeed $\mathcal{F}$ satisfies the two profinite equations $x = x^2$ and $y \leqslant x$ if and only if $(x, y) \in \bigcap_{L\in\mathcal{F}} E_L$ or, equivalently, $(x, y) \notin \bigcup_{L\in\mathcal{F}} E_L^c$. Similarly, K satisfies the equation $x^\omega y x^\omega \leqslant x^\omega$ if and only if $(x, y) \in F_K$. Now, condition (1) is equivalent to saying that $(x, y) \notin \bigcup_{L\in\mathcal{F}} E_L^c$ implies $(x, y) \in F_K$, which is another way to say that $\{F_K\} \cup \{E_L^c \mid L \in \mathcal{F}\}$ is a cover of $\widehat{A^*} \times \widehat{A^*}$. Further, Proposition 2.3 shows that it is an open cover. □

Proposition 2.5. *If K satisfies the equations of $\Sigma(\mathcal{L})$, there is a finite subset $\mathcal{F}$ of $\mathcal{L}$ such that K satisfies the equations of $\Sigma(\mathcal{F})$.*

Proof. Proposition 2.4 shows that $\{F_K\} \cup \{E_L^c \mid L \in \mathcal{L}\}$ is a cover of $\widehat{A^*} \times \widehat{A^*}$. Since $\widehat{A^*}$ is compact, one can extract from this cover a finite cover, say $\{F_K\} \cup \{E_L^c \mid L \in \mathcal{F}\}$. By Proposition 2.4 again, K satisfies the profinite equations of the form $x^\omega y x^\omega \leqslant x^\omega$ such that all the languages of $\mathcal{F}$ satisfy the equations $x = x^2$ and $y \leqslant x$. □

Let K be a regular language satisfying all the equations of $\Sigma(\mathcal{L})$ and let $\eta : A^* \to M$ be its syntactic morphism. Let also $\mathcal{F} = \{L_1, \ldots, L_n\}$ be a finite subset of $\mathcal{L}$ as given by Proposition 2.5. For $1 \leqslant i \leqslant n$, let $\eta_i \colon A^* \to M_i$ be the syntactic morphism of L_i. Let $\mu \colon A^* \to M_1 \times \cdots \times M_n$ be the morphism defined by $\mu(u) = (\eta_1(u), \ldots, \eta_n(u))$. Finally, let $V = \mu(A^*)$ and, for $1 \leqslant i \leqslant n$, let $\pi_i : V \to M_i$ be the natural projection. We set $S = \{(\eta(u), \mu(u)) \mid u \in A^*\}$. Then S is a submonoid of $M \times V$ and the two morphisms $\alpha : S \to M$ and $\beta : S \to V$ defined by $\alpha(m, v) = m$ and $\beta(m, v) = v$ are surjective. Further, the morphism $\delta : A^* \to S$ defined by $\delta(u) = (\eta(u), \mu(u))$ satisfies $\eta = \alpha \circ \delta$ and $\mu = \beta \circ \delta$. The situation is summarized in the following diagram:

$$\begin{tikzcd}
 & A^* \arrow[dl, "\eta"'] \arrow[d, "\delta"] \arrow[dr, "\mu"] \arrow[drr, "\eta_i"] & & \\
M & S \arrow[l, "\alpha"'] \arrow[r, "\beta"] & V \arrow[r, "\pi_i"] & M_i
\end{tikzcd}$$

We now arrive at the last step of the proof of Theorem 2.1, which consists in proving that K belongs to $\mathrm{Pol}(\mathcal{F})$.

We start with three auxiliary lemmas. The first one states that every downward closed language recognized by μ belongs to $\mathcal{L}$ and relies on the fact that $\mathcal{L}$ is a quotienting algebra of languages. The second one gives a key property of S and this is the only place in the proof where we use the equations of $\Sigma(\mathcal{L})$. The third one is an elementary, but useful, observation.

Lemma 2.6. *Let $t \in V$. Then the language $\mu^{-1}(\downarrow t)$ belongs to $\mathcal{L}$.*

Proof. Let $t = (t_1, \ldots, t_n)$ and let z be a word such that $\mu(z) = t$. Then $t_i = \eta_i(z)$ and $\mu^{-1}(\downarrow t) = \bigcap_{1 \leqslant i \leqslant n} \eta_i^{-1}(\downarrow t_i)$. Moreover, one gets for each $i \in \{1, \ldots, n\}$,

$$\eta_i^{-1}(\downarrow t_i) = \{x \in A^* \mid \eta_i(x) \leqslant \eta_i(z)\} = \{x \in A^* \mid x \leqslant_{L_i} z\} = \bigcap_{(u,v) \in E_i} u^{-1} L_i v^{-1}$$

where $E_i = \{(u, v) \in A^* \times A^* \mid uzv \in L_i\}$. Since L_i is regular, there are only finitely many quotients of the form $u^{-1}L_iv^{-1}$ and hence the intersection is finite. The result follows, since $\mathcal{L}$ is a quotienting algebra of languages. □

Lemma 2.7. *For every idempotent $(e, f) \in S$ and for every $(s, t) \in S$ such that $t \leqslant f$, one has $ese \leqslant e$.*

Proof. Let x and y be two words such that $\delta(x) = (e, f)$ and $\delta(y) = (s, t)$. Then $\eta(x) = e$, $\mu(x) = f$, $\eta(y) = s$ and $\mu(y) = t$ and since f is idempotent and $t \leqslant f$, $\mathcal{F}$ satisfies the equations $x = x^2$ and $y \leqslant x$. Therefore K satisfies the equation $x^\omega y x^\omega \leqslant x^\omega$. It follows that $\hat{\eta}(x^\omega y x^\omega) \leqslant \hat{\eta}(x^\omega)$, that is $ese \leqslant e$. □

Before we continue, let us point out a subtlety in the proof of Lemma 2.7. It looks like we have used words instead of profinite words in this proof and the reader may wonder whether one could change "profinite" to "finite" in the statement of our main result. The answer is negative for the following reason: if $\mathcal{F}$ satisfies the equations $x = x^2$ and $y \leqslant x$, it does not necessarily imply that $\mathcal{L}$ satisfies the same equations. In fact, the choice of $\mathcal{F}$ comes from the extraction of the finite cover and hence is bound to K.

We now set, for each idempotent f of V, $L(f) = \mu^{-1}(\downarrow f)$.

Lemma 2.8. *For each idempotent f of V, one has $L(1)L(f)L(1) = L(f)$.*

Proof. Since $1 \in L(1)$, one gets the inclusion $L(f) = 1L(f)1 \subseteq L(1)L(f)L(1)$. Let now $s, t \in L(1)$ and $x \in L(f)$. Then by definition, $\mu(s) \leqslant 1$, $\mu(x) \leqslant f$ and $\mu(t) \leqslant 1$. It follows that $\mu(sxt) = \mu(s)\mu(x)\mu(t) \leqslant 1f1 = f$, whence $sxt \in L(f)$. This gives the opposite inclusion $L(1)L(f)L(1) \subseteq L(f)$. □

We now come to the combinatorial argument of the proof. By Theorem 1.1, there exists a factorization forest F of height $\leqslant 3|S| - 1$ which is Ramseyan modulo δ. We use this fact to associate with each word x a certain language $R(x)$, defined recursively as follows:

$$R(x) = \begin{cases} L(1)xL(1) & \text{if } |x| \leqslant 1 \\ R(x_1)R(x_2) & \text{if } F(x) = (x_1, x_2) \\ R(x_1)L(f)R(x_k) & \text{if } F(x) = (x_1, \ldots, x_k), \text{ with } k \geqslant 3 \text{ and} \\ & \qquad \delta(x_1) = \cdots = \delta(x_k) = (e, f) \end{cases}$$

In particular $R(1) = L(1)$, since $L(1)$ is a submonoid of A^*.

Denote by $\mathcal{E}$ the finite set of languages of the form $L(f)$, where f is an idempotent of V. We know by Lemma 2.6 that $\mathcal{E}$ is a subset of $\mathcal{L}$. Let us say that an $\mathcal{E}$-monomial is in *normal form* if it is of the form $L(1)a_0L(f_1)a_1 \cdots L(f_k)a_kL(1)$ where $f_1, \ldots, f_k$ are idempotents of V.

Lemma 2.9. *For each $x \in A^*$, $R(x)$ is equal to an $\mathcal{E}$-monomial in normal form of degree $\leqslant 2^{h(x)}$.*

Proof. We prove the result by induction on the length of x. The result is true if $|x| \leqslant 1$. Suppose that $|x| \geqslant 2$. If $F(x) = (x_1, x_2)$, then $R(x) = R(x_1)R(x_2)$ otherwise $R(x) = R(x_1)L(f)R(x_k)$. We treat only the latter case, since the first one is similar. By the induction hypothesis, $R(x_1)$ and $R(x_k)$ are equal to $\mathcal{E}$-monomials in normal form. It follows by Lemma 2.8 that $R(x)$ is equal to an $\mathcal{E}$-monomial in normal form, whose degree is lesser than or equal to the sum of the degrees of $R(x_1)$ and $R(x_k)$. The result now follows from the induction hypothesis, since $2^{h(x_1)} + 2^{h(x_k)} \leqslant 2^{1+\max\{h(x_1),\ldots,h(x_k)\}} \leqslant 2^{h(x)}$. □

Lemma 2.10. *For each $x \in A^*$, one has $x \in R(x)$.*

Proof. We prove the result by induction on the length of x. The result is trivial if $|x| \leqslant 1$. Suppose that $|x| \geqslant 2$. If $F(x) = (x_1, x_2)$, one has $x_1 \in R(x_1)$ and $x_2 \in R(x_2)$ by the induction hypothesis and hence $x \in R(x)$ since $R(x) = R(x_1)R(x_2)$. Suppose now that $F(x) = (x_1, \ldots, x_k)$ with $k \geqslant 3$ and $\delta(x_1) = \cdots = \delta(x_k) = (e, f)$. Then $R(x) = R(x_1)L(f)R(x_k)$. Since $x_1 \in R(x_1)$ and $x_k \in R(x_k)$ by the induction hypothesis and $\mu(x_2 \cdots x_{k-1}) = f$, one gets $x_2 \cdots x_{k-1} \in L(f)$ and finally $x \in R(x_1)L(f)R(x_k)$, that is, $x \in R(x)$. □

If R is a language, let us write $\eta(R) \leqslant \eta(x)$ if, for each $u \in R$, $\eta(u) \leqslant \eta(x)$.

Lemma 2.11. *For each $x \in A^*$, one has $\eta(R(x)) \leqslant \eta(x)$.*

Proof. We prove the result by induction on the length of x. First, applying Lemma 2.7 with $e = f = 1$ shows that if $(s, t) \in S$ and $t \leqslant 1$, then $s \leqslant 1$. It follows that $\eta(R(1)) = \eta(L(1)) = \eta(\mu^{-1}(\downarrow 1)) \leqslant 1$.

If $|x| \leqslant 1$, one gets $R(x) = L(1)xL(1)$ and $\eta(R(x)) = \eta(L(1))\eta(x)\eta(L(1)) \leqslant \eta(x)$ since $\eta(L(1)) \leqslant 1$. Suppose now that $|x| \geqslant 2$. If $F(x) = (x_1, x_2)$, then $R(x) = R(x_1)R(x_2)$ and by the induction hypothesis, $\eta(R(x_1)) \leqslant \eta(x_1)$ and $\eta(R(x_2)) \leqslant \eta(x_2)$. Therefore, $\eta(R(x)) = \eta(R(x_1))\eta(R(x_2)) \leqslant \eta(x_1)\eta(x_2) = \eta(x)$. Finally, suppose that $F(x) = (x_1, \ldots, x_k)$ with $k \geqslant 3$ and $\delta(x_1) = \cdots = \delta(x_k) = (e, f)$. Then $R(x) = R(x_1)L(f)R(x_k)$. By the induction hypothesis, $\eta(R(x_1)) \leqslant e$ and $\eta(R(x_k)) \leqslant e$. Now, if $u \in L(f)$, one gets $\mu(u) \leqslant f$. Since $(\eta(u), \mu(u)) \in S$, it follows from Lemma 2.7 that the relation $e\eta(u)e \leqslant e$ holds in M. Finally, we get $\eta(R(x)) = \eta(R(x_1))\eta(L(f))\eta(R(x_k)) \leqslant e\eta(L(f))e \leqslant e = \eta(x)$. □

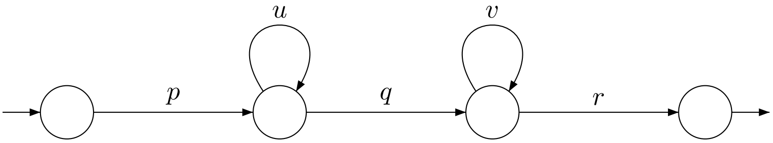
We can now conclude the proof of Theorem 2.1. We claim that $K = \bigcup_{x \in K} R(x)$. The inclusion $K \subseteq \bigcup_{x \in K} R(x)$ is an immediate consequence of Lemma 2.10. To prove the opposite inclusion, consider a word $u \in R(x)$ for some $x \in K$. It follows from Lemma 2.11 that $\eta(u) \leqslant \eta(x)$. Since $\eta(x) \in \eta(K)$, one gets $\eta(u) \in \eta(K)$ and finally $u \in K$. Now, by Lemma 2.9, each language $R(x)$ is an $\mathcal{E}$-monomial of degree $\leqslant 2^{h(x)}$. Since $h(x) \leqslant 3|S| - 1$ for all x, and since $\mathcal{E}$ is finite, there are only finitely many such monomials. Therefore K is equal to an $\mathcal{E}$-polynomial. Finally, Lemma 2.6 shows that each $\mathcal{E}$-polynomial belongs to $\mathrm{Pol}(\mathcal{L})$, and thus $K \in \mathrm{Pol}(\mathcal{L})$. □

3 A Case Study

The *density* of a language $L \subseteq A^*$ is the function which counts the number of words of length n in L. More formally, it is the function $d_L : \mathbb{N} \to \mathbb{N}$ defined by $d_L(n) = |L \cap A^n|$. See [16] for a general reference. If $d_L(n) = O(1)$, then L is called a *slender language*. A regular language of A^* is *slender* if and only if it is a finite union of languages of the form $u_0 v^* u_1$, where $u_0, v, u_1 \in A^*$ (see [16, Theorem 3.6]). A language is *sparse* if it is of polynomial density. One can show that a regular language is sparse if and only if it is a finite union of languages of the form $u_0 v_1^* u_1 \cdots v_n^* u_n$, where u_0, v_1, ..., v_n, u_n are words.

We shall also use the following characterization of regular nonslender languages, in which $i(u)$ denotes the first letter (or *initial*) of a word u.

Proposition 3.1. *A regular language L is nonslender if and only if there exist words $p, q, r \in A^*$ and $u, v \in A^+$ such that $i(u) \neq i(qv)$ and $pu^*qv^*r \subseteq L$.*



If $|A| \leqslant 1$, every regular language is slender, but if $|A| \geqslant 2$, the full language A^* is not slender and thus regular slender languages do not form a lattice of languages. However, the regular languages that are either slender or full form a quotienting algebra of languages, denoted by $\mathcal{S}$ in the sequel. Two sets of profinite equations for $\mathcal{S}$ were given in [5]. We shall just mention the second one, which requires a convenient writing convention. Let L be a regular language of A^* and let $\eta : A^* \to M$ be its syntactic morphism. If x is a profinite word of $\widehat{A^*}$, we say that L satisfies the equation $x \leqslant 0$ $[x = 0]$, if the monoid M has a zero, denoted by 0, and if $\hat{\eta}(x) \leqslant 0$ $[\hat{\eta}(x) = 0]$.

Proposition 3.2. *Suppose that $|A| \geqslant 2$. A regular language of A^* is slender or full if and only if it satisfies the equations $x \leqslant 0$ for all $x \in A^*$ and the equations $x^\omega u y^\omega = 0$ for each $x, y \in A^+$, $u \in A^*$ such that $i(x) \neq i(uy)$.*

We are interested in the polynomial closure of $\mathcal{S}$. The languages of $\mathrm{Pol}(\mathcal{S})$ are finite unions of languages of the form $L_0 a_1 L_1 \cdots a_n L_n$, where the a_i are letters and the L_i are languages of the form A^* or u^* for some word u.[1] In particular,

[1] To see this, it suffices to replace each word $u_i = a_1 \cdots a_k$ by $1^* a_1 1^* a_2 1^* \cdots 1^* a_k 1^*$ in each monomial of the form $u_0 v_1^* u_1 \cdots v_n^* u_n$.

Pol($\mathcal{S}$) contains all regular sparse languages but it also contains the nonsparse language A^* if $|A| \geqslant 2$.

The main result of this section is an equational description of Pol($\mathcal{S}$). Let us denote by $\Sigma'(\mathcal{S})$ the set of equations of the form

$$(x^\omega y^\omega)^\omega z (x^\omega y^\omega)^\omega \leqslant (x^\omega y^\omega)^\omega$$

where $z \in A^*$ and $x, y \in A^+$ and $i(x) \neq i(y)$.

Theorem 3.3. *A regular language of A^* belongs to Pol($\mathcal{S}$) if and only if it satisfies the equations of $\Sigma'(\mathcal{S})$.*

Proof. Let us first settle a trivial case. If $|A| \leqslant 1$, every regular language belongs to Pol($\mathcal{S}$), but on the other hand, the set $\Sigma'(\mathcal{S})$ is empty because the condition $i(x) \neq i(y)$ is never satisfied! We suppose now that $|A| \geqslant 2$.

We show that every language of Pol($\mathcal{S}$) satisfies the equations of $\Sigma'(\mathcal{S})$ by applying Theorem 2.1. It suffices to verify that, if $i(x) \neq i(y)$, $\mathcal{S}$ satisfies the equations $(x^\omega y^\omega)^\omega = \big((x^\omega y^\omega)^\omega\big)^2$ and $z \leqslant (x^\omega y^\omega)^\omega$. But this is trivial, since we know by Proposition 3.2 that $\mathcal{S}$ satisfies the equations $x^\omega y^\omega = 0$ (take $u = 1$ in the equation $x^\omega u y^\omega = 0$) and $z \leqslant 0$.

Let K be a regular language satisfying the equations of $\Sigma'(\mathcal{S})$ and let $\eta : A^* \to M$ be its syntactic morphism. We immediately derive from $\Sigma'(\mathcal{S})$ a more comprehensible property, which is the counterpart of Lemma 2.7 in the proof of Theorem 2.1.

Lemma 3.4. *Let e be an idempotent of M. Then either $\eta^{-1}(e)$ is slender, or for all $s \in M$, $ese \leqslant e$.*

Proof. Let $L = \eta^{-1}(e)$. Since L is not slender, Proposition 3.1 tells us that one can find words $p, q, r \in A^*$ and $u, v \in A^+$ such that $i(u) \neq i(qv)$ and $pu^*qv^*r \subseteq L$. Further, since e is idempotent, L is a semigroup and we have in fact $(pu^*qv^*r)^+ \subseteq L$. It follows that

$$p(u^*(qvrp)^*)^*qr \subseteq p(u^*(qv^*rp)^*)^*u^*qv^*r \subseteq (pu^*qv^*r)^+ \subseteq L$$

Setting $x = u$, $y = qvrp$ and $t = qr$, we get $i(x) \neq i(y)$ and $p(x^*y^*)^*t \subseteq L$. It follows in particular that

$$\hat{\eta}\big(p(x^\omega y^\omega)^\omega t\big) = e \tag{1}$$

Let $s \in M$ and let w be a word such that $\eta(w) = s$. Since L satisfies the equations of $\Sigma'(\mathcal{S})$, it satisfies in particular the equation $(x^\omega y^\omega)^\omega twp(x^\omega y^\omega)^\omega \leqslant (x^\omega y^\omega)^\omega$ and hence also $p(x^\omega y^\omega)^\omega twp(x^\omega y^\omega)^\omega t \leqslant p(x^\omega y^\omega)^\omega t$. This means that

$$\hat{\eta}\big(p(x^\omega y^\omega)^\omega twp(x^\omega y^\omega)^\omega t\big) \leqslant \hat{\eta}\big(p(x^\omega y^\omega)^\omega t\big) \tag{2}$$

Now, using (1), (2) and the relation $\eta(w) = s$, we get $ese \leqslant e$. □

The end of the proof is similar to that of Theorem 2.1, with the major difference that we do not use the morphisms δ and μ anymore. By Theorem 1.1, there exists a factorization forest F of height $\leqslant 3|M| - 1$ which is Ramseyan modulo η. We associate with each idempotent $e \in M$ the language $L(e)$ equal to $\eta^{-1}(e)$ if this language is slender, and to A^* otherwise. Let us denote by $\mathcal{E}$ the set of languages of the form $L(e)$. By definition, every language of $\mathcal{E}$ is slender or full. We also associate with each word x a language $R(x)$, defined as follows:

$$R(x) = \begin{cases} L(1)xL(1) & \text{if } |x| \leqslant 1 \\ R(x_1)R(x_2) & \text{if } F(x) = (x_1, x_2) \\ R(x_1)L(e)R(x_k) & \text{if } F(x) = (x_1, \ldots, x_k), \text{ with } k \geqslant 3 \text{ and} \\ & \quad \eta(x_1) = \cdots = \eta(x_k) = e \end{cases}$$

The proof now consists in adapting Lemmas 2.8, 2.9, 2.10 and 2.11 to our new definitions. We just give here a sketch of these proofs (detailed proofs can be found in the Appendix). For Lemma 2.8, one needs to prove that, for each idempotent $e \in M$, $L(1)L(e)L(1) = L(e)$. The key observation is that if $\eta^{-1}(e)$ is slender, then $\eta^{-1}(1)$ is slender: indeed $\eta^{-1}(1)\eta^{-1}(e) \subseteq \eta^{-1}(e)$ and if the density of $\eta^{-1}(1)$ is not bounded, the density of $\eta^{-1}(e)$ cannot be bounded. Therefore, if $\eta^{-1}(e)$ is slender, one can follow the original proof. If $\eta^{-1}(e)$ is not slender, then $L(e) = A^*$ and the result is trivial, since $1 \in L(1)$.

The proofs of Lemmas 2.9 and 2.10 are unchanged. The proof of Lemma 2.11 requires a slight modification in the case where $F(x) = (x_1, \ldots, x_k)$ with $k \geqslant 3$, $\eta(x_1) = \cdots = \eta(x_k) = e$ and $\eta^{-1}(e)$ nonslender. Then $R(x) = R(x_1)A^*R(x_k)$ and by the induction hypothesis $\eta(R(x_1)) \leqslant e$ and $\eta(R(x_k)) \leqslant e$. Further, Lemma 3.4 shows that, for all $s \in M$, $ese \leqslant e$. Therefore, for each $s_1 \in \eta(R(x_1))$, $s_k \in \eta(R(x_k))$ and $s \in M$, one gets $s_1 s s_k \leqslant ese \leqslant e$. It follows that $\eta(R(x)) \leqslant e$, which completes the proof, since $\eta(x) = e$.

The rest of the proof is unchanged and shows that K is equal to an $\mathcal{E}$-polynomial. Since each $\mathcal{E}$-monomial is itself in $\mathrm{Pol}(\mathcal{S})$, it follows that $K \in \mathrm{Pol}(\mathcal{S})$. □

Corollary 3.5. *There is an algorithm to decide whether a given regular language belongs to Pol($\mathcal{S}$).*

Proof. Let L be a regular language and let $\eta : A^* \to M$ be its syntactic morphism. By Theorem 3.3, L belongs to $\mathrm{Pol}(\mathcal{S})$ if and only if it satisfies the equations of $\Sigma'(\mathcal{S})$. Setting

$$F = \bigcup_{\substack{a,b \in A \\ a \neq b}} \eta(a)M \times \eta(b)M$$

it suffices to verify that the property $(x^\omega y^\omega)^\omega z (x^\omega y^\omega)^\omega \leqslant (x^\omega y^\omega)^\omega$ holds for all $(x, y) \in F$ and all $z \in M$. Since M and F are finite, this property is decidable. □

References

1. Almeida, J.: Finite semigroups and universal algebra. World Scientific Publishing Co. Inc., River Edge (1994); Translated from the 1992 Portuguese original and revised by the author
2. Chalopin, J., Leung, H.: On factorization forests of finite height. Theoret. Comput. Sci. 310(1-3), 489–499 (2004)
3. Colcombet, T.: A combinatorial theorem for trees. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 901–912. Springer, Heidelberg (2007)
4. Colcombet, T.: Factorisation Forests for Infinite Words. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) FCT 2007. LNCS, vol. 4639, pp. 226–237. Springer, Heidelberg (2007)
5. Gehrke, M., Grigorieff, S., Pin, J.-É.: Duality and equational theory of regular languages. In: Aceto, L., et al. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 246–257. Springer, Heidelberg (2008)
6. Kufleitner, M.: The Height of Factorization Forests. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 443–454. Springer, Heidelberg (2008)
7. Pin, J.-E.: Algebraic tools for the concatenation product. Theoret. Comput. Sci. 292, 317–342 (2003)
8. Pin, J.-E.: Profinite methods in automata theory. In: Albers, S. (ed.) 26th International Symposium on Theoretical Aspects of Computer Science (STACS 2009), Schloss Dagstuhl, Germany, Dagstuhl, Germany. Internationales Begegnungs- Und Forschungszentrum für Informatik (IBFI), pp. 31–50 (2009)
9. Pin, J.-E., Straubing, H.: Some results on $\mathcal{C}$-varieties. Theoret. Informatics Appl. 39, 239–262 (2005)
10. Pin, J.-É., Weil, P.: Profinite semigroups, Mal'cev products and identities. J. of Algebra 182, 604–626 (1996)
11. Pin, J.-É., Weil, P.: Polynomial closure and unambiguous product. Theory Comput. Systems 30, 383–422 (1997)
12. Simon, I.: Properties of factorization forests. In: Pin, J.E. (ed.) LITP 1988. LNCS, vol. 386, pp. 65–72. Springer, Heidelberg (1989)
13. Simon, I.: Factorization forests of finite height. Theoret. Comput. Sci. 72(1), 65–94 (1990)
14. Simon, I.: A short proof of the factorization forest theorem. In: Tree automata and languages (Le Touquet, 1990). Stud. Comput. Sci. Artificial Intelligence, vol. 10, pp. 433–438. North-Holland, Amsterdam (1992)
15. Thomas, W.: Classifying regular events in symbolic logic. J. Comput. System Sci. 25(3), 360–376 (1982)
16. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of formal languages, ch. 2, vol. 1, pp. 45–110. Springer, Heidelberg (1997)

Approximating Markov Processes by Averaging

Philippe Chaput[1,⋆], Vincent Danos[2], Prakash Panangaden[1,⋆], and Gordon Plotkin[2,⋆⋆]

[1] School of Computer Science, McGill University
[2] School of Informatics, University of Edinburgh

Abstract. We take a dual view of Markov processes – advocated by Kozen – as transformers of bounded measurable functions. We redevelop the theory of labelled Markov processes from this view point, in particular we explore approximation theory. We obtain three main results:
(i) It is possible to define bisimulation on general measure spaces and show that it is an equivalence relation. The logical characterization of bisimulation can be done straightforwardly and generally. (ii) A new and flexible approach to approximation based on averaging can be given. This vastly generalizes and streamlines the idea of using conditional expectations to compute approximation. (iii) It is possible to show that there is a minimal bisimulation equivalent to a process obtained as the limit of the finite approximants.

1 Introduction

Markov processes with continuous state spaces or continuous time evolution or both arise naturally in many areas of computer science: robotics, performance evaluation, modelling and simulation for example. For discrete systems there was a pioneering treatment of probabilisitic bisimulation and logical characterization [1]. The continuous case, however, was neglected for a time. For a little over a decade now, there has been significant activity among computer scientists [2,3,4] [5] [6,7,8] [9,10] as it came to be realized that ideas from process algebra, like bisimulation and the existence of a modal characterization, would be useful for the study of continuous systems.

In [4] a theory of approximation for LMPs was initiated. Finding finite approximations is vital to give a computational handle on such systems. The previous work was characterized by rather intricate proofs that did not seem to follow from basic ideas in any straightforward way. For example, the logical characterization of (probabilistic) bisimulation requires subtle properties of analytic spaces.

In the present paper we take an entirely new approach, in some ways "dual" to the normal view of probabilistic transition systems. We think of a Markov process as a transformer of functions, rather than as a transformer of the state. Thus, instead of working directly with a Markov kernel $\tau(s, A)$ that takes a state s to a probability distribution over the state space, we think of a Markov process

[⋆] Research supported by NSERC and Office of Naval Research.
[⋆⋆] Research supported by a Royal Society – Wolfson Research Merit Award.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 127–138, 2009.

as transforming a function f into a new function $\int f(s')\tau(s, \mathrm{d}s')$ over the state space. This is the probabilistic analogue of working with predicate transformers, a point of view advocated by Kozen [11].

This new way of looking at things leads to three new results:

1. It is possible to define bisimulation on general spaces – not just on analytic spaces – and show that it is an equivalence relation with easy categorical constructions. The logical characterization of bisimulation can also be done generally, and with no complicated measure theoretic arguments.
2. A new and flexible approach to approximation based on averaging can be given. This vastly generalizes and streamlines the idea of using conditional expectations to compute approximation [5].
3. It is possible to show that there is a minimal bisimulation equivalent to a process obtained as the limit of the finite approximants.

2 Preliminary Definitions

Given a measurable space (X, Σ) with a measure μ we say two measurable functions are μ-equivalent if they differ on a set of μ-measure zero. Given two measurable real-valued functions f and g on X, we say $f \leq_\mu g$ if f is less than g except maybe on a set of measure zero. For $B \in \Sigma$, we let $\mathbf{1}_B$ be the indicator function of the set B. $L_1(X, \mu)$ stands for the space of *equivalence classes* of integrable functions. Similarly we write $L_1^+(X, \mu)$ for equivalence classes of functions that are positive μ-almost everywhere. $L_\infty(X, \mu)$ is the space of equivalence classes of μ-almost everywhere uniformly bounded functions on X, and $L_\infty^+(X, \mu)$ are the μ-almost everywhere positive functions of that space. Given two measures ν, μ on (X, Σ), if we have, for all $A \in \Sigma$, that $\mu(A) = 0 \Rightarrow \nu(A) = 0$, we say that ν is absolutely continuous with respect to μ, and write $\nu \ll \mu$.

Theorem 1. *[12] If $\nu \ll \mu$, where ν, μ are finite measures on (X, Σ) there is a positive measurable function h on X such that for every $B \in \Sigma$*

$$\nu(B) = \int_B h \, \mathrm{d}\mu \ .$$

The function h is defined uniquely, up to a set of μ-measure 0.

The function h is called the Radon-Nikodym derivative of ν with respect to μ; we write $\frac{\mathrm{d}\nu}{\mathrm{d}\mu}$ for the Radon-Nikodym derivative of the measure ν with respect to μ. Note that $\frac{\mathrm{d}\nu}{\mathrm{d}\mu} \in L_1(X, \mu)$.

Given a function $f \in L_1^+(X, \mu)$, we let $f \rhd \mu$ be the measure which has density f with respect to μ. According to the Radon-Nikodym theorem, given $\nu \ll \mu$, we have $\frac{\mathrm{d}\nu}{\mathrm{d}\mu} \rhd \mu = \nu$, and given $f \in L_1^+(X, \mu)$, $\frac{\mathrm{d}f \rhd \mu}{\mathrm{d}\mu} = f$ These two identities just say that the operations $- \rhd \mu$ and $\frac{\mathrm{d}}{\mathrm{d}\mu}$ are inverses of each other as operations from $L_1^+(X, \mu)$ to the space of finite measures on X.

Let **Prb** be the category of probability spaces and measurable maps; we will usually suppress the σ-algebra. There are no conditions relating to the measures but the categories of interest will be subcategories where the morphisms do have extra conditions related to the measures. Given a map $\alpha : (X,p) \longrightarrow (Y,q)$ in **Prb**, where p and q are probability measures, we denote by $M_\alpha(p)$ the image measure of p by α onto Y.

One normally works with vector spaces, but it is more convenient to work with cones. The following definition is due to Selinger [13].

Definition 1. *A* cone *is a set V on which a commutative and associative binary operation, written $+$, is defined and has a 0. Multiplication by positive real numbers is defined and it distributes over addition. The following cancellation law holds:*

$$\forall u, v, w \in V, v + u = w + u \Rightarrow v = w \ .$$

The following strictness property also holds: $v + w = 0 \Rightarrow v = w = 0$.

Cones come equipped with a natural partial order. If $u, v \in V$, a cone, one says $u \leq v$ if and only if there is an element $w \in V$ such that $u + w = v$. One can also put a norm on a cone, with the additional requirement that the norm be monotone with respect to the partial order.

Definition 2. *A* (ω-)complete normed cone *is a normed cone such that its unit ideal is a (ω-)dcpo.*

A (ω-)*continuous* linear map between two cones is one that preserves sups of (ω-)directed sets, i.e. is Scott-continuous. Note that in a (ω-)complete normed cone, the norm is (ω-)Scott-continuous. All the cones that we work with are complete normed cones. For instance, $L_\infty^+(X)$ is a complete normed cone, with the norm $\|{-}\|_\infty$ the usual essential supremum norm.

Let (X, Σ) be a measure space. We write $\mathcal{L}^+(X)$ for the cone of bounded measurable maps from X to $\mathbb{R}_+$; in this cone we have functions, not equivalence classes of functions. Let (X, Σ, μ) be a measure space. We define the cone $\mathcal{M}^{\leq K\mu}(X)$ to be the cone of all measures on (X, Σ) which are uniformly less than a multiple of the measure μ; this minimal multiple is the norm on this cone. The normed cones $\mathcal{M}^{\leq K\mu}(X)$ and $L_\infty^+(X, \Sigma, \mu)$ are isomorphic via the two maps $\frac{\mathrm{d}(-)}{\mathrm{d}\mu}$ and $(-) \rhd \mu$, which are also norm-preserving.

Markov processes can be viewed as linear maps on function spaces. Given τ a Markov process on X, we define $\hat{\tau} : \mathcal{L}^+(X) \longrightarrow \mathcal{L}^+(X)$, for $f \in \mathcal{L}^+(X)$, $x \in X$, as $\hat{\tau}(f)(x) = \int_X f(z)\tau(x, dz)$. This map is well-defined, as per our definition above, $\hat{\tau}(\mathbf{1}_B)$ is measurable for every $B \in \Sigma$. In fact, $\hat{\tau}(\mathbf{1}_B)(x) = \tau(x, B)$ is the probability of jumping from x to B. $\hat{\tau}$ is also linear and continuous and thus $\hat{\tau}(f)$ is measurable for any measurable f. Conversely, any such functional L with $L(\mathbf{1}_X) \leq \mathbf{1}_X$ is a Markov process. *From now on, we shall only consider Markov processes from this functional point of view.*

3 Abstract Markov Processes and Conditional Expectation

In order to reduce the state space, we would like to project the space $\mathcal{L}^+(X)$ onto a smaller space. Let $\Lambda \subseteq \Sigma$ be a sub-σ-algebra, and let p be a finite measure on (X, Σ). We have a positive, linear and continuous map $\mathbb{E}_{\Lambda} : L_1^+(X, \Sigma, p) \longrightarrow L_1^+(X, \Lambda, p)$, the conditional expectation with respect to the sub-σ-algebra Λ. It can be restricted to L_∞^+, as it is a subcone of L_1^+. This map averages the function f over the sets of Λ. However, we cannot use the conditional expectation map in conjunction with Markov processes just yet, as Markov processes are defined as maps on $\mathcal{L}^+$, and not on any L_p^+ space. We therefore make the following definition:

Definition 3. *An* abstract Markov process *(AMP) on a probability space X is a ω-continuous linear map $\tau : L_\infty^+(X) \longrightarrow L_\infty^+(X)$ with $\tau(\mathbf{1}_X) \leq_p \mathbf{1}_X$.*

The condition that $\tau(\mathbf{1}_X) \leq_p \mathbf{1}_X$ is equivalent to requiring that the operator norm of τ be less than one, i.e. that $\|\tau(f)\|_\infty \leq \|f\|_\infty$ for all $f \in L_\infty^+(X)$. This is natural, as the function $\tau(\mathbf{1}_X)$, evaluated at a point x, is the probability of jumping from x to X, which is less than one.

AMPs are often called Markov operators in the literature, and have been first introduced in [14]. The novelty here is that our transition probabilities may be subprobabilities, as in the LMP literature, and we may then examine LMPs from this point of view.

It can be shown that a (usual) Markov process $\tau(x, B)$ on a probability space (X, Σ, p) can be expressed as an AMP if and only if for all $B \in \Sigma$ such that $p(B) = 0$, we have $\tau(x, B) = 0$, p-almost everywhere.

The simplest example of an AMP on a probability space (X, Σ, p) is the identity tranformation on $L_\infty^+(X)$, which sends any $f \in L_\infty^+(X)$ to itself. This AMP corresponds to the Markov process $\delta(x, B) = \mathbf{1}_B(x)$.

We now formalize the notion of conditional expectation. We work in a subcategory of **Prb**, called $\mathbf{Rad}_\infty$, where we require the image measure to be bounded by a multiple of the measure in the codomain; that is, measurable maps $\alpha : (X, \Sigma, p) \longrightarrow (Y, \Lambda, q)$ such that $M_\alpha(p) \leq Kq$ for some real number K.

Let us define an operator $\mathbb{E}_\alpha : L_\infty^+(X, p) \longrightarrow L_\infty^+(Y, q)$, as follows: $\mathbb{E}_\alpha(f) = \frac{\mathrm{d}M_\alpha(f \rhd p)}{\mathrm{d}q}$. As α is in $\mathbf{Rad}_\infty$, the Radon-Nikodym derivative is defined and is in $L_\infty^+(X, p)$. That is, the following diagram commutes by definition:

$$\begin{tikzcd}
L_\infty^+(X,p) \arrow[r, "\rhd p"'] \arrow[d, "\mathbb{E}_\alpha"] & \mathcal{M}^{\leq Kp}(X) \arrow[d, "M_\alpha(-)"] \\
L_\infty^+(Y,q) & \mathcal{M}^{\leq Kq}(Y) \arrow[l, "\frac{\mathrm{d}}{\mathrm{d}q}"']
\end{tikzcd}$$

Note that if (X, Σ, p) is a probability space and $\Lambda \subseteq \Sigma$ is a sub-σ-algebra, then we have the obvious map $\lambda : (X, \Sigma, p) \longrightarrow (X, \Lambda, p)$ which is the identity on the underlying set X. This map is in $\mathbf{Rad}_\infty$ and it is easy to see that $\mathbb{E}_\lambda$ is precisely the conditional expectation onto Λ. Thus the operator $\mathbb{E}_{-}$ truly generalizes conditional expectation. It is easy to show that $\mathbb{E}_{\alpha \circ \beta} = \mathbb{E}_\alpha \circ \mathbb{E}_\beta$ and thus $\mathbb{E}_{-}$ is functorial.

Let us define, for any map $\alpha : (X, p) \longrightarrow (Y, q)$ in $\mathbf{Rad}_\infty$, a function $d(\alpha) = \mathbb{E}_\alpha(\mathbf{1}_X) = \frac{\mathrm{d}M_\alpha(p)}{\mathrm{d}q}$. Note that $d(\alpha)$ is in $L_\infty^+(Y, q)$. It can be shown that the operator norm of $\mathbb{E}_\alpha$ is $\|d(\alpha)\|_\infty$.

Given an AMP on (X, p) and a map $\alpha : (X, p) \longrightarrow (Y, q)$ in $\mathbf{Rad}_\infty$, we thus have the following approximation scheme:

$$\begin{tikzcd}
L_\infty^+(Y,q) \arrow[r, dotted, "\alpha(\tau)"] \arrow[d, "(-)\circ\alpha"'] & L_\infty^+(Y,q) \\
L_\infty^+(X,p) \arrow[r, "\tau"] & L_\infty^+(X,p) \arrow[u, "\mathbb{E}_\alpha"']
\end{tikzcd}$$

Note that $\|\alpha(\tau)\| \leq \|(-) \circ \alpha\| \cdot \|\tau\| \cdot \|\mathbb{E}_\alpha\| = \|\tau\| \cdot \|d(\alpha)\|_\infty$. Here the norm of $(\cdot) \circ \alpha$ is 1. As an AMP has a norm less than 1, we can only be sure that a map α yields an approximation for every AMP on X if $\|d(\alpha)\|_\infty \leq 1$. We call the AMP $\alpha(\tau)$ the projection of τ on Y.

4 Bisimulation

The notion of probabilistic bisimulation was introduced by Larsen and Skou [1] for discrete spaces and by Desharnais et al. [2] for continuous spaces. Subsequently a dual notion called event bisimulation or probabilistic co-congruence was defined independently by Danos et al. [9] and by Bartels et al. [15]. The idea of event bisimulation was that one should focus on the measurable sets rather than on the points. This meshes exactly with the view here.

Definition 4. *Given a (usual) Markov process (X, Σ, τ), an event-bisimulation is a sub-σ-algebra Λ of Σ such that (X, Λ, τ) is still a Markov process [9].*

The only additional condition that needs to be respected for this to be true is that the Markov process $\tau(x, A)$ is Λ-measurable for a fixed $A \in \Lambda$. Translating this definition in terms of AMPs, this implies that the AMP τ sends the subspace $L_\infty^+(X, \Lambda, p)$ to itself, and so that the following commutes:

$$\begin{tikzcd}
L_\infty^+(X,\Sigma) \arrow[r, "\tau"'] & L_\infty^+(X,\Sigma) \\
L_\infty^+(X,\Lambda) \arrow[r, "\tau"'] \arrow[u, hook] & L_\infty^+(X,\Lambda) \arrow[u, hook]
\end{tikzcd}$$

A generalization to the above would be a $\mathbf{Rad}_\infty$ map α from (X, Σ, p) to (Y, Λ, q), respectively equipped with AMPs τ and ρ, such that the following commutes:

$$\begin{tikzcd}
L_\infty^+(X,p) \arrow[r, "\tau"'] & L_\infty^+(X,p) \\
L_\infty^+(Y,q) \arrow[r, "\rho"'] \arrow[u, "(-)\circ\alpha"'] & L_\infty^+(Y,q) \arrow[u, "(-)\circ\alpha"']
\end{tikzcd}$$

We will call such a map a *zigzag*. Note that if there is a zigzag from X to Y, then the AMP on Y is very closely related to the projection $\alpha(\tau)$ on Y. Indeed, we have the following diagram:

$$\begin{tikzcd}
L_\infty(Y)^+ \arrow[rrr, "\rho"] \arrow[dr, "(-)\circ\alpha"] \arrow[dd, equal] & & & L_\infty(Y)^+ \arrow[dl, "(-)\circ\alpha"'] \arrow[dd, "(-)\cdot d(\alpha)"] \\
& L_\infty(X)^+ \arrow[r, "\tau"'] & L_\infty(X)^+ \arrow[dr, "\mathbb{E}_\alpha"'] & \\
L_\infty(Y)^+ \arrow[ur, "(-)\circ\alpha"'] \arrow[rrr, "\alpha(\tau)"'] & & & L_\infty(Y)^+
\end{tikzcd}$$

We have that $\mathbb{E}_\alpha(f \circ \alpha) = f \cdot d(\alpha)$ from a lemma in the full paper. This implies that $\alpha(\tau) = \rho \cdot d(\alpha)$. In particular, if $d(\alpha) = \mathbf{1}_Y$ - which happens if $M_\alpha(p) = q$ - then ρ is equal to $\alpha(\tau)$. Note that the condition $M_\alpha(p) = q$ means that the image measure is precisely the measure in the codomain of α.

We have developed the above theory in a very general setting where the maps between state spaces need only to respect some conditions; for instance, in $\mathbf{Rad}_\infty$, that the image measure be bounded by a multiple of the measure in the target space. From now on we shall considerably restrict the maps between the state spaces. Indeed, we have seen above that zigzags and projections coincided exactly given that the map of the state spaces was particularly well-behaved.

Definition 5. *A map $\alpha : (X, p) \longrightarrow (Y, q)$ in* **Prb** *is said to be* measure-preserving *if $M_\alpha(p) = q$.*

Clearly these maps form a subcategory of **Prb**. In effect, this ensures that the map α is essentially surjective. However, there is no reason why we would consider essentially surjective maps which are not surjective in the usual sense. We shall thus consider the subcategory of **Prb** consisting of the surjective measure-preserving maps. We will also augment this category with additional structure relevant to our situation.

We define the category **AMP** of abstract Markov processes as follows. The objects consist of probability spaces (X, Σ, p), together with an abstract Markov process τ on X. The arrows $\alpha : (X, \Sigma, p, \tau) \longrightarrow (Y, \Lambda, q, \rho)$ are surjective measure-preserving maps from X to Y such that $\alpha(\tau) = \rho$. In words, this means that the Markov processes defined on the codomain are precisely the projection of the Markov processes τ on the domain through α. When working in this category, we will often denote objects by the state space, when the context is clear.

One can define a preorder on **AMP** as follows: given two AMPs (X, Σ, p, τ) and (Y, Λ, q, ρ), we say that $Y \preceq X$ if there is an arrow $\alpha : (X, \Sigma, p, \tau) \longrightarrow (Y, \Lambda, q, \rho)$ in **AMP**.

Definition 6. *We say that two objects of* **AMP**, *(X, Σ, p, τ) and (Y, Λ, q, ρ), are* bisimilar *if there is a third object (Z, Γ, r, π) with a pair of zigzags*

$$\alpha : (X, \Sigma, p, \tau) \longrightarrow (Z, \Gamma, r, \pi)$$
$$\beta : (Y, \Lambda, q, \rho) \longrightarrow (Z, \Gamma, r, \pi)$$

making a cospan diagram

$$\begin{tikzcd}
(X, \Sigma, p, \tau) \arrow[dr, "\alpha"] & & (Y, \Lambda, q, \rho) \arrow[dl, "\beta"'] \\
& (Z, \Gamma, r, \pi) &
\end{tikzcd}$$

Note that the identity function on an AMP is a zigzag, and thus that any zigzag between two AMPs X and Y implies that they are bisimilar.

The great advantage of cospans is that one needs pushouts to exist rather than pullbacks (or weak pullbacks); pushouts are much easier to construct. The following theorem shows that bisimulation is an equivalence.

Theorem 2. *Let $\alpha : X \longrightarrow Y$ and $\beta : X \longrightarrow Z$ be a span of zigzags. Then the pushout W exists and the pushout maps $\delta : Y \longrightarrow W$ and $\gamma : Z \longrightarrow W$ are zigzags.*

Corollary 1. *Bisimulation is an equivalence relation on the objects of* **AMP**.

It turns out that there is a "smallest" bisimulation. Given an AMP (X, Σ, p, τ), one question one may ask is whether there is a "smallest" object $(\tilde{X}, \Xi, r, \xi)$ in **AMP** such that, for every zigzag from X to another AMP (Y, Λ, q, ρ), there is a zigzag from (Y, Λ, q, ρ) to $(\tilde{X}, \Xi, r, \xi)$. It can be shown that such an object exists.

Proposition 1. *Let $\{\alpha_i : (X, \Sigma, p, \tau) \longrightarrow (Y_i, \Lambda_i, q_i, \rho_i)\}$ be the set of all zigzags in* **AMP** *with domain (X, Σ, p, τ). This yields a generalized pushout diagram, and as in Theorem 2, the pushout $(\tilde{X}, \Xi, r, \xi)$ exists and the pushout maps are zigzags.*

This object has important uniqueness properties.

Corollary 2. *Up to isomorphism, the object $(\tilde{X}, \Xi, r, \xi)$ the unique bottom element of $\mathbf{ZZ}_X$, the collection of all zigzags with X as domain. If (W, Ω, q, ρ) is another AMP such that there is a zigzag μ from $\tilde{X}$ to W, then μ is an isomorphism.*

Thus, we can say that $\tilde{X}$ is the meet (or infimum) of all objects Y_i which are bisimilar to the AMP X, with respect to the preorder $\preceq$. This "smallest" object is given in an abstract way; however, it can be constructed explicitly. Its construction is closely linked to a modal logic.

The logical characterization result from LMPs can easily be recast in the context of AMPs. We skip the proofs but give the basic definitions. Let us fix a finite set of labels $\mathcal{A}$ once and for all. We can then speak of objects in a category $\mathbf{AMP}_{\mathcal{A}}$ of *labelled* AMPs, consisting of a probability space (x, Σ, p) and a set of AMPs τ_a indexed by $\mathcal{A}$.

Definition 7. *We define a logic $\mathcal{L}$ as follows, with $a \in \mathcal{A}$:*

$$\mathcal{L} ::= \mathbf{T} | \phi \wedge \psi | \left\langle a \right\rangle_q \psi$$

Given a labelled AMP (X, Σ, p, τ_a), we associate to each formula ϕ a measurable set $[\![\phi]\!]$, defined recursively as follows:

$$[\![\mathbf{T}]\!] = X \quad [\![\phi \wedge \psi]\!] = [\![\phi]\!] \cap [\![\psi]\!]$$
$$\left[\!\!\left[\left\langle a \right\rangle_q \psi \right]\!\!\right] = \left\{ s : \tau_a(\mathbf{1}_{[\![\psi]\!]})(s) > q \right\}$$

We let $[\![\mathcal{L}]\!]$ denote the measurable sets obtained by all formulas of $\mathcal{L}$.

Theorem 3. (From [9]) *Given a labelled AMP (X, Σ, p, τ_a), the σ-field $\sigma([\![\mathcal{L}]\!])$ generated by the logic $\mathcal{L}$ is the smallest event-bisimulation on X. That is, the map $i : (X, \Sigma, p, \tau_a) \longrightarrow (X, \sigma([\![\mathcal{L}]\!]), p, \tau_a)$ is a zigzag; furthermore, given any zigzag $\alpha : (X, \Sigma, p, \tau_a) \longrightarrow (Y, \Lambda, q, \rho_a)$, we have that $\sigma([\![\mathcal{L}]\!]) \subseteq \alpha^{-1}(\Lambda)$.*

Hence, the σ-field obtained on X by the "smallest object" $\tilde{X}$ is precisely the σ-field we obtain from the logic.

5 Approximations of AMPs

Given an arbitrary AMP, it may be very difficult to study its behavior if its state space is very large or uncountable. It is therefore crucial to devise a way to reduce the state space to a manageable size.

In this section, we let the measurable map $i_\Lambda : (X, \Sigma) \longrightarrow (X, \Lambda)$ be the identity on the set X, restricting the σ-field. The resulting AMP morphism is denoted as $i_\Lambda : (X, \Sigma, p, \tau) \longrightarrow (X, \Lambda, p, \Lambda(\tau))$, as p is just restricted on a smaller σ-field, with $\Lambda(\tau)$ being the projection of τ on the smaller σ-field Λ.

Let (X, Σ, p, τ_a) be a labelled AMP. Let $\mathcal{P}$ be a finite set of rationals in $[0, 1]$; we will call it a *rational partition*. We define a family of finite π-systems [12], subsets of Σ, as follows:

$$\Phi_{\mathcal{P},0} = \{X, \emptyset\}$$
$$\Phi_{\mathcal{P},n} = \pi\left(\left\{\tau_a(\mathbf{1}_A)^{-1}(q_i, 1] : q_i \in \mathcal{P}, A \in \Phi_{\mathcal{P},n-1}, a \in \mathcal{A}\right\} \cup \Phi_{\mathcal{P},n-1}\right)$$

where $\pi(\Omega)$ is the π-system generated by the class of sets Ω.

For each pair $(\mathcal{P}, M)$ consisting of a rational partition and a natural number, we define a σ-algebra $\Lambda_{\mathcal{P},M}$ on X as $\Lambda_{\mathcal{P},M} = \sigma\left(\Phi_{\mathcal{P},M}\right)$, the σ-algebra generated by $\Phi_{\mathcal{P},M}$. We shall call each pair $(\mathcal{P}, M)$ consisting of a rational partition and a natural number an *approximation pair*. These σ-algebras have a very important property:

Proposition 2. *Given any labelled AMP (X, Σ, p, τ_a), the σ-field $\sigma\left(\bigcup \Lambda_{\mathcal{P},M}\right)$, where the union is taken over all approximation pairs, is precisely the σ-field $\sigma\,[\![\mathcal{L}]\!]$ obtained from the logic.*

Consider the σ-algebra $\Lambda_{\mathcal{P},M}$. We have the map

$$i_{\Lambda_{\mathcal{P},M}} : (X, \Sigma, p, \tau_a) \longrightarrow (X, \Lambda_{\mathcal{P},M}, p, \Lambda_{\mathcal{P},M}(\tau_a)) \ .$$

Now since $\Lambda_{\mathcal{P},M}$ is finite, it is atomic, and so it partitions our state space X, yielding an equivalence relation. Quotienting by this equivalence relation gives a map $\pi_{\mathcal{P},M} : (X, \Lambda_{\mathcal{P},M}, p, \Lambda_{\mathcal{P},M}(\tau_a)) \longrightarrow (\hat{X}_{\mathcal{P},M}, \Omega, q, \rho_a)$, where $\hat{X}_{\mathcal{P},M}$ is the (finite!) set of atoms of $\Lambda_{\mathcal{P},M}$ and Ω is just the powerset of $\hat{X}_{\mathcal{P},M}$. The measure q is just the image measure and AMPs ρ_a are the projections $\pi_{\mathcal{P},M}(\tau_a)$. Note that $\pi_{\mathcal{P},M}$ is a zigzag as ${\pi_{\mathcal{P},M}}^{-1}(\Omega) = \Lambda_{\mathcal{P},M}$.

As the σ-field on $\hat{X}_{\mathcal{P},M}$ is its powerset, we will refrain from writing Ω when involving a finite approximation. We thus have an approximation map $\phi_{\mathcal{P},M} = \pi_{\mathcal{P},M} \circ i_{\Lambda_{\mathcal{P},M}}$ from our original state space to a finite state space; furthermore it is clear that this map is an arrow in **AMP**.

Let us define an ordering on the approximation pairs by $(\mathcal{P}, M) \leq (\mathcal{Q}, N)$ if $\mathcal{Q}$ refines $\mathcal{P}$ and $M \leq N$. This order is natural as $(\mathcal{P}, M) \leq (\mathcal{Q}, N)$ implies $\Lambda_{\mathcal{P},M} \subseteq \Lambda_{\mathcal{Q},N}$, which is clear from the definition. Thus, this poset is a directed set: given $(\mathcal{P}, M)$ and $(\mathcal{Q}, N)$ two approximation pairs, then the approximation pair $(\mathcal{P} \cup \mathcal{Q}, \max(M, N))$ is an upper bound.

Given two approximation pairs such that $(\mathcal{P}, M) \leq (\mathcal{Q}, N)$, we have a map $i_{(\mathcal{Q},N),(\mathcal{P},M)} : (X, \Lambda_{\mathcal{Q},N}, \Lambda_{\mathcal{Q},N}(\tau_a)) \longrightarrow (X, \Lambda_{\mathcal{P},M}, \Lambda_{\mathcal{P},M}(\tau_a))$ which is well defined by the inclusion $\Lambda_{\mathcal{P},M} \subseteq \Lambda_{\mathcal{Q},N} \subseteq \Sigma$. We therefore have a projective system of such maps indexed by our poset of approximation pairs. It can be shown that these maps induce a map on the finite approximation spaces $\hat{X}_{\mathcal{P},M}$, say $j_{(\mathcal{Q},N),(\mathcal{P},M)} : (\hat{X}_{\mathcal{Q},N}, \phi_{\mathcal{Q},N}(\tau_a)) \longrightarrow (\hat{X}_{\mathcal{P},M}, \phi_{\mathcal{P},M}(\tau_a))$, such that the map $\phi_{(\mathcal{P},M)}$ factors through the map $\phi_{(\mathcal{Q},N)}$ as $\phi_{(\mathcal{P},M)} = j_{(\mathcal{Q},N),(\mathcal{P},M)} \circ \phi_{(\mathcal{Q},N)}$. Hence, the maps $j_{(\mathcal{Q},N),(\mathcal{P},M)}$ together with the approximants $\hat{X}_{(\mathcal{P},M)}$ also form a projective system with respect to our poset of approximation pairs.

A result of Choksi [16] allows us to construct projective limits of measure spaces. We consider the underlying probability spaces of the finite approximants of a labelled AMP (X, Σ, p, τ_a).

Proposition 3. *(From [16]) The probability spaces of finite approximants $\hat{X}_{\mathcal{P},M}$ of an AMP (X, Σ, p, τ_a), indexed by the approximation pairs, form a projective system of surjective measure-preserving maps; furthermore, its projective limit* $(\operatorname{proj\,lim} \hat{X}, \Gamma, \gamma)$ *exists in* **Prb**

Concretely, $\operatorname{proj\,lim} \hat{X}$ is the projective limit in **Set**. Thus we have the usual projection maps, appropriately restricted, $\psi_{\mathcal{P},M} : \operatorname{proj\,lim} \hat{X} \longrightarrow \hat{X}_{\mathcal{P},M}$ for every approximation pair. We also have, in **Set**, a unique map $\kappa : X \longrightarrow \operatorname{proj\,lim} \hat{X}$ such that $\psi_{\mathcal{P},M} \circ \kappa = \phi_{\mathcal{P},M}$.

The σ-field Γ is the smallest σ-field making all of the maps $\psi_{\mathcal{P},M}$ measurable. We must show that $\kappa^{-1}(\Gamma) \subseteq \Sigma$. We shall show something stronger.

Proposition 4. *The σ-field $\kappa^{-1}(\Gamma)$ is precisely equal to $\sigma [\![\mathcal{L}]\!]$; in particular κ is measurable.*

Proof. The σ-field Γ is generated by the preimages of $\psi_{\mathcal{P},M}$. Taking the preimage of this through κ is equivalent to taking the preimage through the approximation maps $\phi_{\mathcal{P},M}$, which is exactly $\Lambda_{\mathcal{P},M}$. These σ-fields generate $\sigma \llbracket \mathcal{L} \rrbracket$.

We now need to show that κ is measure-preserving. γ was defined so that the maps $\psi_{\mathcal{P},M}$ were measure preserving [16]; thus γ and $M_\kappa(p)$ agree on all subsets of $\operatorname{proj}\lim \hat{X}$ which are the preimage of a measurable set in a finite approximant $\hat{X}_{\mathcal{P},M}$. Since these sets generate Γ, and form a π-system, the uniqueness of measure theorem [12] implies that $\gamma = M_\kappa(p)$.

Finally, we define the AMP ζ_a on $\operatorname{proj}\lim \hat{X}$ in the obvious way; that is, as the projection of τ_a through κ. Then the projection of ζ_a onto the finite approximants through $\psi_{\mathcal{P},M}$ is precisely equal to ρ_a as they were previously defined, since $\psi_{\mathcal{P},M} \circ \kappa = \phi_{\mathcal{P},M}$. Thus, the projective limit of measure spaces can be extended to a projective limit of AMPs.

Proposition 5. *The universal map κ obtained from the projective limit is a zigzag.*

Therefore, if we let $(\tilde{X}, \Xi, r, \xi_a)$ be the smallest bisimulation obtained as in proposition 1, we have a zigzag $\omega : (\operatorname{proj}\lim \hat{X}, \Gamma, \gamma, \zeta_a) \longrightarrow (\tilde{X}, \Xi, r, \xi_a)$. This zigzag must be an isomorphism of σ-fields as Ξ is the smallest possible σ-field on $\tilde{X}$. We can show that there is a zigzag going in the other direction.

Proposition 6. *Let $\alpha : (X, \Sigma, p, \tau_a) \longrightarrow (Y, \Theta, q, \rho_a)$ be a zigzag. Then these two AMPs have the same finite approximants. In particular, two bisimilar AMPs have the same finite approximants.*

We conclude with the main result.

Theorem 4. *Given a labelled AMP (X, Σ, p, τ_a), the projective limit of its finite approximants $(\operatorname{proj}\lim \hat{X}, \Gamma, \gamma, \zeta_a)$ is isomorphic to its smallest bisimulation $(\tilde{X}, \Xi, r, \xi_a)$.*

6 Related Work and Conclusions

The main contribution of the present work is to show how one can obtain a powerful and general notion of approximation of Markov processes using the dualized view of Markov processes as transformers of random variables (measurable functions). We view Markov processes as "predicate transformers". Our main result is to show that this way of working with Markov processes greatly simplifies the theory: bisimulation, logical characterization and approximation. Working with the functions (properties) one is less troubled by having to deal with things that are defined only "almost everywhere" as happens when one works with states.

A very nice feature of the theory is the ability to show that a minimal bisimulation exists. Furthermore, this minimal object can be constructed as the projective limit of finite approximants.

Previous work on bisimulation-based approximation of Markov processes began with a paper by Desharnais et al. [4] where the approximation scheme was based on an unfolding of the transition system. The main technical result is that every formula satisfied by a process is satisfied by one of its finite approximants.

In [5] the idea of approximating by averaging was introduced and the main tool used to compute the approximation is the conditional expectation. The mathematical theory developed there is the bare beginnings of the theory developed here. There also the idea of averaging by conditional approximation was used; but none of the results relating to bisimulation and especially the result about constructing a minimal bisimulation by taking a projective limit of finite approximants was known. Moving to AMPs was crucial for all this to work.

One of the problems with any of the approximation schemes is that they are hard to implement. In a recent paper [7], an approach based on Monte Carlo approximation was used to "approximate the approximation." The point is that it is hard to compute τ^{-1} in practice. Our most pressing future work is to explore the possibility of implementing the approximation scheme and, perhaps using some technique like Monte Carlo, to compute the approximations concretely. It is curious that the abstract version of Markov processes makes it more likely that one can compute approximations in practice and is another argument in favour of a "pointless" view of processes.

References

1. Larsen, K.G., Skou, A.: Bisimulation through probablistic testing. Information and Computation 94, 1–28 (1991)
2. Desharnais, J., Edalat, A., Panangaden, P.: Bisimulation for labeled Markov processes. Information and Computation 179(2), 163–193 (2002)
3. de Vink, E., Rutten, J.J.M.M.: Bisimulation for probabilistic transition systems: A coalgebraic approach. Theoretical Computer Science 221(1/2), 271–293 (1999)
4. Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Approximating labeled Markov processes. Information and Computation 184(1), 160–200 (2003)
5. Danos, V., Desharnais, J., Panangaden, P.: Conditional expectation and the approximation of labelled Markov processes. In: Amadio, R.M., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 477–491. Springer, Heidelberg (2003)
6. Ferns, N., Panangaden, P., Precup, D.: Metrics for Markov decision processes with infinite state spaces. In: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, pp. 201–208 (2005)
7. Bouchard-Côté, A., Ferns, N., Panangaden, P., Precup, D.: An approximation algorithm for labelled Markov processes: towards realistic approximation. In: Proceedings of the 2nd International Conference on the Quantitative Evaluation of Systems (QEST), pp. 54–61 (2005)
8. Cattani, S., Segala, R., Kwiatkowska, M., Norman, G.: Stochastic transition systems for continuous state spaces and non-determinism. In: Sassone, V. (ed.) FOSSACS 2005. LNCS, vol. 3441, pp. 125–139. Springer, Heidelberg (2005)
9. Danos, V., Desharnais, J., Laviolette, F., Panangaden, P.: Bisimulation and cocongruence for probabilistic systems. Information and Computation 204(4), 503–523 (2006); Seventh Workshop on Coalgebraic Methods in Computer Science 2004

10. Goubault-Larrecq, J.: Continuous capacities on continuous state spaces. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 764–776. Springer, Heidelberg (2007)
11. Kozen, D.: A probabilistic PDL. Journal of Computer and Systems Sciences 30(2), 162–178 (1985)
12. Billingsley, P.: Probability and Measure. Wiley Interscience, Hoboken (1995)
13. Selinger, P.: Towards a quantum programming language. Mathematical Structures in Computer Science 14(4), 527–586 (2004)
14. Hopf, E.: The general temporally discrete Markoff process. J. Rational Math. Mech. Anal. 3, 13–45 (1954)
15. Bartels, F., Sokolova, A., de Vink, E.: A hierarchy of probabilistic system types. Theoretical Computer Science 327, 3–22 (2004)
16. Choksi, J.: Inverse limits on measure spaces. Proc. London Math. Soc 8(3), 321–342 (1958)

The Theory of Stabilisation Monoids and Regular Cost Functions[*]

Thomas Colcombet

LIAFA/CNRS/Université Paris 7, Denis Diderot, France

Abstract. We introduce the notion of regular cost functions: a quantitative extension to the standard theory of regular languages.

We provide equivalent characterisations of this notion by means of automata (extending the nested distance desert automata of Kirsten), of history-deterministic automata (history-determinism is a weakening of the standard notion of determinism, that replaces it in this context), and a suitable notion of recognisability by stabilisation monoids. We also provide closure and decidability results.

1 Introduction

When considering standard regular languages (say on finite words), some results appear as cornerstones on which the whole theory is constructed. The first such kind of results are the equivalences between many different formalisms: non-deterministic automata, deterministic automata, recognisability by monoids, regular expressions, etc. The second one consists in the numerous closure properties that regular languages enjoy: union, intersection, projection (mapping under a length-preserving morphism), complementation, etc. From these facts one can derive a third kind of results: the equivalence with logical formalisms such as monadic (second-order) logic. Finally, all these properties do not come at an unaffordable price: emptiness is decidable, and hence the satisfaction of the logic is also decidable.

In this paper, we present a quantitative extension to the standard notion of regularity in which those cornerstone results still hold. We consider a quantitative notion of regularity which allows to attach non-negative integer values to words, such as the number of occurrences of a pattern, the length of segments, etc. One also possess some freedom for combining those values, e.g., using minimum or maximum. One can for instance describe the maximum number of occurrences of letter a that are not separated by a letter b. Those integer values are considered modulo an equivalence which preserves the existence of bounds, but does not preserve exact values – as opposed to the usual way one considers quantitative forms of automata. This is the price to pay for keeping all equivalences and closure properties.

Originally, this work aimed at unifying and reinterpreting some recent results from the literature. Let us review them.

[*] Supported by the ANR project JADE: 'Jeux et Automates, Décidabilité et Extensions'.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 139–150, 2009.

First, in [9] Kirsten gives a new proof to the decidability of the (restricted) star-height problem[1]. This problem is known to be decidable from Hashiguchi [7], but with a very difficult proof. The first part in Kirsten's proof consists in reducing the star-height problem to a problem of limitedness: decide the existence of a bound for some function defined by means of a nested distance desert automata, a new form of automata introduced for this purpose. The second part consists in proving the decidability of this limitedness problem. This is done by turning this automata-related question into an algebraic one: the automaton is translated into a monoid equipped with a stabilisation operator $\sharp$. The limitedness problem becomes easy to decide in this presentation. Kirsten's paper is itself the continuation of a long line of research concerning distance automata, tropical semiring, desert automata, etc. [6,8,11,12,13,14,15,16,17].

Second, the paper [3] provides a study of an extension of the monadic second-order logic over infinite words with new 'bound' quantifiers such as: 'there exists a set of arbitrary large size satisfying some property'. The goal being different, the presentation is also significantly different, and getting results comparable to the ones in the present paper requires a translation that we cannot detail here. However, two new forms of automata are introduced in [3] as intermediate objects in the proofs, namely B-automata and S-automata. The class of B-automata corresponds essentially to the non-nested variant of the nested desert distance automata, while the class of S-automata is a new dual variant. The decidability of limitedness can be derived from this work but with a bad complexity (non-elementary, as opposed to [9]). Independently, B-automata were also introduced in [1] under the name of R-automata, and the decidability of the limitedness problem established using another technique, yielding better complexity.

Other applications of the technique have also been described. Still in this framework, the restricted star-height problem for trees has been shown decidable [4], and the Mostowski hierarchy problem[2] has been reduced to the corresponding limitedness problem over infinite trees [5], which remains open. The existence of a bound on the number of iterations necessary for reaching the fixpoint of a monadic second-order formula over words has been also shown decidable using distance automata [2].

Contribution. Our contribution can be roughly described as 1) a unification of the ideas in [9] and [3], and 2) the development of a suitable mathematical background and the establishment of new results in order to make this theory a complete extension of the standard theory of regular languages. Let us be more precise.

The first contribution lies in the definition of a cost function: cost functions are mappings from words (or from any set in general) to $\omega + 1$ quotiented by a suitable equivalence that preserves the notion of bound ($\approx$ in the paper). In our framework, cost functions can be seen as a refinement of the notion of language (each language can be seen as a cost function, while the converse is not true).

[1] Problem: given a regular language L of words and an integer k, is it possible to describe L with a regular expression using at most k nesting of Kleene stars?.

[2] The hierarchy induced by the number of priorities used by a non-deterministic parity automaton running on infinite trees.

We then introduce B- and S-automata, automata that accept cost functions rather than languages. Those are slight extensions of the automata in [3]. We establish the equivalence of the two forms of automata, via an elementary construction, as well as the equivalence with their history-deterministic form. The new notion of history-determinism is a weakening of the classical notion of determinism (deterministic automata are strictly weaker in this framework). It is needed for the further extension of the theory to trees. Quiet naturally, we call regular the cost functions described by one of these formalisms.

The second aspect of the theory that we develop is the algebraic formalism. We introduce the notion of stabilisation monoids: finite monoids equipped with a stabilisation operator, inspired from [9]. We develop a mathematical framework – new to the knowledge of the author – in order to define the semantics of stabilisation monoids. The key result here is the existence of unique semantics (that we call compatible mappings) for each stabilisation monoid[3]. Building on these notions, we introduce the notion of recognisable cost functions. As we may expect, these happen to be exactly the regular cost functions.

While describing the above objects, we prove the closure of regular cost functions under operations which correspond to union, intersection, projection and dual of projection in the world of languages. We also provide decision procedures subsuming the limitedness results from [9].

Structure of the paper. We present in Section 2 cost functions and the automata part of the theory. We present in Section 3 the algebraic framework, and the equivalent notion of recognisability.

Some Notations

As usual, we denote by ω the set of non-negative integers and $\omega + 1$ the set $\omega \cup \{\omega\}$. Those are ordered by $0 < 1 < \cdots < \omega$. The identity mapping over ω is *id*. Given a set E, E^ω is the set of sequences of ω-length of elements in E. Such sequences will be denoted by bold letters ($\boldsymbol{a}$, $\boldsymbol{b}$,...). We fix a *finite alphabet* $\mathbb{A}$ consisting of *letters*. The set of words over $\mathbb{A}$ is $\mathbb{A}^*$. The empty word is ε. The concatenation of a word u and word v is uv. The length of word u is $|u|$. The number of occurrences of a letter a in u is $|u|_a$.

2 Regular Cost Functions

We introduce in Section 2.1 the notion of cost function. We present B and S-automata in Section 2.2, and their history-deterministic form in Section 2.3. The key duality result is the subject of Section 2.4.

2.1 Cost Functions

A *correction function* is a mapping from ω to ω. From now, the symbols $\alpha, \alpha', \ldots$ implicitly designate correction functions. Given x, y in $\omega + 1$, $x \preccurlyeq_\alpha y$ holds

[3] This result is reminiscent of the one for infinite words stating that each finite Wilke algebra can be uniquely extended into an ω-semigroup.

if $x \leq \overline{\alpha}(y)$ in which $\overline{\alpha}$ is the extension of α with $\overline{\alpha}(\omega) = \omega$. For every set E, $\preccurlyeq_\alpha$ is extended to $(\omega+1)^E$ in a natural way by $f \preccurlyeq_\alpha g$ if $f(x) \preccurlyeq_\alpha g(x)$ for all $x \in E$, or equivalently $f \leq \overline{\alpha} \circ g$. Intuitively, f is dominated by g after it has been 'stretched' by α. One also writes $f \approx_\alpha g$ if $f \preccurlyeq_\alpha g$ and $g \preccurlyeq_\alpha f$.

Some elementary properties of $\preccurlyeq_\alpha$ are:

Fact 1. *If $\alpha \leq \alpha'$ and $f \preccurlyeq_\alpha g$, then $f \preccurlyeq_{\alpha'} g$. If $f \preccurlyeq_\alpha g \preccurlyeq_\alpha h$, then $f \preccurlyeq_{\alpha\circ\alpha} h$.*

Example 1. Over $\omega \times \omega$, maximum and sum are equivalent for the doubling correction function (for short, $(\max) \approx_{\times 2} (+)$). *Proof:* for all $x, y \in \omega$, $\max(x,y) \leq x + y \leq 2 \times \max(x,y)$.

Our second example concerns mappings from sequence of words to ω. Given words $u_1, \dots, u_n \in \{a,b\}^*$, we have $|u_1 \dots u_n|_a \approx_\alpha \max(|K|, \max_{i=1\dots n} |u_i|_a)$ in which K is the set of indices i such that $|u_i|_a \geq 1$ and $\alpha(\theta) = \theta^2$. *Proof:* $\max(|K|, \max_{i=1\dots n} |u_i|_a) \leq |u|_a \leq \Sigma_{i\in K} |u_i|_a \leq (\max(|K|, \max_{i=1\dots n} |u_i|_a))^2$.

One also defines $f \preccurlyeq g$ (resp. $f \approx g$) to hold if $f \preccurlyeq_\alpha g$ (resp. $f \approx_\alpha g$) for some α. A *cost function* (over a set E) is an equivalence class of $\approx$ (i.e., a set of mappings from E to $\omega+1$). The relation $\preccurlyeq$ has other characterisations:

Proposition 1. *For all f, g from E to $\omega+1$, the following items are equivalent:*

- *$f \preccurlyeq g$,*
- *$\forall n \in \omega . \exists m \in \omega . \forall x \in E . g(x) \leq n \to f(x) \leq m$, and;*
- *for all $X \subseteq E$, $g|_X$ is bounded implies $f|_X$ is bounded.*

The last characterisation shows that the relation $\approx$ is an equivalence relation that preserves the existence of bounds. Indeed, all this theory can be seen as an automata theoretic method for proving the existence/non-existence of bounds.

Cost functions over some set E ordered by $\preccurlyeq$ form a lattice. Given a subset $X \subseteq E$, one denotes by χ_X its *characteristic mapping* defined by $\chi_X(x) = 0$ if $x \in X$, ω otherwise. It is easy to see that for all $X, Y \subseteq E$, $\chi_X \preccurlyeq \chi_Y$ iff $Y \subseteq X$. To this respect, the lattice of cost functions is a refinement of the lattice of subsets of E equipped with the superset ordering. Keeping this in mind, the notion of regular cost function developed in the paper is an extension of the standard notion of regular language. This extension is strict as soon as E is infinite: there are cost functions that are not equivalent to any characteristic mapping. Consider for instance the size mapping over words, or the number of occurrences of some letter.

2.2 Automata

We present here the automata model we use. A *cost automaton* (that can be either a *B-automaton* or an *S-automaton*) is a tuple $\langle Q, \mathbb{A}, \mathit{In}, \mathit{Fin}, \Gamma, \Delta \rangle$ in which Q is a finite set of *states*, $\mathbb{A}$ is the alphabet, *In* and *Fin* are respectively the set of *initial* and *final* states, Γ is a finite set of *counters*, and $\Delta \subseteq Q \times A \times \{\epsilon, i, r, c\}^\Gamma \times Q$ is the set of transitions. The idea behind the letter in $\{\epsilon, i, r, c\}^\Gamma$ (called an *action*) is that each counter (the value of which ranges over ω) can either be left unchanged (ϵ), be incremented by one (i), be reset to 0 (r), or be checked (c). A *run* σ of an automaton over a word $a_1 \dots a_n$ is defined as a sequence $q_0, a_1, c_1, q_1, \dots, q_{n-1}, a_n, c_n, q_n$ such that q_0 is initial, q_n is final and for all $i = 1 \dots n$, $(q_{i-1}, a_i, c_i, q_i) \in \Delta$. Given a run σ, each counter $\iota \in \Gamma$ is initialized with value 0 and evolves from left to right according to $c_i(\iota)$: if $c_i(\iota)$ is ϵ or c, the value is left unchanged, if it is i, it is incremented by 1, if it is r, the counter is reset. The set $C(\sigma) \subseteq \omega$ is the set of values taken by the counters when checked (i.e., the value of counter ι when $c_i(\iota) = c$). The difference between B-automata and S-automata comes from their dual semantics, $[\![\cdot]\!]_B$ and $[\![\cdot]\!]_S$ respectively:

$$\begin{aligned}\text{for all } u \in \mathbb{A}^*, \quad [\![\mathcal{A}]\!]_B(u) &= \inf\{\sup C(\sigma) \ : \ \sigma \text{ run over } u\}\ , \\ \text{and,} \quad [\![\mathcal{A}]\!]_S(u) &= \sup\{\inf C(\sigma) \ : \ \sigma \text{ run over } u\}\ ,\end{aligned}$$

in which we use the standard convention that $\inf \emptyset = \omega$ and $\sup \emptyset = 0$. Remark that if $\mathcal{A}$ is a non-deterministic finite automaton in the standard sense, accepting the language L, then it can be seen as a cost automaton without counters. Seen as a B-automaton, $[\![\mathcal{A}]\!]_B(u) = \chi_L$, while seen as an S-automaton $[\![\mathcal{A}]\!]_S(u) = \chi_{\mathbb{A}^* \setminus L}$.

Remark 1 (variants). The other similar automata known from the literature can essentially be seen as special instances of the above formalism. The B-automata and S-automata in [3] use only actions in $\{\epsilon, i, cr\}$ in which cr is an atomic operation that checks the counter and immediately resets it. The models are equivalent but the history-determinism (see below) cannot be achieved for S-automata in this restricted form. The *hierarchical automata* correspond to the case when $\Gamma = \{1, \dots, n\}$ and for all transitions (p, a, c, q), if for all $i \in \Gamma$, if $c(i) \neq \epsilon$ implies $c(j) = r$ for all $j < i$. The *nested distance desert automata* of Kirsten corresponds to hierarchical B-automata that use actions in $\{\epsilon, ic, r\}$ in which ic is an atomic operation which increments the counter and immediately checks it. The *R-automata* in [1] use also actions in $\{\epsilon, ic, r\}$, but without the hierarchical constraint. All those models are equivalent, up to $\approx$.

We conclude the section by showing some easy closure properties. Given a mapping f from $\mathbb{A}^*$ to $\omega + 1$ and a length-preserving morphism h from $\mathbb{A}^*$ to $\mathbb{B}^*$ ($\mathbb{B}$ is another alphabet) the *inf-projection* and *sup-projection* of f with respect to h are the mappings $f_{\inf,h}$ and $f_{\sup,h}$ from $\mathbb{B}^*$ to $\omega + 1$ defined for $v \in \mathbb{B}^*$ by:

$$f_{\inf,h}(v) = \inf\{f(u) \ : \ h(u) = v\} \quad \text{and} \quad f_{\sup,h}(v) = \sup\{f(u) \ : \ h(u) = v\}.$$

By simply adapting the standard constructions for intersection, union, and projection of non-deterministic automata, we get:

Proposition 2. *The mappings accepted by B-automata (resp. S-automata) are closed under* min *and* max*. The mappings accepted by B-automata (resp. S-automata) are closed under inf-projection (resp. sup-projection).*

2.3 History-Determinism

In general B and S-automata cannot be determinised (even modulo $\approx$). We consider here automata which possess a weaker property: history-determinism (note that this notion is meaningful even for other kinds of non-deterministic automata). Informally, a non-deterministic automaton is history-deterministic if it possible to choose deterministically the run while accepting an equivalent function. The subtlety comes from the fact that cost automata do not have a sufficient memory for 'implementing' this deterministic choice. History-determinism can be seen as a semantic notion of determinism as opposed to the standard notion that we can refer to as state-determinism[4]. This notion is required for the extension of the theory to trees.

Formally, let us fix ourselves a cost automaton (either B or S) with unique initial state $\mathcal{A} = \langle Q, \mathbb{A}, \{q_0\}, \mathit{Fin}, \mathit{\Gamma}, \mathit{\Delta}\rangle$. A *translation strategy*[5] for $\mathcal{A}$ is a mapping δ from $\mathbb{A}^* \times \mathbb{A}$ to $\mathit{\Delta}$ which tells deterministically how to construct a run of $\mathcal{A}$ over a word. One defines the *run of $\mathcal{A}$ over the word u driven by δ* inductively as follows: if $u = \varepsilon$, the run is q_0. If u is of the form va, the run is the run of $\mathcal{A}$ over v driven by δ prolonged with the transition $\delta(v, a)$ (if this procedure does not provide a valid run over u, then there is no run driven by δ over this entry). If $\mathcal{A}$ is a B-automaton the value $[\![\mathcal{A}]\!]^{\delta}_{B}(u)$ is $\sup C(\sigma)$ where σ is the run of $\mathcal{A}$ over u driven by δ, and ω if there is no such run. If $\mathcal{A}$ is an S-automaton the value $[\![\mathcal{A}]\!]^{\delta}_{S}(u)$ is $\inf C(\sigma)$ where σ is the run of $\mathcal{A}$ over u driven by δ, and 0 if there is no such run.

A B-automaton is *history-deterministic* if there exists α and for all $n \in \omega$ a translation strategy δ_n such that for all words u, $[\![\mathcal{A}]\!]_{B}(u) \leq n$ implies $[\![\mathcal{A}]\!]^{\delta_n}_{B}(u) \leq \alpha(n)$. An S-automaton is *history-deterministic* if there exists α and for all $n \in \omega$ a translation strategy δ_n such that for all word u, $[\![\mathcal{A}]\!]_{S}(u) \geq \alpha(n)$ implies $[\![\mathcal{A}]\!]^{\delta_n}_{S}(u) \geq n$. In other words, the automaton, when driven by δ, computes an $\approx_\alpha$-equivalent function.

2.4 Duality and Regularity

Duality relates all the above notions together. It is central in the theory.

Theorem 1 (duality). *A cost function over words is accepted by an [history-deterministic] [hierarchical] B-automaton, iff it is accepted by an [history-deterministic] [hierarchical] S-automaton. Those equivalences are effective and of elementary complexity. Such cost functions are called* regular*.*

In fact, the proof of Theorem 1 and Theorem 3 below are interdependent. Indeed, the way to transform a cost function accepted by a B-automaton into a cost function accepted by an S-automaton (and vice-versa) is to transform it first into a recognisable cost function, and only then to construct an S-automaton. The results have been separated in this abstract for being easier to present.

One can remark that in the absence of counters, translating a B-automaton into an S-automaton (and vice-versa) is easy to achieve by using any complementation construction for standard non-deterministic automata. Hence Theorem 1

[4] In state-determinism, given the current state and a letter, there is only one possible transition, while in history-determinism, given the prefix of word seen so far (the history), and a letter, it is possible to uniquely choose one transition.

[5] The name comes from the game theoretic part which is not developed here.

can be seen as a replacement for both the results of complementation and determinisation in the classical theory of regular languages.

Even if not explicitly stated, the equivalence between [hierarchical] B-automata and [hierarchical] S-automata, can be derived from the results in [3]. However, using the proof in [3] entails a long theoretical detour, and the constructions in [3] give a non-elementary blowup in the number of states. Furthermore, the notion of history-determinism has no equivalent in [3].

3 Stabilisation Monoids and Recognisable Cost Functions

In this section we describe our algebraic characterisation of regular cost functions. The core algebraic object is the stabilisation monoid that we describe in Section 3.1. In Sections 3.2, 3.3 and 3.4, we show how to attach semantics to stabilisation monoids. In Section 3.5 we introduce recognisability, state that it is equivalent to regularity and give a decidability result.

3.1 Stabilisation Monoids

A *monoid* $\mathbf{M} = \langle M, \cdot \rangle$ is a set M equipped with an associative operation $\cdot$ that has a *neutral element* 1, i.e., such that $1 \cdot x = x \cdot 1 = x$ for all $x \in M$. One extends the product to products of arbitrary length by defining π from M^* to M by $\pi(\varepsilon) = 1$ and $\pi(ua) = \pi(u) \cdot a$. An idempotent in $\mathbf{M}$ is an element $e \in M$ such that $e \cdot e = e$. One denotes by $E(\mathbf{M})$ the set of idempotents in $\mathbf{M}$. An *ordered monoid* $\langle M, \cdot, \leq \rangle$ is a monoid $\langle M, \cdot \rangle$ together with an order $\leq$ over M such that the product $\cdot$ is *compatible* with $\leq$; i.e., $a \leq a'$ and $b \leq b'$ implies $a \cdot b \leq a' \cdot b'$.

We are now ready to introduce the new notion of stabilisation monoid.

Definition 1. *A* stabilisation monoid *$\langle M, \cdot, \leq, \sharp \rangle$ is an ordered monoid $\langle M, \cdot, \leq \rangle$ together with an operator $\sharp$: $E(\mathbf{M}) \to E(\mathbf{M})$ (called* stabilisation*) such that:*

- *for all $a, b \in M$ with $a \cdot b \in E(\mathbf{M})$ and $b \cdot a \in E(\mathbf{M})$, $(a \cdot b)^\sharp = a \cdot (b \cdot a)^\sharp \cdot b$,*[6]
- *for all $e \in E(\mathbf{M})$, $(e^\sharp)^\sharp = e^\sharp \leq e$;*
- *for all $e \leq f$ in $E(\mathbf{M})$, $e^\sharp \leq f^\sharp$;*
- *$1^\sharp = 1$.*

From now, we consider that all stabilisation monoids are finite.

The intuition is that $e^\sharp$ represents what is the value of e^n when n becomes 'very large'. This idea – which is incompatible with the classical view on monoids – fits well with the following consequences of the definition:

$$\text{for all } e \in E(\mathbf{M}), \qquad e^\sharp = e \cdot e^\sharp = e^\sharp \cdot e = e^\sharp \cdot e^\sharp = (e^\sharp)^\sharp\ .$$

Most of the remaining of the section is devoted to the formalisation of this intuition. This requires the development of a suitable mathematical framework. This approach is then validated by Theorem 2 which associates unique semantics to stabilisation monoids.

[6] This equation states that $\sharp$ is a *consistent mapping* in the sense of [9,10].

3.2 Cost Sequences

In order to give quantitative semantics to stabilisation monoids, the basic object is not the element of the monoid, but sequences of such elements. New relations $\preceq_\alpha$ and $\sim_\alpha$ are used to relate such sequences together. Those are tightly connected to $\preccurlyeq_\alpha$ and $\approx_\alpha$ (see Section 3.3 for a formalisation of this link).

From now, θ and θ' implicitely range over ω. Given an ordered set $(E, \leq)$, a correction function α, and two sequences $\boldsymbol{a}, \boldsymbol{b} \in E^\omega$, define $\boldsymbol{a} \preceq_\alpha \boldsymbol{b}$ to hold when:

$$\forall\theta.\forall\theta'. \quad \alpha(\theta) \leq \theta' \rightarrow \boldsymbol{a}(\theta) \leq \boldsymbol{b}(\theta') \ .$$

We set $\sim_\alpha$ to be $\preceq_\alpha \cap \succeq_\alpha$. The following fact is easy (analogue to Fact 1):

Fact 2. *If $\alpha \leq \alpha'$ and $\boldsymbol{a} \preceq_\alpha \boldsymbol{b}$, then $\boldsymbol{a} \preceq_{\alpha'} \boldsymbol{b}$. If $\boldsymbol{a} \preceq_\alpha \boldsymbol{b} \preceq_\alpha \boldsymbol{c}$, then $\boldsymbol{a} \preceq_{\alpha\circ\alpha} \boldsymbol{c}$.*

The mapping α is used as a parameter of 'precision' for $\sim$ and $\preceq$. The above fact states that using one transitivity step costs precision. (In practice, when doing proofs, we omit the correction function subscript, and rather ensure that the proofs conform to a structural property – very natural at use – ensuring that the length of chains of transitivity steps are bounded.)

A sequence $\boldsymbol{a}$ is said *α-non-decreasing* if $\boldsymbol{a} \preceq_\alpha \boldsymbol{a}$. Fact 3 is for helping intuition: it shows that one can almost think of α-non-decreasing sequences as if those were non-decreasing functions, and simplify the relation $\preceq_\alpha$ in this case:

Fact 3. *Every α-non-decreasing sequence is $\sim_\alpha$-equivalent to a non-decreasing sequence. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are non-decreasing, then $\boldsymbol{a} \preceq_\alpha \boldsymbol{b}$ iff $\boldsymbol{a} \leq \boldsymbol{b} \circ \alpha$.*

The above inequality $\boldsymbol{a} \leq \boldsymbol{b} \circ \alpha$ conveys the important intuition that $\boldsymbol{a}$ is 'dominated' by $\boldsymbol{b}$ after 'shrinking' its coordinates (by α). This has to be compared to the definition of $\preccurlyeq_\alpha$ in which the correction function is used for 'stretching'.

From now, we identify each element $a \in E$ with the sequence constant equal to a. According to Fact 3, the relation $\preceq_\alpha$ (whatever is α) coincide with $\leq$ over those sequences. Hence the α-non-decreasing sequences equipped with the relation $\preceq_\alpha$ can be seen as a refinement of $(E, \leq)$.

We introduce now an important tool: α-monotonic mappings. This notion simplifies a lot the work with the $\preceq_\alpha$ and $\sim_\alpha$ relations. Given two ordered sets $(E, \leq)$ and $(F, \leq)$, a mapping f from E to F^ω is said *α-monotonic* if

$$\forall a, b \in E. \quad a \leq b \rightarrow f(a) \preceq_\alpha f(b) \ .$$

You can remark that in particular, for each $a \in E$, since $a \leq a$, we have $f(a) \preceq_\alpha f(a)$, and hence $f(a)$ is α-non-decreasing. Every α-monotonic f from E to F^ω can be turned into a mapping $\tilde{f}$ from E^ω to F^ω by setting:

$$\text{for all } \boldsymbol{a} \in E^\omega \text{ and all } \theta \in \omega, \qquad \tilde{f}(\boldsymbol{a})(\theta) = f(\boldsymbol{a}(\theta))(\theta) \ .$$

The following proposition discloses some key properties of α-monotonicity:

Proposition 3. *Let $f : E \rightarrow F^\omega$ be α-monotonic and $\boldsymbol{a}, \boldsymbol{b} \in E^\omega$, then:*

$$\boldsymbol{a} \preceq_\alpha \boldsymbol{b} \qquad \textit{implies} \qquad \tilde{f}(\boldsymbol{a}) \preceq_\alpha \tilde{f}(\boldsymbol{b}) \ .$$

In particular, if $f : E \rightarrow F^\omega$ and $g : F \rightarrow G^\omega$ are α-monotonic, then $\tilde{g} \circ f$ is α-monotonic. Furthermore $\widetilde{(\tilde{g} \circ f)} = \tilde{g} \circ \tilde{f}$.

3.3 Relationship between $\preceq_\alpha$ and $\preccurlyeq_\alpha$

As mentioned above, $\preceq_\alpha$ and $\preccurlyeq_\alpha$ are tightly connected. We introduce in this section some useful notations and formalise this link in Proposition 4.

Given an ordered set $(E, \leq)$, an *ideal* is a subset $I \subseteq E$ such that for all $a \in I$ and $b \leq a$, $b \in I$. Its complement in E is $\overline{I}$. Given $a \in E$, the *ideal generated by* a is I_a= $\{b \in E \ : \ b \leq a\}$. Given a sequence $\boldsymbol{a} \in E^\omega$ and an ideal I, set $I[\boldsymbol{a}] = \sup\{\theta + 1 \ : \ \boldsymbol{a}(\theta) \in I\}$[7] and $\boldsymbol{a} \in E^\omega$, set $\overline{I}[\boldsymbol{a}]$= $\inf\{\theta \ : \ \boldsymbol{a}(\theta) \in \overline{I}\}$.

One goes back and forth between $\preceq_\alpha$ and $\preccurlyeq_\alpha$ using Proposition 4:

Proposition 4. *For all $\boldsymbol{a}, \boldsymbol{b} \in E^\omega$, $\boldsymbol{a} \preceq_\alpha \boldsymbol{b}$ iff $\overline{I}[\boldsymbol{a}] \succcurlyeq_\alpha I[\boldsymbol{b}]$ for all ideal $I \subseteq E$.*

3.4 Compatible Mappings

In this section, we capture the semantics of stabilisation monoids via the notion of compatible mappings (see definition below). We establish the existence and unicity of this semantics (Theorem 2).

Definition 2. *Given a stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$, a mapping ρ from M^* (words over M) to M^ω is* compatible with $\mathbf{M}$ *if for some α we have:*

Monotonicity. *ρ is α-monotonic,*
(M^* is ordered by $a_1 \dots a_m \leq b_1 \dots b_n$ if $m = n$ and $a_i \leq b_i$ for all i)
Letter. *for all $a \in M$, $\rho(a) \sim_\alpha a$, and $\rho(\varepsilon) \sim_\alpha 1$,*
(a and 1 denote the constant sequences equal to a and 1 respectively)
Product. *for all $a, b \in M$, $\rho(ab) \sim_\alpha a \cdot b$,*
($a \cdot b$ denotes the constant sequence equal to $a \cdot b$)
Stabilisation. *for all $e \in E(\mathbf{M})$, $m \in \omega$, $\rho(e^m) \sim_\alpha (e^\sharp|_m e)$,*
(e^m denotes the word consisting of m occurrences of the letter e)
(for $a \leq b$ in M, set $(a|_m b) \in M^\omega$ to map $[0, m)$ to a and $[m, \omega)$ to b)
Substitution. *for all $u_1, \dots, u_n \in M^*$, $n \in \omega$, $\rho(u_1 \dots u_n) \sim_\alpha \tilde{\rho}(\rho(u_1) \dots \rho(u_n))$.*
(in which $\rho(u_1) \dots \rho(u_n)$ is naturally seen as an α-non-decreasing sequence of words instead of a word over α-non-decreasing sequences)

Example 2. Consider the stabilisation monoid $\mathbf{M}$ with three elements $\perp \leq a \leq b$, for which the product is defined by $x \cdot y = \min_{\leq}(x, y)$ (hence $b = 1$), and the stabilisation by $b^\sharp = b$ and $a^\sharp = \perp^\sharp = \perp$. Given a word $u \in \{\perp, a, b\}^*$, one sets:

$$\rho(u) = \begin{cases} b & \text{if } u \in b^* \\ \perp|_{|u|_a} a & \text{if } u \in b^*(ab^*)^+ \\ \perp & \text{otherwise.} \end{cases}$$

The mapping ρ is compatible with $\mathbf{M}$:

Monotonicity. We prove *id*-monotonicity. Let $u \leq v$. If $v \in b^+$ then $\rho(u) \leq b = \rho(v)$ (since $\rho(v) = b$ is maximal), i.e., $\rho(u) \preceq_{id} \rho(v)$. If u contains the letter $\perp$ then $\rho(u) = \perp \leq \rho(v)$. Otherwise, since $u \leq v$, u and v contain at least one occurrence of a, and no occurrence of $\perp$. Hence $\rho(u) = (\perp|_{|u|_a} a)$ and $\rho(v) = (\perp|_{|v|_a} a)$. But since $u \leq v$, $|u|_a \geq |v|_a$. We get that $\rho(u) \preceq_{id} \rho(v)$.

[7] The $+1$ makes the theory more smooth, e.g., for Proposition 4.

Letter. We have $\rho(b) = b$, $\rho(\bot) = \bot$ and $\rho(a) = \bot|_1 a$. This implies $\rho(a) \sim_\alpha a$ for $\alpha(\theta) = \max(1, \theta)$.

Product. The only non trivial case is $\rho(aa) = (\bot|_2 a) \sim_\alpha (\bot|_1 a) = a \cdot a$ which holds for $\alpha(\theta) = \max(2, \theta)$.

Stabilisation. Every element is an idempotent. Let $m \geq 1$. For all $x \in M$, $\rho(x^m) = x^\sharp|_m x$ by definition.

Substitution. Let $u_1, \ldots, u_n \in M^*$ and $u = u_1 \ldots u_n$. If for all i, $u_i \in b^*$, then $\rho(u) = b = \tilde{\rho}(\rho(u_1) \ldots \rho(u_n))$. If the letter $\bot$ occurs in some u_i, then $\rho(u) = \bot = \tilde{\rho}(\rho(u_1) \ldots \rho(u_n))$. Otherwise u contains no $\bot$, and at least one occurrence of a. Let K be the set of indices i such that u_i contains an occurrence of a. By applying the definition of ρ and $\tilde{\rho}$ we claim that

$$\tilde{\rho}(\rho(u_1) \ldots \rho(u_n)) = \bot|_{\max(|K|, \max_{i \in K} |u_i|_a)} a \ ,$$

indeed if $\theta < |u_i|_a$ for some i, then $\rho(u_i)(\theta) = \bot$, and as a consequence $\tilde{\rho}(\rho(u_1) \ldots \rho(u_n))(\theta) = \bot$. And otherwise, if $\theta < |K|$, $(\rho(u_1) \ldots \rho(u_n))(\theta)$ contains no occurrences of $\bot$, but $|K|$-many occurrences of a, and hence once more $\tilde{\rho}(\rho(u_1) \ldots \rho(u_n))(\theta) = \bot$. Finally, if $\theta \geq \max(|K|, \max_{i \in K} |u_i|_a)$, then $(\rho(u_1) \ldots \rho(u_n))(\theta)$ contains no occurrences of $\bot$ and at most θ occurrences of a. We obtain $\tilde{\rho}(\rho(u_1) \ldots \rho(u_n))(\theta) = a$. Using Example 1 one has $\max(|K|, \max_{i \in K} |u_i|_a) \approx_\alpha |u|_a$ (for $\alpha(\theta) = \theta^2$), and consequently using Proposition 4, $\rho(u) \sim_\alpha \tilde{\rho}(\rho(u_1) \ldots \rho(u_n))$.

Remark 2. Let us state the link with the standard monoids. Consider a monoid $\mathbf{M} = \langle M, \cdot \rangle$. It can be turned into a stabilisation monoid $\langle M, \cdot, \leq, \sharp \rangle$ by a) setting $\leq$ to be the equality, and b) setting $e^\sharp = e$ for all idempotents e. In this case, it is easy to see that defining for all words $u \in M^*$, $\rho(u)$ to be the sequence constant equal to $\pi(u)$ provides a compatible mapping (for $\alpha = id$, i.e., $\sim_\alpha$ is the equality). According to Theorem 2 below, this is the only possible mapping compatible with $\langle M, \cdot, \leq, \sharp \rangle$.

Theorem 2 states that stabilisation monoids have unique semantics. It is reminiscent of the existence of unique extensions of finite Wilke algebras into ω-semigroups in the theory of regular languages of infinite words.

Theorem 2. *For every stabilisation monoid, there exists a mapping ρ compatible with it. Furthermore it is unique up to $\sim$.*

3.5 Recognisability

We now define the notion of recognisability for cost-functions.

(We use the definitions of Section 3.3.) Given a stabilisation monoid $\mathbf{M} = \langle M, \cdot, \leq, \sharp \rangle$, a length-preserving morphism h from $\mathbb{A}^*$ to M^*, and an ideal $I \subseteq M$, the triple $\mathbf{M}, h, I$ *recognises* the mapping $f : \mathbb{A}^* \to \omega + 1$ if there exists α such that for all $u \in \mathbb{A}^*$, $f(u) \approx_\alpha I[\rho(h(u))]$ in which ρ is a mapping compatible with M. A cost function from $\mathbb{A}^*$ to $\omega + 1$ is *recognisable* if some (equivalently all) function in the class are recognised by some $\mathbf{M}, h, I$.

Example 3. For $\mathbb{A} = \{a, b\}$, the function $|\cdot|_a$ which counts the number of occurrences of a in a word is recognisable. For this, consider the monoid of Example 2, the morphism defined by $h(a) = a, h(b) = b$, and the ideal $I = \{\perp\}$. Then we have $|u|_a = I[\rho(u)]$ for all $u \in \mathbb{A}^*$. This means that $|\cdot|_a$ is recognisable.

As one can expect, recognisability and regularity coincide:

Theorem 3. *A cost function over words is regular iff it is recognisable.*

As mentioned above, this theorem and Theorem 1 are proved at the same time. This proof is much more involved than the equivalent one for regular languages.

We conclude our description by a decidability result.

Theorem 4. *The relation $\preccurlyeq$ is decidable over recognisable cost functions.*

This decidability result extends previous results. For instance, the *boundedness problem* (deciding the existence of $n \in \omega$ such that $f(u) \leq n$ for all words u) corresponds to $f \preccurlyeq 0$. The standard *limitedness problem* (the boundedness over the support of the function) corresponds to $f \preccurlyeq \chi_L$ where L is $\{u \ : \ f(u) < \omega\}$.

4 Conclusion

We have introduced the notion of regular cost functions over words: equivalence classes of functions from words to $\omega + 1$. We have shown that those cost functions enjoy many equivalent representations: algebraic and automata theoretic. This paper is mainly oriented toward the algebraic part, in particular with Theorem 2 which shows that stabilisation monoids have a semantics independent from the automata counterpart. From those equivalences we obtain that the class of regular cost functions enjoy closure under min, max, inf-projection and sup-projection. From those closure properties, it is possible to derive an equivalence with a suitable extension of monadic second-order logic (not presented in the paper). We also provide a decision procedures for the $\preccurlyeq$ relation, and as a consequence the equivalence of cost functions. This result generalises the decidability of the limitedness problem in [9] and [1].

The results were carefully stated so that the extension of the theory to trees is possible (subject of a following paper). In particular we have introduced the notion of history-determinism, a semantic notion which replaces the classical notion of determinism in this framework. Let us finally remark that this whole framework can be extended without any problem to infinite words.

Acknowledgment

I am deeply grateful to Achim Blumensath, Olivier Carton, Pierre-Louis Curien, Denis Kuperberg, Christof Löding, Sylvain Lombardy and Konrad Zdanowski for their comments and helpful discussions on this work.

References

1. Abdulla, P.A., Krcál, P., Yi, W.: R-automata. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 67–81. Springer, Heidelberg (2008)
2. Blumensath, A., Otto, M., Weyer, M.: Boundedness of monadic second-order formulae over finite words. In: ICALP 2009 (2009)
3. Bojańczyk, M., Colcombet, T.: Bounds in omega-regularity. In: LICS 2006, pp. 285–296 (2006)
4. Colcombet, T., Löding, C.: The nesting-depth of disjunctive mu-calculus for tree languages and the limitedness problem. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 416–430. Springer, Heidelberg (2008)
5. Colcombet, T., Löding, C.: The non-deterministic mostowski hierarchy and distance-parity automata. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 398–409. Springer, Heidelberg (2008)
6. Hashiguchi, K.: Limitedness theorem on finite automata with distance functions. J. Comput. Syst. Sci. 24(2), 233–244 (1982)
7. Hashiguchi, K.: Algorithms for determining relative star height and star height. Inf. Comput. 78(2), 124–169 (1988)
8. Hashiguchi, K.: Improved limitedness theorems on finite automata with distance functions. Theor. Comput. Sci. 72(1), 27–38 (1990)
9. Kirsten, D.: Distance desert automata and the star height problem. RAIRO 3(39), 455–509 (2005)
10. Kirsten, D.: Distance desert automata and star height substitutions. Habilitation, Universität Leipzig, Fakultät für Mathematik und Informatik (2006)
11. Leung, H.: Limitedness theorem on finite automata with distance functions: An algebraic proof. Theor. Comput. Sci. 81(1), 137–145 (1991)
12. Leung, H.: The limitedness problem on distance automata: Hashiguchi's method revisited. Theoretical Computer Science 310(1-3), 147 (2004)
13. Simon, I.: Limited subsets of a free monoid. In: FOCS, pp. 143–150 (1978)
14. Simon, I.: Recognizable sets with multiplicities in the tropical semiring. In: Koubek, V., Janiga, L., Chytil, M.P. (eds.) MFCS 1988, vol. 324, pp. 107–120. Springer, Heidelberg (1988)
15. Simon, I.: On semigroups of matrices over the tropical semiring. ITA 28(3-4), 277–294 (1994)
16. Weber, A.: Distance automata having large finite distance or finite ambiguity. Mathematical Systems Theory 26(2), 169–185 (1993)
17. Weber, A.: Finite-valued distance automata. Theor. Comput. Sci. 134(1), 225–251 (1994)

A Tight Lower Bound for Determinization of Transition Labeled Büchi Automata

Thomas Colcombet and Konrad Zdanowski[*]

LIAFA/CNRS/Université Paris 7, Denis Diderot, Paris, France

Abstract. In this paper we establish a lower bound hist(n) for the problem of translating a Büchi word automaton of size n into a deterministic Rabin word automaton when both the Büchi and the Rabin condition label transitions rather than states. This lower bound exactly matches the known upper bound to this problem. The function hist(n) is in $\Omega((1.64n)^n)$ and in $o((1.65n)^n)$.

Our result entails a lower bound of hist($n-1$) when the input Büchi automaton has its Büchi acceptance condition labeling states (as it is usual). Those lower bounds remain when the output deterministic Rabin automaton has its Rabin acceptance condition labeling states.

1 Introduction

Since the seminal work of Büchi [Büc62], automata running over infinite words (of length ω) have become a fundamental object in automata theory. The automata introduced by Büchi are non-deterministic and use a so called *Büchi acceptance condition*: a run of the automaton is accepting if some set of states is visited infinitely often. Büchi established that it is possible to complement them, though those automata cannot be made deterministic in general. The next fundamental step in this theory is the result of McNaughton [McN66] stating that Büchi automata can be effectively transformed into deterministic automata (of doubly exponential size) using a *Muller acceptance condition*; i.e., a run is accepting if it satisfies a boolean combination of atomic properties of the form 'the state q is visited infinitely often'[1]. Those two operations of complementation and determinization of automata running on infinite words are the two key results in this theory, and a long line of research is dedicated to the search of the exact complexity of those two operations (in terms of state blow-up).

The complementation saga. The quest for the exact complexity for the complementation of Büchi automata has been the subject of a long list of works (see e.g., [Sch09a]). Now both lower bounds and upper bounds are well known and tightly related. Indeed, there is a function *tight* which is in $O((0.76n)^n)$ such that (lower bound) for all n, there exists a Büchi automaton of size n such

[*] Both authors are supported by the ANR project JADE ("Jeux et Automates, Décidabilité et Extensions").

[1] The original statement is that every Büchi automaton is equivalent to a boolean combination of deterministic Büchi automata.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 151–162, 2009.

that every Büchi automaton for the complement language has $\Omega(\mathit{tight}(n-1))$ states [Yan08], and (upper bound) for all Büchi automaton of size n, there exists a Büchi automaton for its complement of size $O(\mathit{tight}(n+1))$ [Sch09a]. Since $O(n^2\mathit{tight}(n-1)) = O(\mathit{tight}(n+1))$, lower and upper bound to the complementation problem only differ by a quadratic factor. This is the end of the quest.

The determinization problem: The origins. In the theory of finite automata, determinization is easy to describe [RS64]. Given a non-deterministic (finite word) automaton of size n, one constructs a deterministic automaton that maintains the set of states that the original automaton could have reached at the current position. The resulting automaton has size 2^n (2^n-1 if one does not require completeness). This construction is known to be optimal (it is even optimal for complementation). For automata running over infinite words, the information maintained by the above construction is not sufficient. Safra was the first to provide a solution oriented toward efficiency [Saf88]. In his construction the reachable states are maintained, and furthermore organized in a tree-structure (called a Safra tree) which carries information on the history of the possible runs reaching each state. The resulting construction takes as input a non-deterministic Büchi automaton of size n, and produces a deterministic automaton of size at most $(12)^n n^{2n}$ states using *a Rabin acceptance condition*; i.e., a run is accepting if it satisfies a disjunction of properties of the form 'the set of state E is visited infinitely often, and the set of states F finitely often' (each such pair (E,F) is named a *Rabin pair*). The automaton constructed by Safra uses $2n$ Rabin pairs: this is achieved by furthermore maintaining in each Safra tree a name attached to each node. This naming mechanism allows a dynamic reuse of the Rabin pairs. The ideas developed by Safra in his seminal work are underlying most of the other solutions to the determinization problem.

The determinization problem: Further developments. Piterman [Pit07] proposes a modification to Safra's constructions for producing an automaton using a *parity acceptance condition*; i.e., every state has a priority in ω, and a run is accepting if the least priority seen infinitely often is even. This construction, not only produces a parity condition (which is a further restriction on the Rabin condition), but also improves on the original upper bound of Safra, reaching a deterministic automaton of size at most $2nn^n n!$. By a finer analysis of Piterman's solution, Liu and Wang reach an upper bound of $2n(n!)^2$ [LW]. Independently, Schewe in [Sch09b] gives a solution in $o((2.66n)^n)$ for the size of a Rabin deterministic automaton using 2^{n-1} Rabin pairs[2]. In the same paper, Schewe provides the best known upper bound for producing a deterministic parity automaton in $O((n!)^2)$ (recall that $n! \approx (0.36n)^n$).

Extensions and Variants. Safra has generalized his construction: he shows how to directly determinize automata that use *Streett acceptance condition*; i.e., the dual of Rabin: a run is accepting if it satisfies a conjunction of properties of the

[2] In fact one can argue on the respective advantages of this solution with respect to others since it is obtained by completely removing the naming system in Safra's original solution, and the price to pay for that is the 2^{n-1} number of Rabin pairs.

form 'if the set of states E is visited infinitely often, then the set of states F is visited infinitely often'. A construction with similar characteristics has been described by Muller and Schupp using different techniques [MS95]. The latter has been revisited by Kähler and Wilke for giving a unified view on complementation, determinization, and disambiguisation[3] of Büchi automata[KW08]. Finally, Piterman in [Pit07] describes the best known determinization procedure that takes a Streett automaton as input, and produces a deterministic parity automaton as output.

Lower bounds. The first non-trivial lower bound to the problem of determinization was $n!$ [Löd99], derived from an early lower bound for complementation [Mic88]. The best known lower bound is in $\Omega((0.76n)^n)$ [Yan06], and holds even if the resulting automaton uses a Muller acceptance condition.

Decomposition. Schewe [Sch09b] develops the idea that the correct way of describing determinization constructions consists in first isolating what he calls the *principle acceptance mechanism* that captures the core of the construction, and then waving on this construction for producing deterministic automata of various characteristics. The *principle acceptance mechanism* takes the form of a deterministic Rabin-automaton which has the non-standard characteristic that the Rabin condition labels transitions rather than states. It has 2^{n-1} Rabin pairs and $hist(n) = o((1.65n)^n)$ states, where $hist(x)$ is the function giving the number of states of an output automaton in Schewe's construction.

Contributions. In this paper, we establish that the principle acceptance mechanism isolated by Schewe is optimal. More precisely, we prove that there exists a transition-labeled Büchi automaton of size n accepting a language L_n such that every transition-labeled deterministic Rabin automaton accepting the language L_n has at least $hist(n)$ states. If the input Büchi automaton has its Büchi condition labeling states, then we have a lower bound of $hist(n-1)$. Since transition-labeled automata are at least as compact as their state-labeled counterparts, those lower bounds can be directly transferred to state-labeled Rabin-automata as output. Since both $hist(n)$ and $hist(n-1)$ are in $\Omega((1.64n)^n)$, this improves the previous lower bound of $\Omega((0.76n)^n)$ to $\Omega((1.64n)^n)$.

2 Basic Definitions and Facts

2.1 Basic Notions and Notations

For a sequence u, the i-th element of u is denoted by $u(i)$ with the convention that elements in a sequence are labeled from 0. The empty sequence is denoted by ε and the length of a sequence u is denoted by $|u|$. If we assume an ordering between elements of sequences then the *lexicographic ordering* of sequences is defined as $u <_{\text{lex}} v$ if there exists i such that $u(i) < v(i)$ and for all $j < i$, $u(j) = v(j)$ or $|u| < |v|$ and for all $i < |u|$, $u(i) = v(i)$. Then $u \leq_{\text{lex}} v$ if $u <_{\text{lex}} v$ or $u = v$.

[3] An automaton is non-ambiguous if there is at most one accepting run per input. This is weaker than determinism.

A *tree* T is a subset of $\omega^{<\omega}$ such that if $xi \in T$ then $x \in T$ and $xj \in T$ for all $j < i$. The elements of T are called *nodes*. The *size* of T is the number of nodes in T. A node of the form xi is called a *child* of the node x. If x is a prefix of y we write $x \preceq y$, and we write $x \prec y$ if $x \preceq y$ and $x \neq y$. When $x \preceq y$, x is called an *ancestor of* y, and y a *descendant of* x. When $x \prec y$, x is called a *strict ancestor of* y, and y a *strict descendant of* x.

2.2 Automata

We define here automata running on infinite inputs. Le us emphasize the fact that, in this work, the accepting condition refers to transitions rather than states.

A *(finite) automaton* is a tuple $\mathcal{A} = (Q, \Sigma, I, \Gamma, \Delta)$, where Q is a set of *states*, Σ is an *input alphabet*, I is a set of *initial states*, Γ is an *output alphabet* and $\Delta \subseteq Q \times \Sigma \times \Gamma \times Q$ is the *transition relation*.

We see a finite automaton $\mathcal{A}$ as a non-deterministic transducer from Σ^ω to Γ^ω. For a word $u \in \Sigma^\omega$, $\rho = (p_0, b_0, p_1)(p_1, b_1, p_2)(p_2, b_3, p_3)\ldots$ is a *run of* $\mathcal{A}$ *over* u if $p_0 \in I$ and $\forall i\, (p_i, u(i), b_i, p_{i+1}) \in \Delta$. The *output of* ρ, $\mathrm{Out}(\rho)$ is the word $b_0 b_1 b_2 \ldots$.

The automaton $\mathcal{A}$ is called *deterministic* if $\mathcal{A}$ has exactly one initial state and Δ is a functional relation, that is for each $q \in Q$ and $a \in \Sigma$ there is at most one transition of the form (q, a, b, p) in Δ. We will take the convention that when an automaton is deterministic, we denote its transition relation by δ, and we use it also as a mapping from $Q \times \Sigma$ to Q. This mapping is naturally extended into a mapping from $Q \times \Sigma^*$ to Q in the usual manner. Thus, $\delta(q, u)$ is the unique state p such that after reading u when starting in q the automaton $\mathcal{A}$ is in the state p.

An automaton $\mathcal{B}$ is a *Büchi automaton* if $\Gamma = \{0, 1\}$. The language L_B is the set of words v in $\{0, 1\}^\omega$ such that v contains infinitely many zeros. A run ρ of of $\mathcal{B}$ is *accepting* if $\mathrm{Out}(\rho) \in L_B$. An automaton $\mathcal{B}$ *accepts* a word $u \in \Sigma^\omega$ if there is an accepting run of $\mathcal{B}$ on u.

An automaton $\mathcal{R}$ is a *Rabin automaton* with h conditions (i.e, h Rabin pairs) for $h \in \omega$, if $\Gamma = \mathcal{P}(\{r_1, s_1, r_2, s_2, \ldots, r_h, s_h\})$. The Rabin language L_R is the set of words v in Γ^ω such that for some i, $r_i \in v(n)$ for finitely many n, and $s_i \in v(n)$ for infinitely many n. As above, a run ρ of a Rabin automaton $\mathcal{R}$ is *accepting* if $\mathrm{Out}(\rho) \in L_R$ and $\mathcal{R}$ *accepts* u if there is an accepting run of $\mathcal{R}$ on u.

We are interested in providing a lower bound to the number of states needed for the determinization of a Büchi automaton. Hence, we define the *size* of an automaton as the number of its states. For a Büchi automaton $\mathcal{B}$, let $D(\mathcal{B})$ be the size of a smallest deterministic Rabin automaton which recognizes the same language. Let $D(n)$= $\max\{D(\mathcal{B}) : \mathcal{B}$ is a Büchi automaton of size $n\}$.

An important remark made by Yan in [Yan08] is that for each n there exists a canonical Büchi automaton with n states which can simulate every Büchi automaton of this size: the full automaton. In our context, it gives the following definition.

Definition 1 (Full automaton). *The full Büchi automaton of size n is the automaton $\mathcal{B}_n = (Q, \Sigma, Q, \{0, 1\}, \Delta)$ such that $Q = \{1, \ldots, n\}$ and $\Sigma = \mathcal{P}(Q \times \{0, 1\} \times Q)$ and $\Delta = \{(q, a, b, p) \colon (q, b, p) \in a\}$.*

Thus, the language of $\mathcal{B}_n$ is a set of words u such that there is a path $\rho = (p_0, b_0, p_1)(p_1, b_1, p_2)(p_2, b_2, p_3)\dots$ such that for each i, $(p_i, b_i, p_{i+1}) \in u(i)$ and for infinitely many i, $b_i = 0$. We denote this language L_n.

Since full automata can simulate every automaton of the same size, we naturally get the following lemma. It follows from it that to prove lower or upper bounds on $D(n)$ it is enough to consider only full automata $\mathcal{B}_n$.

Lemma 1 ([Yan06]). *$D(n) = D(\mathcal{B}_n)$.*

2.3 Games

A *game* is a tuple $\mathcal{G} = (V, V_E, V_A, p_I, \mathit{\Gamma}, \mathrm{Move}, L)$, where V is a set of *positions* which is partitioned into the *positions for Eva V_E* and the *positions for Adam V_A*, $p_I \in V$ is the *initial position* of $\mathcal{G}$, $\mathit{\Gamma}$ is the *labeling alphabet*, $\mathrm{Move} \subseteq V \times \mathit{\Gamma} \times V$ is the set of possible *moves*, and $L \subseteq \mathit{\Gamma}^\omega$ is the *winning condition.* A tuple $(v, a, w) \in \mathrm{Move}$ indicates that there is a move from v to w which produces letter a. A game using the winning condition L is called an *L-game.*

During a play of the game $\mathcal{G}$, the two players, Adam and Eva, make moves according to Move, the player to whom belong the current positions choosing the next move. Formally, a *play* is a maximal sequence $\pi = (p_0, a_0, p_1, a_1, p_2, a_2, \dots)$ such that $p_0 = p_I$ and for each i, $(p_i, a_i, p_{i+1}) \in \mathrm{Move}$. Let $\pi_{\mathit{\Gamma}} = (a_0, a_1, a_2, \dots)$. Eva *wins the play* π if $\pi_{\mathit{\Gamma}} \in L$. Otherwise, Adam wins the play.

A strategy for the player X is a function which tells the player what moves he should choose depending on the finite history of moves played so far. Formally, a *strategy* (for Eva or for Adam) is a total mapping from finite sequences $(p_0, a_0, \dots, a_{n-1}, p_n)$ into Move. A play π is compatible with a strategy σ for Eva (resp. Adam) if for all prefixes $(p_0, \dots, p_{n-1}, a_{n-1}, p_n)$ of π, if $p_{n-1} \in V_E$ (resp. in V_A), then $\sigma(p_0, \dots, p_{n-1}) = (p_{n-1}, a_{n-1}, p_n)$. A strategy σ for Eva (resp. Adam) *is winning* if Eva (resp. Adam) wins every play compatible with σ.

A *strategy with memory m for Eva* is described as $(M, \mathrm{update}, \mathrm{choice}, \mathrm{init})$ in which M is a set of size m called the *memory*, update is a mapping from $M \times \mathrm{Move}$ to M, choice is a mapping from $V_E \times M$ to Move, and init $\in M$. The mapping update is defined for moves, but can naturally be extended to paths in the game. The strategy described by $(M, \mathrm{update}, \mathrm{choice}, \mathrm{init})$ is the one which to each path $\pi = (p_0, a_0, \dots, a_{n-1}, p_n)$ with $p_n \in V_E$ associates $\mathrm{choice}(p_n, \mathrm{update}(\mathrm{init}, \pi))$. A player X wins a game with memory n if it has a winning strategy with memory m. A *positional strategy* corresponds to the case $m = 1$.

We call a game $\mathcal{G}$ a *Rabin-game* if L is the Rabin language L_R over some alphabet $\mathit{\Gamma}$. Eva has positional winning strategies in Rabin games:

Theorem 1 ([Kla94],[Zie98]). *For every Rabin-game, if Eva wins, she can win using a positional strategy.*

2.4 Reduction

A standard way to use a deterministic automaton is for game reduction. Indeed, given a deterministic automaton for a language L with (e.g. Rabin) acceptance condition F with n states, and an L-game, one can perform the product of the game with the automaton, yielding an F-game. From the determinism of the automaton, one can derive that the winner of the two games is the same. Pushing further, if the F-game admits positional strategies for Eva (and it is the case for Rabin-games according to Theorem 1), one can see the states of the automaton as maintaining the memory for a strategy in the L-game: Eva needs memory n in the original game. We obtain:

Lemma 2. *If Eva wins an L-game, and there exists a deterministic Rabin-automaton for L with n states, then Eva wins using a strategy with memory n.*

It is standard to use this result for proving upper bound results on the memory needed for the winner of a game. In this work, we take the opposite point of view and read the above lemma as follows: *"If Eva wins an L-game, and requires memory n for that, then every deterministic Rabin-automaton for L has size at least n"*. This provides an argument for proving lower bounds on determinization problems.

3 The Determinization Construction

Now, we will describe briefly the construction of Schewe from [Sch09b]. This construction takes as input a Büchi automaton $\mathcal{B}$, and produces as output a deterministic Rabin automaton $\mathcal{R}$ such that $L(\mathcal{B}) = L(\mathcal{R})$. This construction captures the core mechanism of the famous construction of Safra [Saf88] (it removes from it the node naming system that was used for reducing the number of Rabin pairs). For more explanation of the construction we use here, we refer the reader to [Sch09b].[4] For a more extensive description of the original Safra construction, we suggest [Tho97].

Let us fix a Büchi automaton $\mathcal{B} = (Q, I, \Sigma, \{0,1\}, \Delta)$ with n states. For a set $S \subseteq Q$ and a letter $a \in \Sigma$, let $\Delta(S,a) = \{q : \exists p \in S \exists b\,(p,a,b,q) \in \Delta\}$ and $\Delta_0(S,a) = \{q : \exists p \in S\,(p,a,0,q) \in \Delta\}$.

In the construction we work with trees with nodes labeled by subsets of states of a Büchi automaton. A *labeling* of T is a function from nodes of T to the set of labels (in our case: subsets of Q). A label of a node $x \in T$ is denoted by $T(x)$. We use a convention that for $x \notin T$, $T(x) = \emptyset$. It does not cause any ambiguity because the labels of nodes of T will be always nonempty. For a node x we order its children by the "*older than*" relation and we say that xi is older than xj for $i < j$.

Recall that siblings in a tree have, by definition, consecutive numbers starting from 0. For this reason, the process of deleting a node requires some explanations. *Deleting* a node x in a tree T consists in a) removing the node x and all its descendants, and b) shifting all the younger siblings of x to the left, i.e., the direct right sibling of x takes the place of x, etc...

[4] Schewe's construction takes a state-labeled Büchi automaton as input rather than a transition labeled automaton. Nevertheless, his approach easily adapts to this case.

Now, we define the deterministic Rabin automaton $\mathcal{R}$ accepting $L(\mathcal{B})$. The set of states of $\mathcal{R}$ is the set of trees T of size at most $|Q|$ labeled with subsets of Q and such that

1. each node in T is labeled by a nonempty set of states,
2. the label of a node x is a strict superset of the union of its children labels,
3. labels of siblings are disjoint.

We call such trees, after [Sch09b], *history trees*. According to our notation for $x \in T$, $T(x)$ is the label of the node x in T. In particular, by the second point above, $T(\varepsilon)$ denotes the set of states which occur in some label of T. We denote by $\mathcal{H}$ the set of history trees.

The initial state of $\mathcal{R}$ is a one node tree labeled by the set of initial states of $\mathcal{B}$. We always call this initial tree T_0 (in general T_0 depends on the set of initial states of $\mathcal{B}$ but in the cases we consider it will be just a one node tree labeled with Q). Let T be the current state of $\mathcal{R}$, and a be the currently read letter. The transition function is defined in multiple steps as follows.

1. For each node $x \in T$, replace the label of x with $\Delta(T(x), a)$.
2. For each node $x \in T$ originally labeled by S, if $\Delta_0(S, a)$ is nonempty then form a new youngest child of x and label it with $\Delta_0(S, a)$.
3. For each node x and for each state $q \in T(x)$, if q belongs to an older sibling of x, then delete q from labels of x and all its descendants.
4. Now, we contract the tree obtained so far:
   4.1 for each node x such that its label is nonempty and is equal to the sum of labels of its descendants, delete all strict descendants of x, and call x *green*,
   4.2 for each node xi that has an empty label, mark xi and all the nodes in trees rooted at younger siblings of xi as red, and delete the subtree rooted in xi.
5. The output of the transition is a set $\mathcal{E}$ of what we call *events*. For each red node x we put $(x, A) \in \mathcal{E}$ and for each green x we put (x, E) in $\mathcal{E}$.

The events (x, A) play the role of r_i in the definition of a Rabin language L_R and events (x, E) correspond to s_i. Thus, the Rabin condition simply states that there exists x such that (x, A) is seen finitely many times and (x, E) infinitely many times. In other words, there will be a node x such that from some time on it will never be deleted or moved to the left during point 4.2 of the transition and it will be green infinitely many times. We will denote by Λ the set of possible *events*, i.e., pairs of the form either (x, A) or (x, E) in which x is a possible node in an history tree (there are 2^{n-1}-many such x).

Since the Rabin automaton defined above is deterministic, its transition relation can be seen as a partial function from $\mathcal{H} \times \Sigma$ to $\mathcal{P}(\Lambda) \times \mathcal{H}$. Given a history tree T, and a letter a, $\delta(T, a)$ denotes the tree obtained by the above procedure, and $\mathcal{E}(T, a)$ the set of produced events. The mapping δ is extended into a mapping from $\mathcal{H} \times \Sigma^*$ to $\mathcal{H}$ by $\delta(T, \varepsilon) = T$, and $\delta(T, ua) = \delta(\delta(T, u), a)$. The mapping $\mathcal{E}$ is extended into a mapping from $\mathcal{H} \times \Sigma^*$ to $\mathcal{P}(\Lambda)$ by $\mathcal{E}(T, \varepsilon) = \emptyset$ and $\mathcal{E}(T, ua) = \mathcal{E}(T, u) \cup \mathcal{E}(\delta(T, u), a)$.

Theorem 2 ([Sch09b], variant of [Saf88]). *Let $\mathcal{B}$ be a nondeterministic Büchi automaton and let $\mathcal{R}$ be a deterministic Rabin automaton constructed as above for $\mathcal{B}$. Then, $L(\mathcal{B}) = L(\mathcal{R})$.*

3.1 Complexity of the Construction

The size complexity of the construction described above is measured as the number of states of the constructed Rabin automaton. Then, let us define the function *hist*(n) as the number of history trees for Q of size n. The subject of the paper is to prove that this value also provides a lower bound to the determinization construction. This function is analyzed in [Sch09b] where it is proved to be in $o((1.65n)^n)$. On the other hand, the following lower bound on *hist*(n) can be proved.

Lemma 3 (M. Bouvel, D. Rossin). *The function hist(n) is in $\Omega((1.64n)^n)$.*

4 Optimality of Determinization

During this section we fix a full automaton of size n, $\mathcal{B}_n = (Q, \Sigma, Q, \{0,1\}, \Delta)$, and we set Σ to be its alphabet $\mathcal{P}(Q \times \{0,1\} \times Q)$. The subject of this section is to prove Theorem 4 that establishes our lower bound.

The proof consists first in restricting ourselves to a constant set of reachable states, second to prove a lower bound in this restricted context, third to reconstruct our main result.

4.1 Fixing the Set of Reachable States

We define the set of states reachable by a word u, $\mathrm{Reach}(u)$, by induction. $\mathrm{Reach}(\varepsilon) = Q$ and $\mathrm{Reach}(va) = \{q : p \in \mathrm{Reach}(v),\ (p,b,q) \in a\}$ for $v \in \Sigma^*$ and $a \in \Sigma$. Let Σ_S be the set of letters $a \in \Sigma$ such that $S = \{q : (p,b,q) \in a,\ p \in S\}$, or equivalently $\mathrm{Reach}(a) = \mathrm{Reach}(aa) = S$. Let L_n^S be $L_n \cap \Sigma_S^\omega$.

Of course, each history tree T maintains in $T(\varepsilon)$ the set of states reachable by the original automaton at the current position in the word. Hence it is natural for all $S \subseteq Q$ to consider $\mathcal{H}_S = \{T \in \mathcal{H} : T(\varepsilon) = S\}$. We have for all words $u \in \Sigma_S^*$ and all $T \in \mathcal{H}_S$ that $\delta(T,u) \in \mathcal{H}_S$.

4.2 The Game

Let us fix ourselves a set $S \subseteq Q$. We establish in this section Theorem 3 which provides a lower bound for the size of a deterministic Rabin automaton accepting L_n^S. For this, we define a game $\mathcal{G}$ such that Eva wins $\mathcal{G}$ but she cannot win with memory less than $|\mathcal{H}_S|$. That proves, by Lemma 2, that any deterministic Rabin automaton accepting L_n^S has at least $|\mathcal{H}_S|$ states.

We order history trees: we say that T is *strictly smaller* than T' at position x, written $T <_x T'$, if $T'(x) \subsetneq T(x)$ and for all $y <_{\mathrm{lex}} x$, $T'(y) = T(y)$. We can remark that if $T <_x T'$, x may not be a node of T' (i.e., $T'(x) = \emptyset$), but in any case it is a node of T.

We construct the L_n^S-game $\mathcal{G}= (V, V_E, V_A, p_I, \Sigma_S^+, \mathrm{Move}, L_n^S)$, where V_E is a singleton set $\{p_E\}$ and V_A consists of the initial position in the game p_I and one position p_T for each history tree $T \in \mathcal{H}_S$. The moves in $\mathcal{G}$ are the following:

1. $(p_I, u, p_E) \in$ Move, for all $u \in \Sigma_S^+$,
2. $(p_E, \mathrm{id}_S, p_T) \in$ Move, for each history tree T, where $\mathrm{id}_S = \{(q, 1, q) : q \in S\}$,
3. $(p_T, u, p_E) \in$ Move, for each history tree T and word $u \in \Sigma_S^+$ if there exists a node x in $\delta(T, u)$ such that either
   - $(x, E) \in \mathcal{E}(T, u)$, and for all $y \leq_{\mathrm{lex}} x$, $(y, A) \notin \mathcal{E}(T, u)$ and $\delta(T, u)(y) = T(y)$ or;
   - $\delta(T, u) <_x T$ and for all $y <_{\mathrm{lex}} x$, $(y, A) \notin \mathcal{E}(T, u)$.

Essentially, this game has a flower shape. The central node is controlled by Eva, and from this node, she can decide to go to any of the petals of the flower, each one corresponding to a history tree p_T. Then, it is Adam's turn to choose a word u, and come back to the center. Adam's moves are restricted in the following way: playing a word u in petal p_T is valid if it is possible to witness in the behaviour of $\mathcal{R}$ from the state T when reading u that it is profitable for Eva. This witness takes the form of a position x in the tree T such that nothing happened above or to the left of x, and something good for Eva happened in x: either some Eva-good event in $\mathcal{E}$, or a local advance of the $<_x$ ordering.

It should be clear that both players can always perform a move from their positions. Thus, every completed play has the length ω.

Lemma 4. *Eva has a winning strategy in $\mathcal{G}$.*

To prove Lemma 4 we show that a winning strategy for Eva is as follows: if a word u was played after a finite play and Eva is to make a move from p_E, then she chooses to go to a position indexed by $\delta(T_0, u)$. Then, possible moves for Adam forces him to generate infinitely often an event (x, E) without generating infinitely many (x, A).

4.3 Memory Lower Bound for the Game

Now, for each history tree $T \in \mathcal{H}_S$ we define a game $\mathcal{G}_T$ which is a modification of $\mathcal{G}$ by removing a petal p_T, i.e., removing one of Adam's positions, and hence removing to Eva the ability to take the corresponding move. All other elements in the game are unchanged. Our crucial Lemma 5 states that for any two history trees $T \neq T'$ in $\mathcal{H}_S$, there exists always a word u such that $(p_{T'}, u, p_E)$ is a valid move in the game, and such that nothing good for Eva happens when $\mathcal{R}$ reads u from state T. This is formalized as follows:

Lemma 5. *Let $T \neq T'$ be history trees in $\mathcal{H}_S$. There exists a word u such that*

1. *$(p_{T'}, u, p_E)$ is a move in $\mathcal{G}_T$,*
2. *$T = \delta(T, u)$,*
3. *for all x, $(x, E) \notin \mathcal{E}(T, u)$.*

This lemma is the technical core of our proof. It requires an analysis of the differences between the two trees. This results in many situations of very different nature. With the help of Lemma 5 we can prove the following.

Lemma 6. *For every $T \in \mathcal{H}_S$ Adam has a winning strategy in $\mathcal{G}_T$.*

A winning strategy for Adam is as follows. From p_I he plays a word u such that $\delta(T_0, u) = T$, where T_0 is the initial state of the deterministic Rabin automaton $\mathcal{R}$ from Theorem 2. The good answer for Eva would be to move to p_T (according to the proof of Lemma 4), but since the petal is removed, she has to choose another move, say to $p_{T'}$ for some $T' \neq T$. Then Adam answers according to Lemma 5. Of course, using this strategy, Adam maintains the property that if a word w has been played so far, then $\delta(T_0, w) = T$, and hence Adam can always answer to Eva's proposal using Lemma 5. Now, when the play gets infinite, say producing an infinite word u, we can see from the property of the words u in Lemma 5 that $\mathcal{R}$ does not accept this word (no events good for Eva are ever produced). Hence, $u \notin L_n^S$ and Adam wins.

From Lemma 6 we may easily infer the following.

Corollary 1. *Eva has no winning strategy with memory $|\mathcal{H}_S| - 1$ in $\mathcal{G}$.*

Indeed, if Eva had a strategy with memory $|\mathcal{H}_S| - 1$, then there would be a position p_T which is never visited by this strategy. But this would mean that Eva wins $\mathcal{G}_T$ with the exact same strategy.

Using Lemma 2, we now get:

Theorem 3. *Every deterministic Rabin automaton accepting L_n^S has size at least $|\mathcal{H}_S|$.*

4.4 Reduction to the General Case

What remains to be done is a reduction to the general case. We do this by decomposing a given deterministic Rabin automaton accepting L_n into disjoint sets of states. The following lemma gives an argument for such a decomposition.

Lemma 7. *Let $\mathcal{R}$ be a deterministic Rabin automaton which accepts L_n with a transition function δ and an initial state q_0. If $\delta(q_0, u) = \delta(q_0, v)$ then* $\mathrm{Reach}(u) = \mathrm{Reach}(v)$.

Now, it is possible to prove the main theorem.

Theorem 4. *Every deterministic Rabin automaton accepting L_n has size at least $hist(n)$.*

The theorem is proved by defining for a given deterministic Rabin automaton $\mathcal{R}$ accepting L_n a partition of its states. For S a nonempty subset of Q, let R_S be the set of states $q \in Q$ such that there exists u with $\mathrm{Reach}(u) = S$ and $\delta(q_0, u) = q$, in which q_0 is the initial state of $\mathcal{R}$ and δ is its transition function. The automaton $\mathcal{R}$ restricted to R_S can be seen as a Rabin automaton which accepts the restriction of L_n to letters in Σ_S. Hence the lower bound of Theorem 3 holds for the automaton $\mathcal{R}$ restricted to R_S.

Theorem 4 then follows from the fact that the sets R_S are disjoint, and that

$$\mathit{hist}(n) = |\mathcal{H}| = \sum_{S \subseteq \{1,\dots,n\}} |\mathcal{H}_S| \ .$$

5 Discussion

We have proved an exact lower bound for the determinization of Büchi automata using a non-standard definition of automata in which the acceptance condition labels transitions rather than states as it is usual.

State-labeled Büchi automaton as input. A *state-labeled Büchi automaton* over alphabet Σ and of states Q, has transitions $\Delta \subseteq Q \times \Sigma \times Q$ (the Büchi label has disappeared from transitions), and a set of final states $F \subseteq Q$. A run of the automaton is accepting if it visits a state in F infinitely often. Given a (transition-labeled) Büchi automaton with n states $\mathcal{A}$, one can transform it into a state-labeled Büchi automaton with $2n$ states $\mathcal{B}$ in such a way that the action of every letter in $\mathcal{A}$ can be simulated by a sequence of two letters on $\mathcal{B}$. This is sufficient for proving that translating a state-labeled automaton of size $2n$ into a deterministic Rabin-automaton requires at least $\mathit{hist}(n)$ states.

However it is possible to do much better if one inspects the proof of our theorem more closely. For this one remarks that in our proof of optimality one does not need to use the whole alphabet Σ. By a close inspection of all the cases in the proof of Lemma 5, one can remark that all the letters appearing in the word u described by this lemma do satisfy the following property: either there is no 0-transitions in a letter, or all transitions labeled by 0 have the same state as origin. Using this remark, one can optimise the above argument and get the following lower bound.

Theorem 5. *There is a language L'_n accepted by a state-labeled Büchi automaton with n-states such that every deterministic Rabin automaton accepting L'_n has size at least $\mathit{hist}(n-1)$.*

State-labeled Rabin automaton as output. It is very simple to transform a state-labeled automaton accepting some language (whatever is its acceptance condition) into a transition-labeled one using the same acceptance condition and the same number of states. The idea is simply to use transitions that have as label the label of the origin state in the original automaton. From this we deduce that all the lower bounds presented above can be directly transferred to transition-labeled automata as output.

Acknowledgements

We are grateful to Mathilde Bouvel and Dominique Rossin for their proof of Lemma 3. We also thank Christof Löding for his insightful comments.

References

[Büc62] Richard Büchi, J.: On a decision method in restricted second-order arithmetic, pp. 1–11 (1962)

[Kla94] Klarlund, N.: Progress measures, immediate determinacy, and a subset construction for tree automata. Annals of Pure and Applied Logic 69(2–3), 243–268 (1994)

[KW08] Kähler, D., Wilke, T.: Complementation, disambiguation, and determinization of Büchi automata unified. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 724–735. Springer, Heidelberg (2008)

[Löd99] Löding, C.: Optimal bounds for transformations of ω-automata. In: Pandu Rangan, C., Raman, V., Ramanujam, R. (eds.) FSTTCS 1999. LNCS, vol. 1738, pp. 97–109. Springer, Heidelberg (1999)

[LW] Liu, W., Wang, J.: A tighter analysis of Piterman's Büchi determinization (submitted)

[McN66] McNaughton, R.: Testing and generating infinite sequences by a finite automaton. Information and Control 9, 521–530 (1966)

[Mic88] Michel, M.: Complementation is more difficult with automata on infinite words. In: CNET, Paris (1988)

[MS95] Muller, D.E., Schupp, P.E.: Simulating alternating tree automata by nondeterministic automata: New results and new proofs of the theorems of Rabin, McNaughton and Safra. Theor. Comput. Sci. 141(1&2), 69–107 (1995)

[Pit07] Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. Logical Methods in Computer Science 3(3) (2007)

[RS64] Rabin, M.O., Scott, D.: Finite automata and their decision problems. Technical report (1964)

[Saf88] Safra, S.: On the complexity of ω-automata. In: FOCS, pp. 319–327 (1988)

[Sch09a] Schewe, S.: Büchi complementation made tight. In: STACS 2009 (2009)

[Sch09b] Schewe, S.: Tighter bounds for the determinisation of Büchi automata. In: de Alfaro, L. (ed.) FoSSaCS 2009. LNCS, vol. 5504, pp. 167–181. Springer, Heidelberg (2009)

[Tho97] Thomas, W.: Languages, automata and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 3. Springer, Heidelberg (1997)

[Yan06] Yan, Q.Q.: Lower bounds for complementation of ω-automata via the full automata technique. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 589–600. Springer, Heidelberg (2006)

[Yan08] Yan, Q.Q.: Lower bounds for complementation of ω-automata via the full automata technique. Logical Methods in Computer Science 4 (2008)

[Zie98] Zielonka, W.: Infinite games on finitely coloured graphs with applications to automata on infinite trees. Theoretical Computer Science 200(1–2), 135–183 (1998)

On Constructor Rewrite Systems and the Lambda-Calculus[*]

Ugo Dal Lago and Simone Martini

Dipartimento di Scienze dell'Informazione, Università di Bologna
Mura Anteo Zamboni 7, 40127 Bologna, Italy
{dallago,martini}@cs.unibo.it

Abstract. We prove that orthogonal constructor term rewrite systems and lambda-calculus with weak (i.e., no reduction is allowed under the scope of a lambda-abstraction) call-by-value reduction can simulate each other with a linear overhead. In particular, weak call-by-value beta-reduction can be simulated by an orthogonal constructor term rewrite system in the same number of reduction steps. Conversely, each reduction in an orthogonal term rewrite system can be simulated by a constant number of weak call-by-value beta-reduction steps. This is relevant to implicit computational complexity, because the number of beta steps to normal form is polynomially related to the actual cost (that is, as performed on a Turing machine) of normalization, under weak call-by-value reduction. Orthogonal constructor term rewrite systems and lambda-calculus are thus both polynomially related to Turing machines, taking as notion of cost their natural parameters.

1 Motivations

Implicit computational complexity is a young research area, whose main aim is the description of complexity phenomena based on language restrictions, and not on external measure conditions or on explicit machine models. It borrows techniques and results from mathematical logic (model theory, recursion theory, and proof theory) and in doing so it has allowed the incorporation of aspects of computational complexity into areas such as formal methods in software development and programming language design. The most developed area of implicit computational complexity is probably the model theoretic one – finite model theory being a very successful way to describe complexity problems. In the design of programming language tools (e.g., type systems), however, syntactical techniques prove more useful. In the last years we have seen much work restricting recursive schemata and developing general proof theoretical techniques to enforce resource bounds on programs. Important achievements have been the characterizations of several complexity classes by means of limitations of recursive definitions (e.g., [3,9]) and, more recently, by using the "*light*" fragments of

[*] The authors are partially supported by PRIN project "CONCERTO" and FIRB grant RBIN04M8S8, "Intern. Inst. for Applicable Math."

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 163–174, 2009.

linear logic [6]. Moreover, rewriting techniques such as recursive path orderings and the interpretation method have recently been proved useful in the field [10]. By borrowing the terminology from software design technology, we may dub this area as implicit computational complexity *in the large*, aiming at a broad, global view on complexity classes. We may have also an implicit computational complexity *in the small* — using logic to study single machine-free models of computation. Indeed, many models of computations do not come with a natural cost model — a definition of cost which is both intrinsically rooted in the model of computation, and, at the same time, it is polynomially related to the cost of implementing that model of computation on a standard Turing machine. The main example is the λ-calculus: The most natural intrinsic parameter of a computation is its number of beta-reductions, but this very parameter bears no relation, in general, with the actual cost of performing that computation, since a beta-reduction may involve the duplication of arbitrarily big subterms[1]. What we call implicit computational complexity in the small, therefore, gives complexity significance to notions and results for computation models where such natural cost measures do not exist, or are not obvious. In particular, it looks for cost-explicit simulations between such computational models.

The present paper applies this viewpoint to the relation between λ-calculus and orthogonal (constructor) term rewrite systems. We will prove that these two machine models simulate each other with a linear overhead. That each constructor term rewrite system could be simulated by λ-terms and beta-reduction is well known, in view of the availability, in λ-calculus, of fixed-point operators, which may be used to solve the mutual recursion expressed by first-order rewrite rules. Here (Section 4) we make explicit the complexity content of this simulation, by showing that any first-order rewriting of n steps can be simulated by kn beta steps, where k depends on the specific rewrite system but *not* on the size of the involved terms. Crucial to this result is the encoding of constructor terms using Scott's schema for numerals [17]. Indeed, Parigot [11] (see also [12]) shows that in the pure λ-calculus Church numerals do not admit a predecessor working in a constant number of beta steps. Moreover, Splawski and Urzyczyn [15] show that it is unlikely that our encoding could work in the typed context of System F.

Section 3 studies the converse – the simulation of (weak) λ-calculus reduction by means of orthogonal constructor term rewrite systems. We give an encoding of λ-terms into a (first-order) constructor term rewrite system. We write $[\cdot]_\Phi$ for the map returning a first-order term, given a λ-term; $[M]_\Phi$ is, in a sense, a complete defunctionalization of the λ-term M, where any λ-abstraction is represented by an atomic constructor. This is similar, although not technically the same, to the use of supercombinators (e.g., [8]). We show that λ-reduction is simulated step by step by first-order rewriting (Theorem 1).

As a consequence, taking the number of beta steps as a cost model for weak λ-calculus is equivalent (up to a linear function) to taking the number of

[1] In full beta-reduction, the size of the duplicated term is indeed arbitrary and does not depend on the size of the original term the reduction started from. The situation is much different with weak reduction, as we will see.

rewritings in orthogonal constructor term rewrite systems. This is relevant to implicit computational complexity "in the small", because the number of beta steps to normal form is polynomially related to the actual cost (that is, as performed on a Turing machine) of normalization, under weak call-by-value reduction. This has been established by Sands, Gustavsson, and Moran [14], by a fine analysis of a λ-calculus implementation based on a stack machine. Constructor term rewrite systems and λ-calculus are thus both *reasonable* machines (see the "invariance thesis" in [16]), taking as notion of cost their natural, instrinsic parameters.

As a byproduct, in Section 5 we sketch a different proof of the cited result in [14]. Instead of using a stack machine, we show how we could encode constructor term rewriting in term graph rewriting. In term graph rewriting we avoid the explicit duplication and substitution inherent to rewriting (and thus also to beta-reduction) and, moreover, we exploit the possible sharing of subterms.

An extended version of this paper with full proofs is available [5], where we describe also an extension of our results to λ-calculus with call-by-name.

2 Preliminaries

The language we study is the pure untyped λ-calculus endowed with weak (that is, we never reduce under an abstraction) call-by-value reduction.

Definition 1. *The following definitions are standard:*

- Terms *are defined as follows:*

$$M ::= x \mid \lambda x.M \mid MM,$$

  where x ranges a denumerable set Υ. Λ denotes the set of all λ-terms. We assume the existence of a fixed, total, order on Υ; this way $\mathtt{FV}(M)$ will be a sequence (without repetitions) of variables, not a set. A term M is said to be closed *if $\mathtt{FV}(M) = \varepsilon$, where ε is the empty sequence.*
- *Values are defined as follows:*

$$V ::= x \mid \lambda x.M.$$

- *Weak call-by-value reduction is denoted by $\rightarrow_v$ and is obtained by closing call-by-value reduction under any applicative context:*

$$\frac{}{(\lambda x.M)V \rightarrow_v M\{V/x\}} \quad \frac{M \rightarrow_v N}{ML \rightarrow_v NL} \quad \frac{M \rightarrow_v N}{LM \rightarrow_v LN}$$

  Here M ranges over terms, while V ranges over values.
- *The length $|M|$ of M is defined as follows, by induction on M: $|x| = 1$, $|\lambda x.M| = |M| + 1$ and $|MN| = |M| + |N| + 1$.*

Weak call-by-value reduction enjoys many nice properties. In particular, the one-step diamond property holds and, as a consequence, the number of beta steps to normal form (if any) is invariant on the reduction order [4] (this justifies the way we defined reduction, which is slightly more general than Plotkin's one [13]). It is then meaningful to define $Time_v(M)$ as *the* number of beta steps to normal form (or ω if such a normal form does not exist). This cost model will be referred to as the *unitary* cost model, since each beta (weak call-by-value) reduction step counts for 1 in the global cost of normalization. Moreover, notice that α-conversion is not needed during reduction of closed terms: if $M \rightarrow_v N$ and M is closed, then the reduced redex will be in the form $(\lambda x.L)V$, where V is a *closed* value. As a consequence, arguments are always closed and open variables cannot be captured.

The following lemma gives us a generalization of the fixed-point (call-by-value) combinator (but observe the explicit limit k on the reduction length):

Lemma 1. *For every natural number n, there are terms $H_1, \ldots, H_n$ and a natural number m such that for any sequence of values $V_1, \ldots, V_n$ and for any $1 \leq i \leq n$:*

$$H_i V_1 \ldots V_n \rightarrow_v^k V_i(\lambda x.H_1 V_1 \ldots V_n x) \ldots (\lambda x.H_n V_1 \ldots V_n x),$$

where $k \leq m$.

The proof of Lemma 1 can be found in [5].

We will consider in this paper orthogonal constructor (term) rewrite systems (CRS, see [2]). A constructor (term) rewrite system is a pair $\Xi = (\Sigma_\Xi, \mathcal{R}_\Xi)$ where:

- Symbols in the signature Σ_Ξ can be either *constructors* or *function symbols*.
  - terms in $\mathcal{C}(\Xi)$ are those built from constructors and are called *constructor terms*;
  - terms in $\mathcal{P}(\Xi, \Upsilon)$ are those built from constructors and variables and are called *patterns*;
  - terms in $\mathcal{T}(\Xi)$ are those built from constructor and function symbols and are called *closed terms*.
  - terms in $\mathcal{V}(\Xi, \Upsilon)$ are those built from constructors, functions symbols and variables in Υ and are dubbed *terms*.
- Rules in $\mathcal{R}_\Xi$ are in the form $\mathbf{f}(\mathbf{p}_1, \ldots, \mathbf{p}_n) \rightarrow_\Xi t$ where $\mathbf{f}$ is a function symbol, $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathcal{P}(\Xi, \Upsilon)$ and $t \in \mathcal{V}(\Xi, \Upsilon)$. We here consider orthogonal rewrite systems only, i.e. we assume that no distinct two rules in $\mathcal{R}_\Xi$ are overlapping and that every variable appears at most once in the lhs of rules in $\mathcal{R}_\Xi$. Moreover, we assume that reduction is call-by-value, i.e. the substitution triggering any reduction must assign constructor terms to variables. This restriction is anyway natural in constructor rewriting.

For any term t in a CRS, $|t|$ denotes the number of symbol occurrences.

3 From Lambda-Calculus to Constructor Term Rewriting

Definition 2 (The CRS Φ). *The constructor rewrite system Φ is defined as a set of rules $\mathcal{R}_\Phi$ over an infinite signature Σ_Φ. In particular:*

- *The signature Σ_Φ includes the binary function symbol* **app** *and constructor symbols $\mathbf{c}_{x,M}$ for every $M \in \Lambda$ and every $x \in \Upsilon$. The arity of $\mathbf{c}_{x,M}$ is the length of $\mathtt{FV}(\lambda x.M)$. To every term $M \in \Lambda$ we can associate a term $[M]_\Phi \in \mathcal{V}(\Phi, \Upsilon)$ as follows:*

$$\begin{aligned}[x]_\Phi &= x;\\ [\lambda x.M]_\Phi &= \mathbf{c}_{x,M}(x_1,\ldots,x_n),\ \textit{where}\ \mathtt{FV}(\lambda x.M) = x_1,\ldots,x_n;\\ [MN]_\Phi &= \mathbf{app}([M]_\Phi,[N]_\Phi).\end{aligned}$$

  Observe that if M is closed, then $[M]_\Phi \in \mathcal{T}(\Phi)$.
- *The rewrite rules in $\mathcal{R}_\Phi$ are all the rules in the following form:*

$$\mathbf{app}(\mathbf{c}_{x,M}(x_1,\ldots,x_n),x) \rightarrow [M]_\Phi,$$

  where $\mathtt{FV}(\lambda x.M) = x_1,\ldots,x_n$.
- *A term $t \in \mathcal{T}(\Phi)$ is* canonical *if either $t \in \mathcal{C}(\Phi)$ or $t = \mathbf{app}(u,v)$ where u and v are themselves canonical.*

Notice that the signature Σ_Φ contains an infinite amount of constructors.

Example 1. Consider the λ-term $M = (\lambda x.xx)(\lambda y.yy)$. $[M]_\Phi$ is $t \equiv \mathbf{app}(\mathbf{c}_{x,xx},\mathbf{c}_{y,yy})$. Moreover, $t \rightarrow \mathbf{app}(\mathbf{c}_{y,yy},\mathbf{c}_{y,yy}) \equiv u$, as expected. Finally, we have $u \rightarrow u$.

To any term in $\mathcal{V}(\Phi,\Upsilon)$ corresponds a λ-term in Λ:

Definition 3. *To every term $t \in \mathcal{V}(\Phi,\Upsilon)$ we can associate a term $\langle t\rangle_\Lambda \in \Lambda$ as follows: $\langle x\rangle_\Lambda = x$, $\langle \mathbf{app}(u,v)\rangle_\Lambda = \langle u\rangle_\Lambda\langle v\rangle_\Lambda$ and $\langle \mathbf{c}_{x,M}(t_1,\ldots t_n)\rangle_\Lambda = (\lambda x.M)\{\langle t_1\rangle_\Lambda/x_1,\ldots,\langle t_n\rangle_\Lambda/x_n\}$ where $\mathtt{FV}(\lambda x.M) = x_1,\ldots,x_n$.*

Canonicity holds for terms in Φ obtained as images of (closed) λ-terms via $[\cdot]_\Phi$. Moreover, canonicity is preserved by reduction in Φ:

Lemma 2. *For every closed $M \in \Lambda$, $[M]_\Phi$ is canonical. Moreover, if t is canonical and $t \rightarrow u$, then u is canonical.*

For canonical terms, being a normal form is equivalent of being mapped to a normal form via $\langle\cdot\rangle_\Lambda$. This is not true, in general: take as a counterexample $\mathbf{c}_{x,y}(\mathbf{app}(\mathbf{c}_{z,z},\mathbf{c}_{z,z}))$, which corresponds to $\lambda x.(\lambda z.z)(\lambda z.z)$ via $\langle\cdot\rangle_\Lambda$.

Lemma 3. *A canonical term t is a normal form iff $\langle t\rangle_\Lambda$ is a normal form.*

Reduction in Φ can be simulated by reduction in the λ-calculus, provided the starting term is canonical.

Lemma 4. *If t is canonical and $t \rightarrow u$, then $\langle t\rangle_\Lambda \rightarrow_v \langle u\rangle_\Lambda$.*

Conversely, call-by-value reduction in the λ-calculus can be simulated in Φ:

Lemma 5. *If $M \rightarrow_v N$, t is canonical and $\langle t\rangle_\Lambda = M$, then $t \rightarrow u$, where $\langle u\rangle_\Lambda = N$.*

The previous lemmas altogether imply the following theorem, by which λ-calculus normalization can be mimicked (step-by-step) by reduction in Φ:

Theorem 1 (Term Reducibility). *Let $M \in \Lambda$ be a closed term. The following two conditions are equivalent:*

1. *$M \rightarrow_v^n N$ where N is in normal form;*
2. *$[M]_\Phi \rightarrow^n t$ where $\langle t \rangle_\Lambda = N$ and t is in normal form.*

Proof. Suppose $M \rightarrow_v^n N$, where N is in normal form. Then, by applying Lemma 5, we obtain a term t such that $[M]_\Phi \rightarrow^n t$ and $\langle t \rangle_\Lambda = N$. By Lemma 2, t is canonical and, by Lemma 3, it is in normal form. Now, suppose $[M]_\Phi \rightarrow^n t$ where $\langle t \rangle_\Lambda = N$ and t is in normal form. By applying n times Lemma 4, we obtain $\langle [M]_\Phi \rangle_\Lambda \rightarrow_v^n \langle t \rangle_\Lambda = N$. But $\langle [M]_\Phi \rangle_\Lambda = M$ by an easy induction on M and N is a normal form by Lemma 3, since $[M]_\Phi$ and t are canonical by Lemma 2.□

There is another nice property of Φ, that will be crucial in proving the main result of this paper:

Proposition 1. *For every $M \in \Lambda$, for every t with $[M]_\Phi \rightarrow^* t$ and for every occurrence of a constructor $\mathbf{c}_{x,N}$ in t, N is a subterm of M.*

Example 2. Let us consider the λ-term $M = (\lambda x.(\lambda y.x)x)(\lambda z.z)$. Notice that

$$M \rightarrow_v (\lambda y.(\lambda z.z))(\lambda z.z) \rightarrow_v \lambda z.z.$$

Clearly $[M]_\Phi = \mathbf{app}(\mathbf{c}_{x,(\lambda y.x)x}, \mathbf{c}_{z,z})$. Moreover:

$$\mathbf{app}(\mathbf{c}_{x,(\lambda y.x)x}, \mathbf{c}_{z,z}) \rightarrow \mathbf{app}(\mathbf{c}_{y,x}(\mathbf{c}_{z,z}), \mathbf{c}_{z,z}) \rightarrow \mathbf{c}_{z,z}.$$

For every constructor $\mathbf{c}_{w,N}$ occurring in any term in the previous reduction sequence, N is a subterm of M.

A remark on Φ is now in order. Φ is an infinite CRS, since Σ_Φ contains an infinite amount of constructor symbols and, moreover, there are infinitely many rules in $\mathcal{R}_\Phi$. As a consequence, what we have presented here is an embedding of the (weak, call-by-value) λ-calculus into an infinite (orthogonal) CRS. Consider, now, the following scenario: suppose the λ-calculus is used to write a *program* M, and suppose that inputs to M form an infinite set of λ-terms Θ which can anyway be represented by a finite set of constructors in Φ. In this scenario, Proposition 1 allows to conclude the existence of finite subsets of Σ_Φ and $\mathcal{R}_\Phi$ such that *every* MN (where $N \in \Theta$) can be reduced via Φ by using only constructors and rules in those *finite* subsets. As a consequence, we can see the above schema as one that puts any program M in correspondence to a *finite* CRS. Finally, observe that assuming *data* to be representable by a finite number of constructors in Φ is reasonable. Scott's scheme [17], for example, allows to represent any term in a given free algebra in a finitary way, e.g. the natural number 0 becomes $\lceil 0 \rceil \equiv \mathbf{c}_{y,\lambda z.z}$ while $n+1$ becomes $\lceil n+1 \rceil \equiv \mathbf{c}_{y,\lambda z.yx}(\lceil n \rceil)$. Church's scheme, on the other hand, does not have this property.

4 From Constructor Term Rewriting to Lambda-Calculus

In this Section, we will show that any rewriting step of a constructor rewrite system can be simulated by a fixed number of weak call-by-value beta-reductions.

Let Ξ be an orthogonal constructor rewrite system over a finite signature Σ_Ξ. Let $\mathbf{c}_1, \ldots, \mathbf{c}_g$ be the constructors of Ξ and let $\mathbf{f}_1, \ldots, \mathbf{f}_h$ be the function symbols of Ξ. The following constructions work independently of Ξ.

We will first concentrate on constructor terms, encoding them as λ-terms using Scott's schema [17]. Constructor terms can be easily put in correspondence with λ-terms by way of a map $\langle\!\langle \cdot \rangle\!\rangle_\Lambda$ defined by induction as follows:

$$\langle\!\langle \mathbf{c}_i(t_1 \ldots, t_n) \rangle\!\rangle_\Lambda \equiv \lambda x_1. \ldots .\lambda x_g.\lambda y.x_i \langle\!\langle t_1 \rangle\!\rangle_\Lambda \ldots \langle\!\langle t_n \rangle\!\rangle_\Lambda.$$

This way constructors become functions:

$$\langle\!\langle \mathbf{c}_i \rangle\!\rangle_\Lambda \equiv \lambda x_1. \ldots .\lambda x_{ar(\mathbf{c}_i)}.\lambda y_1. \ldots .\lambda y_g.\lambda z.y_i x_1 \ldots x_{ar(\mathbf{c}_i)}.$$

Trivially, $\langle\!\langle \mathbf{c}_i \rangle\!\rangle_\Lambda \langle\!\langle t_1 \rangle\!\rangle_\Lambda \ldots \langle\!\langle t_n \rangle\!\rangle_\Lambda$ rewrites to $\langle\!\langle \mathbf{c}_i(t_1 \ldots t_n) \rangle\!\rangle_\Lambda$ in $ar(\mathbf{c}_i)$ steps. The λ-term $\lambda x_1. \ldots .\lambda x_g.\lambda y.y$ corresponds to an error value, and is denoted as $\perp$. A λ-term built in this way, i.e. a λ-term which is either $\perp$ or in the form $\langle t \rangle_\Lambda$ is denoted with metavariables like X or Y.

The map $\langle\!\langle \cdot \rangle\!\rangle_\Lambda$ defines encodings of constructor terms. But what about terms containing function symbols? The goal is defining another map $[\cdot]_\Lambda$ returning a λ-term given any term t in $\mathcal{T}(\Xi)$, in such a way that $t \rightarrow^* u$ and $u \in \mathcal{C}(\Xi)$ implies $[t]_\Lambda \rightarrow_v^* \langle\!\langle u \rangle\!\rangle_\Lambda$. Moreover, $[t]_\Lambda$ should rewrite to $\perp$ whenever the rewriting of t causes an error (i.e. whenever t has a normal form containing a function symbol). The map we are looking for should act compositionally on terms:

$$\begin{aligned}
[\mathbf{c}(t_1, \ldots, t_{ar(\mathbf{c})})]_\Lambda &= [\mathbf{c}]_\Lambda [t_1]_\Lambda \ldots [t_{ar(\mathbf{c})}]_\Lambda \\
[\mathbf{f}(t_1, \ldots, t_{ar(\mathbf{f})})]_\Lambda &= [\mathbf{f}]_\Lambda [t_1]_\Lambda \ldots [t_{ar(\mathbf{f})}]_\Lambda.
\end{aligned}$$

As a consequence, we only need to define our map on constructors and on function symbols. Defining the λ-term $[\mathbf{c}]_\Lambda$ corresponding to a constructor $\mathbf{c}$ is relatively easy: we simply need to take into account the case when some argument is $\perp$. Treating function symbols is more complicated, since the rewrite rules governing $\mathbf{f}$ must be "embedded" inside $[\mathbf{f}]_\Lambda$. One of the main ingredients in rewriting is first-order matching, which is not natively available in the *pure* λ-calculus, and should therefore be coded into the λ-term. This is the purpose of the following lemma.

Lemma 6 (Pattern matching). *Let $\alpha_1, \ldots, \alpha_n$ be non-overlapping sequences of patterns of the same length m. Then there are a term $M^m_{\alpha_1,\ldots,\alpha_n}$ and an integer l such that for every sequence of values $V_1, \ldots, V_n$, if $\alpha_i = \mathbf{p}_1, \ldots, \mathbf{p}_m$ then*

$$\begin{aligned}
M^m_{\alpha_1,\ldots,\alpha_n} \langle\!\langle \mathbf{p}_1(t_1^1, \ldots, t_1^{k_1}) \rangle\!\rangle_\Lambda \ldots \langle\!\langle \mathbf{p}_m(t_m^1, \ldots, t_m^{k_m}) \rangle\!\rangle_\Lambda V_1 \ldots V_n \\
\rightarrow_v^k \; V_i \langle\!\langle t_1^1 \rangle\!\rangle_\Lambda \ldots \langle\!\langle t_1^{k_1} \rangle\!\rangle_\Lambda \ldots \langle\!\langle t_m^1 \rangle\!\rangle_\Lambda \ldots \langle\!\langle t_m^{k_m} \rangle\!\rangle_\Lambda,
\end{aligned}$$

where $k \leq l$, whenever the t_i^j are constructor terms. Moreover,

$$M^m_{\alpha_1,\ldots,\alpha_n} X_1,\ldots,X_m V_1 \ldots V_n \rightarrow_v^k \bot,$$

where $k \leq l$, whenever $X_1,\ldots,X_m$ do not unify with any of the sequences $\alpha_1,\ldots,\alpha_n$ or any of the $X_1,\ldots,X_m$ is itself $\bot$.

Rewrite rules define the computational behavior of function symbols in a mutually recursive way. We can exploit the powerful fixed-point combinators of the untyped λ-calculus (see Lemma 1) to define $[\mathbf{f}]_\Lambda$.

Theorem 2. *There is a natural number k such that for every function symbol $\mathbf{f}$ and for every $t_1,\ldots,t_{ar(\mathbf{f})} \in \mathcal{C}(\Xi)$, the following three implications hold (where u stands for $\mathbf{f}(t_1,\ldots,t_{ar(\mathbf{f})})$ and M stands for $[\mathbf{f}]_\Lambda \langle\langle t_1\rangle\rangle_\Lambda \ldots \langle\langle t_{ar(\mathbf{f})}\rangle\rangle_\Lambda$):*

- *If u rewrites to $v \in \mathcal{C}(\Xi)$ in n steps, then M rewrites to $\langle\langle v\rangle\rangle_\Lambda$ in at most kn steps.*
- *If u rewrites to a normal form $v \notin \mathcal{C}(\Xi)$, then M rewrites to $\bot$.*
- *If u diverges, then M diverges.*

Informally, any term $[\mathbf{f}_i]_\Lambda$ has the form $H_i V_1 \ldots V_h$ where

$$V_i = \lambda x_1.\ldots.\lambda x_h.\lambda y_1.\ldots.\lambda y_{ar(\mathbf{f}_i)}.M_{\alpha_1,\ldots,\alpha_n} y_1 \ldots y_{ar(\mathbf{f}_i)} W_1 \ldots W_n,$$

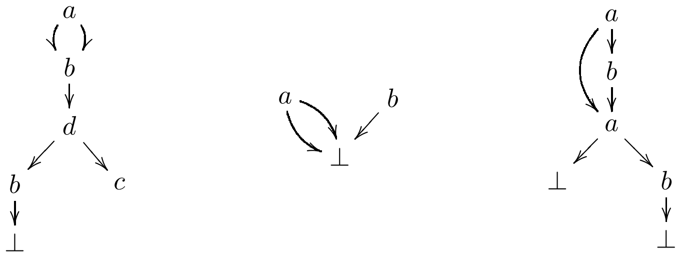
n is the number of reduction rules "governing" $\mathbf{f}_i$, α_i is the sequence of patterns appearing in the i-th such rule and W_i corresponds to the rhs of the same rule. As a consequence, the natural number k of Theorem 2 can be obtained from the corresponding constants from Lemma 1 and Lemma 6. Clearly, the constant k in Theorem 2 depends on Ξ, but is independent on the particular term u.

5 By-product: Polynomial Invariance

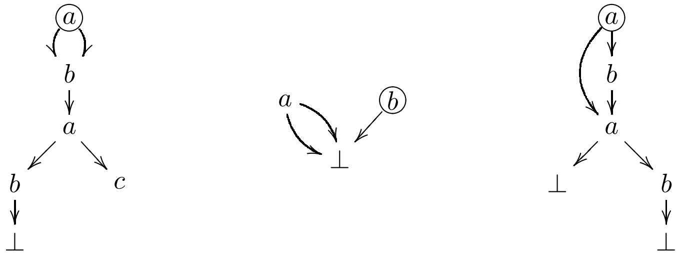
In this section, we will show how the correspondence between λ-calculus and constructor rewrite systems allows to prove an invariance result on λ-calculus originally established by Sands, Gustavsson, and Moran [14]. We sketch here how we may obtain the same result by using our simulation results. The interested reader could find all the technical details in [5].

The result we are interested in is the following: for weak call-by-value λ-calculus, the number of beta-reductions to normal form is polynomially related to the actual cost of computing that normal form on a Turing machine (and thus on any other reasonable machine model). Both in constructor term rewriting and in the λ-calculus, proving the number of reduction steps to normal form to be a "good" cost model is not trivial. Indeed, the substitution process could involve the manipulation (in particular, the copying) of terms whose size is not even polynomially related to the size of the original term nor to the number of reduction steps performed insofar. As a consequence, substitution must be implemented carefully, and terms have to be represented implicitly, exploiting sharing of subterms. The term rewriting community has already developed a formalism which allows us to exploit sharing, namely term graph rewriting. The desired result thus follows easily once we observe that *every* (orthogonal) constructor rewrite system is graph reducible (which is the property expressed by Theorem 3, below; recall that this does not hold for arbitrary term rewriting systems [1]).

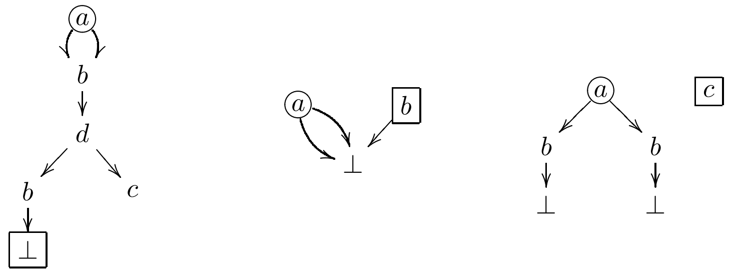
Given a signature Σ, a *labelled graph over* Σ consists of a directed acyclic graph together with an ordering on the outgoing edges of each node, and a (partial) labelling of nodes with symbols from Σ such that the out-degree of each node matches the arity of the corresponding symbols (and is 0 if the labelling is undefined). As an example, consider the signature $\Sigma = \{a, b, c, d\}$, where arities of a, b, c, d are 2, 1, 0, 2 respectively, and b, c, d are constructors. Examples of labelled graphs over the signature Σ are the following ones:



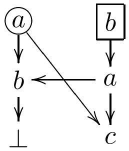
The symbol $\perp$ denotes vertices where the underlying labelling function is undefined (and, as a consequence, no edge departs from such vertices). Their role is similar to the one of variables in terms. A *term graph* is a labelled graph G together with a node r of G, the *root* of the term graph. As an example, the following are graphic representations of some term graphs (over the same signature Σ considered in the previous examples).



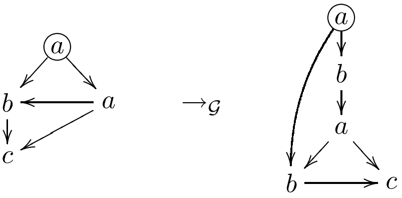
The root is the only vertex drawn inside a circle. We now need a way to write programs acting on term graphs. A *graph rewrite rule* over a signature Σ consists of a labelled graph G together with two vertices r, s of G. In the spirit of constructor rewriting, r must be labelled with a function symbol, while the label of any vertex reachable from r (if any) must be a constructor symbol. Moreover, every non-labelled vertex reachable from s can be reached from r, too. The following are examples of graph rewriting rules, assuming a to be a function symbol and b, c, d to be constructors:



The circled vertex corresponds to r, while the squared vertex corresponds to s. A set of graph rewrite rules over a signature Σ defines a constructor graph rewrite system $\mathcal{G}$. The relation $\rightarrow_{\mathcal{G}}$ is defined in the natural way (see [5]). As an example, if $\mathcal{G}$ includes the following rewrite rule



then



Given a constructor rewrite system $\mathcal{R}$ over Σ, the corresponding constructor graph rewrite system $\mathcal{G}$ is defined as the class of graph rewrite rules corresponding to those in $\mathcal{R}$. Given a term t, $[t]_{\mathcal{G}}$ will be the corresponding graph, while the term graph G corresponds to the term $\langle G\rangle_{\mathcal{R}}$. Contrarily to what happens in term rewriting, any construtor rewrite system is graph reducible:

Theorem 3 (Graph Reducibility). *For every constructor rewrite system $\mathcal{R}$ over Σ and for every term t over Σ, the following two conditions are equivalent:*

1. $t \rightarrow^n u$*, where* u *is in normal form;*
2. $[t]_{\mathcal{G}} \rightarrow^n G$*, where* G *is in normal form and* $\langle G\rangle_{\mathcal{R}} = u$*.*

As a corollary (and thanks to Proposition 1), we get:

Corollary 1. *There is a polynomial* $p : \mathbb{N}^2 \rightarrow \mathbb{N}$ *such that for every* λ*-term* M*, the normal form of* $[[M]_{\Phi}]_{\Theta}$ *can be computed in time at most* $p(|M|, \mathit{Time}_v(M))$*.*

This cannot be achieved when using explicit representations of λ-terms. Moreover, reading back a λ-term from a term graph may take exponential time.

Example 3. Consider the λ-terms

$$M_n \equiv \underbrace{\overline{2}(\overline{2}(\overline{2}(\ldots(\overline{2}}_{n \text{ times}}(\lambda x.x))\ldots))),$$

where $\overline{2}$ is the Church numeral for 2. Their size is proportional to n, while the size of their normal form is exponential in n. However, the number of reduction steps leading to the normal form is *linear* in n. Altogether, this implies that the number of reduction steps to normal form cannot be a cost model *if we represent λ-terms explicitly*, i.e., if we require the output to be a λ-term expressed as a string. Observe, on the other hand, that the normal form $[[M_n]_\Phi]_\Theta$ is a graph whose size is again linear in n.

We can complement Corollary 1 with a completeness statement — any universal computational model with an invariant cost model can be embedded in the λ-calculus with a polynomial overhead. We can exploit for this the analogous result we proved in [4] (Theorem 1) — the unitary cost model is easily proved to be more parsimonious than the difference cost model considered in [4].

Theorem 4. *Let $f : \Sigma^* \to \Sigma^*$ be computed by a Turing machine $\mathcal{M}$ in time g. Then, there are a λ-term $N_\mathcal{M}$ and a suitable encoding $\ulcorner\cdot\urcorner : \Sigma^* \to \Lambda$ such that $N_\mathcal{M}\ulcorner v\urcorner$ normalizes to $\ulcorner f(v)\urcorner$ in $O(g(|v|))$ beta steps.*

6 Conclusions

We have shown that the most naïve cost models for weak call-by-value λ-calculus (each beta-reduction step has unitary cost) and orthogonal constructor term rewriting (each rule application has unitary cost) are linearly related. Since, in turn, this cost model for λ-calculus is polynomially related to the actual cost of reducing a λ-term on a Turing machine, the two machine models we considered are both *reasonable* machines, when endowed with their natural, intrinsic cost models (see also Gurevich's opus on Abstract State Machine simulation "at the same level of abstraction", e.g. [7]). This strong (the embeddings we consider are compositional), complexity-preserving equivalence between a first-order and a higher-order model is the most important technical result of the paper.

Ongoing and future work includes the investigation of how much of this simulation could be recovered either in a typed setting (see [15] for some of the difficulties), or in the case of λ-calculus with strong reduction, where we reduce under an abstraction. Novel techniques have to be developed, since the analysis we performed in the present paper cannot be easily extended to these cases.

Acknowledgments

The authors wish to thank Kazushige Terui for stimulating discussions on the topics of this paper.

References

1. Barendregt, H., Eekelen, M., Glauert, J., Kennaway, J., Plasmeijer, M., Sleep, M.: Term graph rewriting. In: de Bakker, J., Nijman, A., Treleaven, P. (eds.) Parallel Languages on PARLE: Parallel Architectures and Languages Europe, vol. II, pp. 141–158. Springer, Heidelberg (1986)
2. Barendsen, E.: Term graph rewriting. In: Bezem, M., Klop, J.W., de Vrijer, R. (eds.) Term Rewriting Systems, pp. 712–743. Cambridge Univ. Press, Cambridge (2003)
3. Bellantoni, S., Cook, S.: A new recursion-theoretic characterization of the polytime functions. Computational Complexity 2, 97–110 (1992)
4. Dal Lago, U., Martini, S.: An invariant cost model for the lambda-calculus. In: Beckmann, A., Berger, U., Löwe, B., Tucker, J.V. (eds.) CiE 2006. LNCS, vol. 3988, pp. 105–114. Springer, Heidelberg (2006)
5. Dal Lago, U., Martini, S.: On constructor rewrite systems and the lambda-calculus. Extended Version (2009), http://arxiv.org/abs/0904.4120
6. Girard, J.-Y.: Light linear logic. Inform. and Comp. 143(2), 175–204 (1998)
7. Gurevich, Y.: The sequential ASM thesis. In: Current trends in theoretical computer science, pp. 363–392. World Scientific, Singapore (2001)
8. Jones, S.P.: The Implementation of Functional Programming Languages. Prentice-Hall, Englewood Cliffs (1987)
9. Leivant, D.: Ramified recurrence and computational complexity I: word recurrence and poly-time. In: Feasible Mathematics II, pp. 320–343. Birkhäuser, Basel (1995)
10. Marion, J.-Y., Moyen, J.-Y.: Efficient first order functional program interpreter with time bound certifications. In: Parigot, M., Voronkov, A. (eds.) LPAR 2000. LNCS, vol. 1955, pp. 25–42. Springer, Heidelberg (2000)
11. Parigot, M.: On the representation of data in lambda-calculus. In: Börger, E., Kleine Büning, H., Richter, M.M. (eds.) CSL 1989. LNCS, vol. 440, pp. 309–321. Springer, Heidelberg (1990)
12. Parigot, M., Rozière, P.: Constant time reductions in lambda-caculus. In: Borzyszkowski, A.M., Sokolowski, S. (eds.) MFCS 1993. LNCS, vol. 711, pp. 608–617. Springer, Heidelberg (1993)
13. Plotkin, G.D.: Call-by-name, call-by-value and the lambda-calculus. Theoretical Computer Science 1(2), 125–159 (1975)
14. Sands, D., Gustavsson, J., Moran, A.: Lambda calculi and linear speedups. In: Mogensen, T.Æ., Schmidt, D.A., Sudborough, I.H. (eds.) The Essence of Computation. LNCS, vol. 2566, pp. 60–82. Springer, Heidelberg (2002)
15. Splawski, Z., Urzyczyn, P.: Type fixpoints: Iteration vs. recursion. In: 4th International Conference on Functional Programming, Proceedings, pp. 102–113. ACM Press, New York (1999)
16. van Emde Boas, P.: Machine models and simulation. In: Handbook of Theoretical Computer Science, Algorithms and Complexity, pp. 1–66. MIT Press, Cambridge (1990)
17. Wadsworth, C.: Some unusual λ-calculus numeral systems. In: Seldin, J.P., Hindley, J.R. (eds.) To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism. Academic Press, London (1980)

On Regular Temporal Logics with Past[*, **]

Christian Dax[1], Felix Klaedtke[1], and Martin Lange[2]

[1] ETH Zurich, Switzerland
[2] Ludwig-Maximilians-University Munich, Germany

Abstract. The IEEE standardized *Property Specification Language*, PSL for short, extends the well-known linear-time temporal logic LTL with so-called semi-extended regular expressions. PSL and the closely related *SystemVerilog Assertions*, SVA for short, are increasingly used in many phases of the hardware design cycle, from specification to verification. In this paper, we extend the common core of these specification languages with past operators. We name this extension RTL. Although all ω-regular properties are expressible in PSL, SVA, and RTL, past operators often allow one to specify properties more naturally and concisely. In fact, we show that RTL is exponentially more succinct than the cores of PSL and SVA. Furthermore, we present a translation of RTL into language-equivalent nondeterministic Büchi automata, which is based on novel constructions for 2-way alternating automata. Our translation has almost the same worst-case complexity in terms of the size of the resulting nondeterministic Büchi automata as the existing translations for PSL and SVA. Consequently, the satisfiability and the model-checking problem for RTL fall into the same complexity classes as the corresponding problems for PSL and SVA. From the translation it also follows that the blowup of translating RTL formulas into initially equivalent PSL/SVA formulas is at most triply exponential.

1 Introduction

The industry standardized temporal logics PSL [1] and SVA (the assertion language of SystemVerilog [2]) are increasingly used in the hardware industry to formally express, validate, and verify the requirements of circuit designs. The linear-time core of PSL extends the well-known linear-time temporal logic LTL with semi-extended regular expressions (SEREs), which are essentially regular expressions with an additional operator for expressing the intersection of languages. The core of SVA can be seen as a subset of PSL.[1] The prominence of PSL and SVA in industry over other specification languages like LTL [23],

[*] Partly supported by the Swiss National Science Foundation (SNF).

[**] Due to space limitations, most proofs have been omitted. These can be found in an extended version of the paper, which is available from the authors' webpages.

[1] For the ease of exposition, we identify, similar to [5,9,7,24], PSL and SVA with their respective cores. In particular, the cores are "unclocked," they do not contain local variables (which are not part of the PSL standard), and their semantics is only defined over infinite words.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 175–187, 2009.

μLTL [4], and ETL [28] is that PSL and SVA balance well the competing needs of a specification language like *expressiveness*, *usability*, and *implementability* [3]: all ω-regular languages are expressible in PSL/SVA, specifications in PSL/SVA are fairly easy to read and write, and relevant verification problems (e.g. model checking) for PSL/SVA are automatically solvable in practice.

Although temporal operators that refer to the past have been found natural and useful when expressing temporal properties [20, 16, 21, 10, 9], the PSL and SVA standards support temporal past operators only in a restrictive way. This design choice has already been made for the predecessor ForSpec [3] of PSL/SVA and has been justified by the argument that handling "arbitrary mixing of past and future operators results in nonnegligible implementation cost" [3]. One reason for this belief is that in the automata-theoretic approach to model checking [27], one uses 2-way automata to deal with past and future operators rather than 1-way automata when only future operators are present. The nowadays used automata constructions for 2-way automata are more involved than the corresponding ones for 1-way automata. For instance, with the state-of-the-art construction in [16], we can translate a 2-way alternating Büchi automaton with n states into a language-equivalent nondeterministic Büchi automaton (NBA) with $2^{\mathcal{O}(n^2)}$ states. For a given 1-way alternating Büchi automaton, we obtain with the Miyano-Hayashi construction [22] an NBA with only $2^{\mathcal{O}(n)}$ states. Nevertheless, in this paper, we give arguments in favor of extending PSL and SVA with past operators and we argue against this assumed additional implementation cost. In particular, one of our results shows that a restricted class of 2-way automata suffices and the additional cost for this class is small.

In more detail, the content of the paper is as follows. We first propose an extension of PSL with past operators, which we name *Regular Temporal Logic*, RTL for short. RTL extends PSL by the standard past operators from linear-time temporal logic and by the corresponding past operators of the PSL/SVA-specific operators for SEREs. For example, the PSL/SVA-specific operator $\alpha \mathrel{\Diamond\!\!\!\rightarrow} \varphi$ describes that a system trace fulfills from the current time point the pattern given by the SERE α and at the end the *post-condition* φ holds, where φ is a PSL/SVA formula. RTL additionally contains the corresponding counterpart $\alpha \mathrel{\Diamond\!\!\!\!-\!\!\!\rightarrow} \varphi$. This describes that the *pre-condition* φ holds at some time point in the past and at that time point the system trace fulfills up to the current time point the pattern α. Note that the temporal operator $\alpha \mathrel{\Diamond\!\!\!\rightarrow} \varphi$ is closely related to the modality $\langle\alpha\rangle\varphi$ in dynamic logic [15]. However, PSL/SVA uses SEREs over state predicates and in dynamic logic, the expressions are over program statements.

PSL, SVA, and RTL have the same expressive power: they all describe the class of ω-regular languages. However, RTL allows one to describe ω-regular languages more concisely than PSL and SVA. To show this, we establish a lower bound on the succinctness of RTL and SVA. We define a family of ω-regular languages and prove that these languages can be described in RTL exponentially more succinctly than in SVA. For the LTL-expressible properties, i.e, the ω-regular languages that are star-free, we obtain as a byproduct that RTL is double exponentially more succinct than LTL, even when extended with the classical temporal past operators Y (yesterday) and S (since).

Furthermore, we investigate the additional computational cost for solving the satisfiability problem and the model-checking problem for RTL. As for PSL and SVA, these problems are EXPSPACE-complete for RTL. In practice, the satisfiability problem and the model-checking problem for PSL and SVA are solved by using an automata-theoretic approach [5, 9, 7], translating a given formula into an NBA. With the standard automata constructions for PSL and SVA, one obtains for a PSL/SVA formula of size n an NBA of size $\mathcal{O}(2^{2\cdot 2^{2n}})$ [5,7]. We present a novel construction for RTL that translates an RTL formula of size n into an NBA of size $\mathcal{O}(2^{3\cdot 2^{2n}})$. Note that the upper bounds of the sizes of the resulting automata for PSL/SVA and RTL only differ by a small constant in the exponent despite the richer structure of RTL. Our translation is based on alternation-elimination constructions for restricted classes of 2-way alternating automata that were recently presented in [12] and which we further improve in this paper for the alternating automata that we obtain from our translation of RTL formulas into alternating automata. This construction can also be used to translate a given RTL formula into an initially equivalent SVA formula whose size is triple exponentially larger, not quite matching the lower bound mentioned above. One of these three exponentials is due to the fact that the resulting SVA formulas do not contain SEREs anymore, but only regular expressions.

We point out that our translation for RTL into NBAs significantly improves over translations that we obtain when utilizing automata constructions that do not take the given special class of alternating automata into account. For instance, when using the state-of-the-art construction [16] for translating 2-way alternating automata into NBAs, one obtains an NBA of size $\mathcal{O}(2^{4\cdot 2^{4n}+2^{2n}})$, where n is again the size of the given RTL formula. Overall, the presented translation indicates that extensions of temporal logics with past operators can be handled with only a minor overhead in the automata-theoretic model-checking approach when adequate constructions for 2-way alternating automata are used.

2 Preliminaries

Words and Trees. We denote the set of finite words over the alphabet Σ by Σ^* and the set of infinite words over Σ by Σ^ω. The length of a word $w \in \Sigma^*$ is written as $|w|$ and ε denotes the empty word. For a finite or infinite word w, w_i denotes the symbol of w at position $i \in \mathbb{N}$, where we assume that $i < |w|$ if w is finite. We write $v \preceq w$ if v is a prefix of the word w. For $i, j < |w|$, we write $w_{i..}$ for the suffix $w_i w_{i+1} \dots$ and $w_{i..j}$ for the subword $w_i w_{i+1} \dots w_j$.

A (Σ-labeled) *tree* is a function $t : T \to \Sigma$, where $T \subseteq \mathbb{N}^*$ satisfies the conditions: (i) T is prefix-closed (i.e., if $v \in T$ and $u \preceq v$ then $u \in T$) and (ii) if $vi \in T$ and $i > 0$ then $v(i-1) \in T$. The elements in T are called the *nodes* of t and the empty word ε is called the *root* of t. A node $vi \in T$ with $i \in \mathbb{N}$ is called a *child* of the node $v \in T$. An (infinite) *path* in t is a word $\pi \in \mathbb{N}^\omega$ such that $v \in T$, for every prefix v of π. We write $t(\pi)$ for the word $t(\pi_0)t(\pi_1)\dots \in \Sigma^\omega$.

Propositional Logic. We denote the set of *Boolean formulas* over the set P of propositions by $\mathcal{B}(P)$, i.e., $\mathcal{B}(P)$ consists of the formulas that are inductively built from the propositions in P and the connectives $\vee$, $\wedge$, and $\neg$. For $M \subseteq P$ and $b \in \mathcal{B}(P)$, we write $M \models b$ iff b evaluates to true when assigning true to the propositions in M and false to the propositions in $P \setminus M$. We write $\mathcal{B}^+(P)$ for the set of Boolean formulas in which the connective $\neg$ does not occur.

Regular Expressions. The syntax of *semi-extended regular expressions* (SEREs) over the proposition set P is defined by the grammar $\alpha ::= \varepsilon \mid b \mid \alpha \star \alpha \mid \alpha^*$, where $b \in \mathcal{B}(P)$ and $\star \in \{\cup, \cap, ;, :\}$. The language of an SERE over the proposition set P is inductively defined: (i) $L(\varepsilon) := \{\varepsilon\}$, (ii) $L(b) := \{w \in (2^P)^* \mid |w| = 1 \text{ and } w \models \alpha\}$, for $b \in \mathcal{B}(P)$, (iii) $L(\beta \star \gamma) := L(\beta) \star L(\gamma)$, for $\star \in \{\cup, \cap, ;, :\}$, where $L \,;\, L' := \{uv \mid u \in L \text{ and } v \in L'\}$ is the concatenation of L and L', and $L : L' := \{ubv \mid ub \in L \text{ and } bv \in L' \text{ with } b \in 2^P\}$ the fusion, and (iv) $L(\beta^*) := \bigcup_{n \in \mathbb{N}} L^n(\beta)$, where $L^0 := \{\varepsilon\}$ and $L^{i+1} := L \,;\, L^i$, for all $i \in \mathbb{N}$. The *size* of an SERE is its syntactic length, i.e., $\|\varepsilon\| := 1$, $\|b\| := 1$, for $b \in \mathcal{B}(P)$, $\|\beta \star \gamma\| := 1 + \|\beta\| + \|\gamma\|$, for $\star \in \{\cup, \cap, ;, :\}$, and $\|\beta^*\| := 1 + \|\beta\|$.

Automata. In the following, we define 2-way alternating automata, which scan input words letter by letter with their read-only head. Let $\mathbb{D} := \{-1, 0, 1\}$ be the set of directions in which the read-only head can move. A *2-way alternating Büchi automaton* (2ABA) $\mathcal{A}$ is a tuple $(Q, \Sigma, \delta, q_I, F)$, where Q is a finite set of states, Σ is a finite nonempty alphabet, $\delta : Q \times \Sigma \to \mathcal{B}^+(Q \times \mathbb{D})$ is the transition function, $q_I \in Q$ is the initial state, and $F \subseteq Q$ is the acceptance condition. The *size* $\|\mathcal{A}\|$ of the automaton $\mathcal{A}$ is $|Q|$.

A *configuration* of $\mathcal{A}$ is a pair $(q, i) \in Q \times \mathbb{N}$. Intuitively, q is the current state and the read-only head is at position i of the input word. A *run* of $\mathcal{A}$ on $w \in \Sigma^\omega$ is a tree $r : T \to Q \times \mathbb{N}$ such that $r(\varepsilon) = (q_I, 0)$ and

$$\big\{(q', j' - j) \in Q \times D \mid r(y) = (q', j'), \text{ where } y \text{ is a child of } x \text{ in } r\big\} \models \delta(q, w_j)\,,$$

for each node $x \in T$ with $r(x) = (q, j)$. For $\pi := (q_0, i_0)(q_1, i_1) \ldots \in (Q \times \mathbb{N})^\omega$, we define $\mathit{Inf}(\pi) := \{q \mid q \text{ occurs infinitely often in } q_0 q_1 \ldots \in Q^\omega\}$. A path $\pi \in T$ in a run r is *accepting* if $\mathit{Inf}(r(\pi)) \cap F \neq \emptyset$. The run r is *accepting* if every path in r is accepting. The *language* of $\mathcal{A}$ is the set $L(\mathcal{A}) := \{w \in \Sigma^\omega \mid \text{there is an accepting run of } \mathcal{A} \text{ on } w\}$.

The automaton $\mathcal{A}$ is *1-way* if $\delta(q, a) \in \mathcal{B}^+(Q \times \{1\})$, for all $q \in Q$ and $a \in \Sigma$. That means, $\mathcal{A}$ can only move the read-only head to the right. If $\mathcal{A}$ is 1-way, we assume that δ is of the form $\delta : Q \times \Sigma \to \mathcal{B}^+(Q)$. We call a 1-way automaton a *nondeterministic Büchi automaton* (NBA) if its transition function returns a disjunction of states for all inputs. We view the transition function δ of an NBA as a function of the form $\delta : Q \times \Sigma \to 2^Q$. This means that clauses are written as sets. Note that a run $r : T \to Q \times \mathbb{N}$ of an NBA $\mathcal{A}$ on $w \in \Sigma^\omega$ can be reduced to a single path π in r that is consistent with the transition function. Using standard terminology, we also call $r(\pi) \in (Q \times \mathbb{N})^\omega$ a run of $\mathcal{A}$ on w.

$$\begin{array}{lll}
w,i \models p & \text{iff} & p \in w_i \\
w,i \models \mathsf{cl}(\alpha) & \text{iff} & \exists k \geq i : w_{i..k} \in L(\alpha), \text{ or } \forall k \geq i : \exists v \in L(\alpha) : w_{i..k} \preceq v \\
w,i \models \varphi \wedge \psi & \text{iff} & w,i \models \varphi \text{ and } w,i \models \psi \\
w,i \models \neg\varphi & \text{iff} & w,i \not\models \varphi \\
w,i \models \mathsf{X}\varphi & \text{iff} & w,i+1 \models \varphi \\
w,i \models \varphi \,\mathsf{U}\, \psi & \text{iff} & \exists k \geq i : w,k \models \psi \text{ and } \forall j : \text{if } i \leq j < k \text{ then } w,j \models \varphi \\
w,i \models \alpha \mathrel{\Diamond\!\!\rightarrow} \varphi & \text{iff} & \exists k \geq i : w_{i..k} \in L(\alpha) \text{ and } w,k \models \varphi \\
w,i \models \mathsf{Y}\varphi & \text{iff} & i > 0 \text{ and } w,i-1 \models \varphi \\
w,i \models \varphi \,\mathsf{S}\, \psi & \text{iff} & \exists k \leq i : w,k \models \psi \text{ and } \forall j : \text{if } k < j \leq i \text{ then } w,j \models \varphi \\
w,i \models \alpha \mathrel{\Diamond\!\!\!\!\!-\!\!\rightarrow} \varphi & \text{iff} & \exists k \leq i : w_{k..i} \in L(\alpha) \text{ and } w,k \models \varphi
\end{array}$$

Fig. 1. Interpretation of an RTL formula over P at a position $i \geq 0$ of a word $w \in (2^P)^\omega$

3 Temporal Logics with Expressions and Past Operators

In this section, we extend LTL with SEREs and past operators. We call the extension *Regular Temporal Logic*, RTL for short. The cores of the two industrial-standard property-specification languages PSL [1] and SVA [2] are fragments of RTL. The syntax of RTL over the set P of propositions is given by the grammar

$$\varphi ::= p \mid \mathsf{cl}(\alpha) \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathsf{X}\varphi \mid \varphi \,\mathsf{U}\, \varphi \mid \alpha \mathrel{\Diamond\!\!\rightarrow} \varphi \mid \mathsf{Y}\varphi \mid \varphi \,\mathsf{S}\, \varphi \mid \alpha \mathrel{\Diamond\!\!\!\!\!-\!\!\rightarrow} \varphi\,,$$

where $p \in P$ and α is an SERE over P. The semantics of RTL is given in Figure 1. A word $w \in (2^P)^\omega$ is a *model* of an RTL formula φ if $w,0 \models \varphi$. The *language* of an RTL formula φ is $L(\varphi) := \{w \in (2^P)^\omega \mid w,0 \models \varphi\}$. The RTL formulas φ and ψ are *initially equivalent* if $L(\varphi) = L(\psi)$. They are *logically equivalent*, written as $\varphi \equiv \psi$, if $w,i \models \varphi \Leftrightarrow w,i \models \psi$, for all $i \in \mathbb{N}$ and $w \in (2^P)^\omega$. As for SEREs, we define the *size* $\|\varphi\|$ of an RTL formula φ as its syntactic length.

We define the following fragments of RTL. We call an RTL formula a *PSL formula* if it does not contain the operators Y, S, and $\mathrel{\Diamond\!\!\!\!\!-\!\!\rightarrow}$. An *LTL formula* is a PSL formula that does not contain the operators cl and $\mathrel{\Diamond\!\!\rightarrow}$. An *SVA formula* is a PSL formula that does not contain the operators cl, X, and U. The fragments PLTL and PSVA, which extend LTL and SVA, respectively, with past operators, are defined as expected. Note that RTL and PSL extended with the past operators Y, S, and $\mathrel{\Diamond\!\!\!\!\!-\!\!\rightarrow}$ coincide.

We use standard syntactic sugar, like the Boolean constants and connectives ff, tt, $\vee$, $\rightarrow$, and we define $\varphi \,\mathsf{R}\, \psi := \neg(\neg\varphi \,\mathsf{U}\, \neg\psi)$, $\varphi \,\mathsf{T}\, \psi := \neg(\neg\varphi \,\mathsf{S}\, \neg\psi)$, $\mathsf{Z}\varphi := \mathsf{Ytt} \rightarrow \mathsf{Y}\varphi$. Moreover, for an RTL formula φ and an SERE α, we write $\alpha \mathrel{\Box\!\!\rightarrow} \varphi$ for $\neg(\alpha \mathrel{\Diamond\!\!\rightarrow} \neg\varphi)$ and $\alpha \mathrel{\boxminus\!\!\rightarrow} \varphi$ for $\neg(\alpha \mathrel{\Diamond\!\!\!\!\!-\!\!\rightarrow} \neg\varphi)$. Note that the standard unary temporal operators can easily be defined in the respective fragment. For instance, for PSVA we define $\mathsf{G}\varphi := \mathsf{tt}^* \mathrel{\Box\!\!\rightarrow} \varphi$, $\mathsf{F}\varphi := \mathsf{tt}^* \mathrel{\Diamond\!\!\rightarrow} \varphi$, $\mathsf{H}\varphi := \mathsf{tt}^* \mathrel{\boxminus\!\!\rightarrow} \varphi$, and $\mathsf{O}\varphi := \mathsf{tt}^* \mathrel{\Diamond\!\!\!\!\!-\!\!\rightarrow} \varphi$.

Remark 1. In the PSL standard [1], we also have atomic formulas of the form $\mathsf{ended}(\alpha)$ and $\mathsf{prev}(\alpha)$, where α is an SERE. For instance, the word w satisfies $\mathsf{ended}(\alpha)$ at position i iff there is a subword u of w that ends at i and $u \in L(\alpha)$. The operators ended and prev can be seen as restricted variants of the past operator $\mathrel{\Diamond\!\!\!\!\!-\!\!\rightarrow}$. For instance, in RTL, if $\varepsilon \notin L(\alpha)$, $\mathsf{ended}(\alpha)$ is syntactic sugar for $\alpha \mathrel{\Diamond\!\!\!\rightarrow} \mathsf{tt}$, and tt otherwise. Observe that ended and prev can only be applied to SEREs, and in contrast to $\mathrel{\Diamond\!\!\!\rightarrow}$, it is not possible to define the classical past operators Y, H, and O with them. We also remark that the literature, e.g. [5,9,17,24,7] usually considers the essential core of the PSL standard to which the operators ended and prev do not belong. We follow this convention, i.e., the formulas in our fragment PSL of RTL do not contain $\mathsf{ended}(\alpha)$ and $\mathsf{prev}(\alpha)$. Finally, we remark that the automata constructions [5,7] for PSL and SVA cannot cope with the operators ended and prev, which are handled by our construction in Section 4 for RTL.

Example 2. A standard example for showing that the past operators of PLTL can lead to more intuitive specifications is $\mathsf{G}(\mathit{grant} \rightarrow \mathsf{O}\mathit{request})$, i.e., every grant is preceded by a request [20]. An initially equivalent LTL formula is $\mathit{request}\ \mathsf{R}\ (\neg \mathit{grant} \vee \mathit{request})$. Let us now illustrate the beneficial use of SEREs and past operators. Suppose that a request is not a single event but a sequence of events, e.g., a request consists of a *start* event followed eventually by an *end* event and no *cancel* event happens between the *start* and the *end* event. Such sequences are naturally described by the SERE $(\mathit{start} \,;\, \mathsf{tt}^* \,;\, \mathit{end}) \cap (\neg \mathit{cancel})^*$. Using this SERE and the new past operator $\mathrel{\Diamond\!\!\!\rightarrow}$, we can easily express in RTL the property that every grant is preceded by a request:

$$\mathsf{G}\big(\mathit{grant} \rightarrow \big(((\mathit{start} \,;\, \mathsf{tt}^* \,;\, \mathit{end}) \cap (\neg \mathit{cancel})^*) \,;\, \mathsf{tt}^* \mathrel{\Diamond\!\!\!\rightarrow} \mathsf{tt}\big)\big)\,. \tag{1}$$

Note that according to the semantics of the operator $\mathrel{\Diamond\!\!\!\rightarrow}$, the *end* event has to happen before or at the same time as the *grant* event. Alternatively, we can express the property in PLTL as

$$\mathsf{G}\big(\mathit{grant} \rightarrow \mathsf{O}\big(\mathit{end} \wedge \neg \mathit{cancel} \wedge \mathsf{Y}(\neg \mathit{cancel}\ \mathsf{S}\ (\mathit{start} \wedge \neg \mathit{cancel}))\big)\big)\,. \tag{2}$$

Although debatable, we consider that the RTL formula (1) is easier to understand than the PLTL formula (2). In SVA, we can express the property as $\mathit{norequest} \mathrel{\Box\!\!\!\rightarrow} \neg \mathit{grant}$, where the SERE *norequest* describes the complement of the language $L\big(\mathsf{tt}^* \,;\, ((\mathit{start} \,;\, \mathsf{tt}^* \,;\, \mathit{end}) \cap (\neg \mathit{cancel})^*) \,;\, \mathsf{tt}^*\big)$, that is, $\mathit{norequest} := (a \cup b \,;\, d^* \,;\, c)^* \,;\, (c^* \cup b \,;\, d)$, where a, b, c, and d are the Boolean formulas $\neg \mathit{start} \vee \mathit{cancel}$, $\mathit{start} \wedge \neg \mathit{cancel}$, cancel, and $\neg \mathit{cancel} \wedge \neg \mathit{end}$, respectively. Note that in general, complementation of SEREs is difficult and can result in an exponential blowup with respect to the size of the given SERE.

Example 3. Let us give another example to illustrate the usefulness of past operators, in particular, the operator $\mathrel{\Diamond\!\!\!\rightarrow}$. For $N \geq 1$ and $i \in \{0, \dots, N-1\}$, consider the RTL formula $\Phi_{N,i} := \mathsf{G}\big(\mathit{send}_i \rightarrow \big(\mathit{switch}_i \cap (\mathit{init} \,;\, (\neg \mathit{init})^*) \mathrel{\Diamond\!\!\!\rightarrow} \mathsf{tt}\big)\big)$, where switch_i counts the number of *switch* events modulo N, i.e.,

$$\begin{aligned} \mathit{switch}_i := \big(&\underbrace{(\neg \mathit{switch})^* \,;\, \mathit{switch} \,;\, \dots \,;\, (\neg \mathit{switch})^* \,;\, \mathit{switch}}_{N \text{ times}}\big)^* \,; \\ &\underbrace{(\neg \mathit{switch})^* \,;\, \mathit{switch} \,;\, \dots \,;\, (\neg \mathit{switch})^* \,;\, \mathit{switch}}_{i \text{ times}} \,;\, (\neg \mathit{switch})^*\,. \end{aligned} \tag{3}$$

Intuitively, $\Phi_{N,i}$ expresses the property that the process i is only allowed to send a data item if it possesses the token. The process i possesses the token iff $i \equiv 0$ mod N *switch* events occurred previously since the last *init* event. Note that this property is not expressible in LTL since it is not star-free.

The negation of the PSL formula

$$\big((\neg init)^* \mathbin{\Diamond\!\!\rightarrow} send_i\big) \vee \mathsf{F}\big(init \wedge \big((\mathsf{tt}\,;(\neg init)^*) \cap (\textstyle\bigcup_{j \neq i} switch_j) \mathbin{\Diamond\!\!\rightarrow} send_i\big)\big) \quad (4)$$

is initially equivalent to $\Phi_{N,i}$. Note that the size of the formula (4) is quadratic in N, whereas the size of the formula (3) is only linear in N. In Section 5, we prove that PSVA is exponentially more succinct than PSL.

In general, for writing specifications, RTL possesses the advantage of PLTL over LTL and the advantage of PSL/SVA over LTL, namely, additional operators for referring to the past and SEREs for describing sequences of events.

4 From RTL to Nondeterministic Automata

In this section, we present a translation from RTL formulas into language-equivalent NBAs. Similar to the well-known translation for LTL formulas into NBAs, our translation comprises two steps: for a given RTL formula, we first construct an alternating automaton, which we then translate into an NBA. Throughout this section, we fix a finite set P of propositions.

4.1 From RTL to Loop-Free and Locally 1-Way 2ABAs

In this subsection, we assume that φ is an RTL formula over P and φ is in *negation normal form*, i.e., the negation symbol $\neg$ only occurs directly in front of the atomic subformulas of φ. Note that every RTL formula ψ can be rewritten into a logically equivalent RTL formula in negation normal form over an extended language, where we use the additional Boolean connective $\vee$ and the additional operators R, T, Z, $\mathbin{\Box\!\!\rightarrow}$, and $\mathbin{\boxminus\!\!\rightarrow}$ as primitives. The size of the resulting formula is at most $2\|\psi\|$. For rewriting a formula into negation normal form, we use the logical equivalences $\neg\neg\gamma \equiv \gamma$, $\neg\mathsf{X}\gamma \equiv \mathsf{X}\neg\gamma$, $\neg\mathsf{Y}\gamma \equiv \mathsf{Z}\neg\gamma$, and $\neg\mathsf{Z}\gamma \equiv \mathsf{Y}\neg\gamma$.

Due to space limitations, we do not provide the construction of the 2ABA $\mathcal{A}_\varphi$ for the RTL formula φ here. Instead, we only briefly highlight the similarities and the differences to the standard constructions for LTL, PLTL, SVA, and PSL [26,14,5,7]. The construction in [7] additionally handles SEREs with local variables. Our construction can easily be extended by this feature. However, for the ease of exposition, we focus here on how to handle the temporal past and future operators of RTL efficiently. As the standard construction for PSL [5], the state space of the 2ABA $\mathcal{A}_\varphi$ consists of the subformulas of the given RTL formula and the states of the automata for the SEREs. We introduce a special symbol # to mark the beginning of the input word. With this symbol, $\mathcal{A}_\varphi$ checks in a run whether the read-only head is at the first position of the input word. We need some auxiliary states for such a check. The new operators $\mathbin{\boxminus\!\!\rightarrow}$ and $\mathbin{\Diamond\!\!\!-\!\!\!\rightarrow}$ are then easily handled since $\mathcal{A}_\varphi$ is alternating and 2-way. From the construction, we obtain the following lemmas.

Lemma 4. *The 2ABA $\mathcal{A}_\varphi$ accepts the language $\{\#w \mid w \in L(\varphi)\}$.*

Lemma 5. *The 2ABA $\mathcal{A}_\varphi$ has size at most $4 + 2^{\|\varphi\|}$.*

The 2ABA $\mathcal{A}_\varphi$ has some additional properties, which we exploit in Section 4.2 for constructing the NBA. Namely, $\mathcal{A}_\varphi$ is loop-free [14,12] and locally 1-way.

Intuitively speaking, loop-freeness means that an automaton cannot visit a configuration twice on the same computation branch. Formally, it is defined as follows for a 2ABA $\mathcal{B} = (S, \Sigma, \eta, s_I, E)$. Let $\Pi(\mathcal{B})$ be the set of words of the form $(s_0, j_0)(s_1, j_1)\ldots \in (S \times \mathbb{N})^\omega$ such that $(s_0, j_0) = (s_I, 0)$ and for all $i \in \mathbb{N}$, there is some $a \in \Sigma$ and a set $M \subseteq S \times \mathbb{Z}$ with $(s_{i+1}, j_{i+1} - j_i) \in M$ and M is a minimal model of $\eta(s_i, a)$, i.e, $M \models \eta(s_i, a)$ and $M \setminus \{c\} \not\models \eta(s_i, a)$, for all $c \in M$. The automaton $\mathcal{B}$ is *loop-free* if for all words $\pi \in \Pi(\mathcal{B})$, there are no integers $i, j \in \mathbb{N}$ with $i \neq j$ such that $\pi_i = \pi_j$. Recall that π_i and π_j are configurations, which consist of the current state and the current position of the read-only head.

Lemma 6. *The 2ABA $\mathcal{A}_\varphi$ is loop-free.*

A 2ABA $\mathcal{B} = (S, \Sigma, \eta, s_I, E)$ is *locally 1-way* if $\eta(s, b) \in \mathcal{B}^+(S \times \{0, 1\}) \cup \mathcal{B}^+(S \times \{-1, 0\})$, for every $s \in S$ and $b \in \Sigma$. We remark that any 2ABA can be transformed into a language-equivalent 2ABA that is locally 1-way by doubling the state space. However, such a transformation is not needed for $\mathcal{A}_\varphi$, since A_φ is already constructed in such a way that it is locally 1-way.

Lemma 7. *The 2ABA $\mathcal{A}_\varphi$ is locally 1-way.*

4.2 From Loop-Free and Locally 1-Way 2ABAs to NBAs

In the following, we show how the alternating automaton from the previous subsection for an RTL formula in negation normal form can be translated into an NBA. The presented construction is based on an improvement of an alternation-elimination construction from [12]. Here, we additionally exploit the fact that the given 2ABA is locally 1-way. Overall, for an RTL formula ψ, the resulting language-equivalent NBA has size $\mathcal{O}(2^{3 \cdot 2^{2\|\psi\|}})$. With the construction in [12], we would obtain an NBA of size $\mathcal{O}(2^{4 \cdot 2^{2\|\psi\|}})$. Another advantage of the new construction is that it avoids the explicit representation of an extended alphabet, which is used in one of the intermediate construction steps in [12] and which is of exponential size. The presented construction also allows for a symbolic implementation [11], which can be used in tools like NuSMV [8] for satisfiability and finite-state model checking. See [6] for such implementations and an evaluation of constructions for the special case of 1-way alternating Büchi automata.

Theorem 8. *For a loop-free and locally 1-way 2ABA $\mathcal{A}$, there is a language-equivalent NBA $\mathcal{B}$ of size $\mathcal{O}(|\Sigma| \cdot 2^{2\|\mathcal{A}\|})$, where Σ is the alphabet of $\mathcal{A}$.*

The intuition for the construction of Theorem 8 is as follows. For an input word w, the NBA $\mathcal{B}$ guesses a run r of $\mathcal{A} = (Q, \Sigma, \delta, q_I, F)$ on w and checks whether this run is accepting. For this, as in [25,12], $\mathcal{B}$ represents r as a sequence of state sets $R_0 R_1 \ldots \in (2^Q)^\omega$, where each R_i contains the state q iff there is a path in

r that visits (q, i). In the case where $\mathcal{A}$ is 1-way, each R_i consists of the states that occur in the ith level of the run r. Note that in the general case where $\mathcal{A}$ is 2-way, R_i might contain states that occur in different levels of r. For instance, R_i contains the states q and q' from different levels if r contains a path of the form $(q_I, 0) \dots (q, i) \dots (q', i) \dots$. Since $\mathcal{A}$ is locally 1-way, we can locally check whether such a sequence $R_0 R_1 \dots$ represents a run of $\mathcal{A}$ on w. For doing so, $\mathcal{B}$ stores the set R_{i+1} and the letter w_{i+2} after reading the ith letter of w. For a state $q \in R_i$ with $\delta(q, w_i) \in \mathcal{B}^+(Q \times \{0, 1\})$, the set $(R_i \times \{0\}) \cup (R_{i+1} \times \{1\})$ must be a model of $\delta(q, w_i)$. $\mathcal{B}$ checks this when reading the letter w_i. For $\delta(q, w_i) \in \mathcal{B}^+(Q \times \{-1, 0\})$ and $i > 0$, $(R_{i-1} \times \{-1\}) \cup (R_i \times \{0\})$ must be a model of $\delta(q, w_i)$. $\mathcal{B}$ already checks this when it reads the $(i-1)$th input letter by using the guessed letter w_i. Additionally, $\mathcal{B}$ must check that every path in r visits configurations with an accepting state infinitely often. Since $\mathcal{A}$ is loop-free the run r is accepting iff there are indexes $i_0 < i_1 < \dots$ such that each path in r that visits a configuration (q, i_j) visits a configuration with an accepting state before visiting (q', i_{j+1}), for every $j \in \mathbb{N}$. Similar to the alternation-elimination construction by Miyano and Hayashi [22] for 1-way alternating Büchi automata, $\mathcal{B}$ checks this property with an additional component in the state space and its set of accepting states.

We obtain the following result by putting the two constructions together.

Theorem 9. *For any RTL formula ψ, there is a language-equivalent NBA $\mathcal{C}$ of size $\mathcal{O}(2^{3 \cdot 2^{2\|\psi\|}})$.*

Proof. First, we transform ψ into a logically equivalent formula ψ' that is in negation normal of size $2\|\psi\|$. Let $\mathcal{A}_{\psi'}$ be the 2ABA that we obtain from ψ' by the construction in Section 4.1. By the Lemmas 5, 6, and 7, $\mathcal{A}_{\psi'}$ is loop-free, locally 1-way, and $\|\mathcal{A}_{\psi'}\| \leq 4 + 2^{2\|\psi\|}$. By Lemma 4, $\mathcal{A}_{\psi'}$ accepts the language $\{\#w \mid w \in L(\psi)\}$. By Theorem 8, we translate $\mathcal{A}_{\psi'}$ into a language-equivalent NBA $\mathcal{B}$ with $\mathcal{O}(2^{3 \cdot 2^{2\|\psi\|}})$ states. From $\mathcal{B}$, it is easy to obtain an NBA $\mathcal{C}$ with $L(\mathcal{C}) = L(\psi)$ and $\|\mathcal{C}\| \in \mathcal{O}(2^{3 \cdot 2^{2\|\psi\|}})$. □

We remark that the upper bound of the NBA in Theorem 9 can be improved by taking the number of distinct subformulas into account instead of the syntactic length of the given RTL formula. We omit such a refined analysis here.

4.3 Consequences of the Translation

We conclude this section by proving some facts that follow from Theorem 9.

Since SVA can already express all ω-regular languages, we have that RTL describes exactly the ω-regular languages. Moreover, SVA, PSL, and RTL share the same computational complexity. In particular, the satisfiability and the model-checking problem for RTL are EXPSPACE-complete in general and PSPACE-complete for RTL formulas with a bounded number of intersection operators. Another similarity between the logics is that they all have the small model property of doubly exponential size. In particular, there is a constant $c > 0$ such that a satisfiable RTL formula φ has a model of the form uv^ω with $|uv| \leq c \cdot 2^{3 \cdot 2^{2\|\varphi\|}}$.

Since PSL/SVA and RTL describe the same class of properties, the question arises of their relative succinctness. The next theorem states an upper bound on the translation from RTL to SVA. Roughly speaking, for the proof, we translate an RTL formula into an NBA and then into an ω-regular expression, which we finally translate into an SVA formula.

Theorem 10. *For any RTL formula φ, there is an initially equivalent SVA formula of size $2^{\mathcal{O}(2^{2^{\|\varphi\|+2}})}$ and in which the intersection operator does not occur.*

It is fair to ask whether the upper bound in Theorem 10 is optimal, i.e., whether there is a family of RTL formulas such that every initially equivalent family of PSL formulas must be triply exponentially larger. The result on the small model property shows that such a lower bound cannot be proved by comparing the model sizes (see, e.g., the Gap Lemma in [18]). We were only able to establish an exponential lower bound. This result is presented in the next section.

5 Succinctness Gaps

In this section, we prove an exponential succinctness gap between RTL and PSL/SVA, i.e., there is a family $(\varPhi_n)_{n>0}$ of RTL formulas such that for every family $(\varPsi_n)_{n>0}$ of PSL or SVA formulas, if $\varPsi_n$ is initially equivalent to $\varPhi_n$ for all $n > 0$, then $\|\varPsi_n\|$ is exponential in $\|\varPhi_n\|$. In fact, our result is stronger since the formulas $\varPhi_n$ that we define are just PSVA formulas. The proof of this succinctness result can easily be adapted to show that PSVA and, hence, RTL, is double exponentially more succinct than PLTL.

Our proof for the succinctness gap between PSVA and SVA has a similar flavor as the proof in [21], which shows that PLTL is exponentially more succinct than LTL. However, our proof is more involved since we must take SEREs into account. In fact, the formulas in the family of PLTL formulas that is used in [21] are initially equivalent to SVA formulas of linear size. From this observation, we conclude that SVA is exponentially more succinct than LTL.

Lemma 11. *For every $n > 0$, there is an SVA formula $\varTheta_n$ such that for any LTL formula $\varXi_n$, if $L(\varXi_n) = L(\varTheta_n)$ then $\|\varXi_n\| \in \varOmega(2^{\|\varTheta_n\|})$.*

Let us now turn to the succinctness gap between PSVA and SVA. For this, we first introduce so-called n-counting words, which can be defined in SVA by formulas of size $\mathcal{O}(n)$. In the following, let $n > 0$, P_n be the set $\{c_0, \dots, c_{n-1}, p, q\}$ of propositions, and $\varSigma_n$ the alphabet 2^{P_n}. The *n-value* of the letter $b \in \varSigma_n$ is $val_n(b) := \sum_{0 \leq i < n} 2^{c'_i}$ with $c'_i := 1$ if $c_i \in b$ and $c'_i := 0$, otherwise. In other words, the n-value of b is obtained by reading $c_0, \dots, c_{n-1}$ as bits of a positive integer in binary representation. A word $w \in \varSigma_n^\omega$ is *n-counting* if $val_n(w_0) = 0$ and $val_n(w_{i+1}) \equiv val_n(w_i) + 1 \mod 2^n$, for all $i \in \mathbb{N}$.

Lemma 12. *For every $n > 0$, there is an SVA formula $count_n$ of size $\mathcal{O}(n)$ such that $L(count_n) \subseteq \varSigma_n^\omega$ is the language of n-counting words.*

An *n-segment* of a word $w \in \varSigma_n^\omega$ is a subword $v = w_i \dots w_{i+2^n-1}$ such that $i \equiv 0 \mod 2^n$, for some $i \in \mathbb{N}$. The n-segment v is *initial* if $i = 0$. For a proposition

$r \in P$, the words $u, v \in \Sigma_n^*$ are *r-equal* if $|u| = |v|$ and $r \in u_i \Leftrightarrow r \in v_i$, for all $i \in \mathbb{N}$ with $i < |v|$. Let L_n and L'_n be the following languages:

- L_n consists of the n-counting words $w \in \Sigma_n^\omega$ such that if an n-segment of w is p-equal to the initial n-segment w then they are also q-equal.
- L'_n consists of the n-counting words $w \in \Sigma_n^\omega$ such that if the n-segments u and v of w are p-equal then they are also q-equal.

Lemma 13. *For every $n > 0$, there is a PSVA formula Φ_n of size $\mathcal{O}(n)$ such that $L(\Phi_n) = L_n$.*

Lemma 14. *For every $n > 0$, if $\mathcal{B}$ is an NBA with $L(\mathcal{B}) = L'_n$ then $\|\mathcal{B}\| \geq 2^{2^{2^n}}$.*

With the above lemmas we obtain our succinctness result for PSVA and SVA.

Theorem 15. *For every $n > 0$, there is a PSVA formula Φ_n such that $L(\Phi_n) = L_n$ and for every SVA formula Ψ_n, if $L(\Psi_n) = L_n$ then $\|\Psi_n\| \in \Omega(2^{\|\Phi_n\|})$.*

Proof. For a given $n > 0$, take the PSVA formula Φ_n from Lemma 13. Suppose that Ψ_n is an SVA formula that is initially equivalent to Φ_n. Let $\Psi'_n := \mathit{count}_n \wedge \mathsf{G}(\neg c_0 \wedge \cdots \wedge \neg c_{n-1} \rightarrow \Psi_n)$. Note that Ψ'_n expresses that a model is n-counting and each two p-equal n-segments in a model are also q-equal, i.e., $L(\Psi'_n) = L'_n$. By Theorem 9, there is an NBA $\mathcal{B}$ of size $2^{2^{\mathcal{O}(\|\Psi'_n\|)}}$ and $L(\mathcal{B}) = L(\Psi'_n)$. By Lemma 14, we have that $\|\mathcal{B}\| \geq 2^{2^{2^n}}$. It follows that $\|\Psi'_n\| \in \Omega(2^{\|\Phi_n\|})$. Since Ψ'_n is linear in the size of Ψ_n, we conclude that $\|\Psi_n\| \in \Omega(2^{\|\Phi_n\|})$. □

Note that L_n is a star-free language, i.e., there is an LTL formula φ_n such that $L(\varphi_n) = L_n$. We can easily adapt the proof of Theorem 15 to obtain a double exponential succinctness gap between PSVA and PLTL.

Corollary 16. *For every $n > 0$, there is a PSVA formula Φ_n such that $L(\Phi_n) = L_n$ and for any PLTL formula Ξ_n, if $L(\Xi_n) = L_n$ then $\|\Xi_n\| \in \Omega(2^{2^{\|\Phi_n\|}})$.*

Remark 17. We conclude this section by stating some open problems related to the presented succinctness gaps. First, it remains open whether the exponential succinctness gap still holds between RTL and extensions of PSL/SVA with restricted variants of the past operators like the ones discussed in Remark 1. We did not succeeded in proving such a gap, neither did we succeed in expressing the languages L_n concisely in such an extension. Second, it remains open whether the succinctness gaps carry over to a fixed and finite proposition set. Note that the proposition sets P_n over which the PSVA formulas Φ_n are defined grow linearly in n. As shown in [13], we can encode any number of propositions by a single proposition. However, the sizes of the adapted formulas for Φ_n are no longer linear in n. In particular, the sizes of the adapted SEREs in Lemma 13 are quadratic in n. It is not obvious how to adapt these SEREs so that their sizes remain linear in n. Therefore, for a fixed and finite proposition set, we only obtain a superpolynomial succinctness gap between PSVA and SVA. Note that for similar reasons, the adapted proof of the succinctness gap between PLTL and LTL in [21,19] for a fixed and finite proposition set also only shows that PLTL is superpolynomially more succinct than LTL.

6 Conclusion

In this paper, we have proposed the temporal logic RTL, which extends PSL and SVA with past operators. We have analyzed its complexity and our results show that RTL and PSL/SVA are similarly related as PLTL and LTL with respect to expressiveness, succinctness, and the computational complexities of the satisfiability and the model-checking problem. It remains to be seen whether the advantages of RTL over PSL and SVA pay off in practice. The presented translation for RTL into NBAs shows that the additional cost for handling past operators is small and should not be a burden in implementing RTL in system verification. Our preliminary experience with a prototype implementation for the model checker NuSMV are promising.[2]

References

1. IEEE standard for Property Specification Language (PSL). IEEE Std 1850TM (October 2005)
2. IEEE standard for SystemVerilog—unified hardware design, specification, and verification language. IEEE Std 1800TM (November 2005)
3. Armoni, R., Fix, L., Flaisher, A., Gerth, R., Ginsburg, B., Kanza, T., Landver, A., Mador-Haim, S., Singerman, E., Tiemeyer, A., Vardi, M.Y., Zbar, Y.: The ForSpec temporal logic: A new temporal property-specification language. In: Katoen, J.-P., Stevens, P. (eds.) TACAS 2002. LNCS, vol. 2280, pp. 296–311. Springer, Heidelberg (2002)
4. Banieqbal, B., Barringer, H.: Temporal logic with fixed points. In: Banieqbal, B., Pnueli, A., Barringer, H. (eds.) Temporal Logic in Specification. LNCS, vol. 398, pp. 62–74. Springer, Heidelberg (1989)
5. Ben-David, S., Bloem, R., Fisman, D., Griesmayer, A., Pill, I., Ruah, S.: Automata construction algorithms optimized for PSL. Technical report, The Prosyd Project (2005), http://www.prosyd.org
6. Bloem, R., Cimatti, A., Pill, I., Roveri, M.: Symbolic implementation of alternating automata. Int. J. Found. Comput. Sci. 18(4), 727–743 (2007)
7. Bustan, D., Havlicek, J.: Some complexity results for SytemVerilog assertions. In: Ball, T., Jones, R.B. (eds.) CAV 2006. LNCS, vol. 4144, pp. 205–218. Springer, Heidelberg (2006)
8. Cimatti, A., Clarke, E.M., Giunchiglia, E., Giunchiglia, F., Pistore, M., Roveri, M., Sebastiani, R., Tacchella, A.: NuSMV 2: An opensource tool for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002, vol. 2404, pp. 359–364. Springer, Heidelberg (2002)
9. Cimatti, A., Roveri, M., Semprini, S., Tonetta, S.: From PSL to NBA: A modular symbolic encoding. In: FMCAD 2006, pp. 125–133. IEEE Computer Society Press, Los Alamitos (2006)
10. Cimatti, A., Roveri, M., Sheridan, D.: Bounded verification of Past LTL. In: Hu, A.J., Martin, A.K. (eds.) FMCAD 2004. LNCS, vol. 3312, pp. 245–259. Springer, Heidelberg (2004)
11. Clarke, E.M., Grumberg, O., Hamaguchi, K.: Another look at LTL model checking. Form. Method. Syst. Des. 10(1), 47–71 (1997)

[2] See www.inf.ethz.ch/~daxc/rtl2ba for the most recent version of our tool.

12. Dax, C., Klaedtke, F.: Alternation elimination by complementation. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS, vol. 5330, pp. 214–229. Springer, Heidelberg (2008)
13. Demri, S., Schnoebelen, P.: The complexity of propositional linear temporal logics in simple cases. Inf. Comput. 174(1), 84–103 (2002)
14. Gastin, P., Oddoux, D.: LTL with past and two-way very-weak alternating automata. In: Rovan, B., Vojtáš, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 439–448. Springer, Heidelberg (2003)
15. Harel, D., Kozen, D., Tiuryn, J.: Dynamic Logic. MIT Press, Cambridge (2000)
16. Kupferman, O., Piterman, N., Vardi, M.Y.: Extended temporal logic revisited. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 519–535. Springer, Heidelberg (2001)
17. Lange, M.: Linear time logics around PSL: Complexity, expressiveness, and a little bit of succinctness. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 90–104. Springer, Heidelberg (2007)
18. Lange, M.: A purely model-theoretic proof of the exponential succinctness gap between CTL$^+$ and CTL. Inform. Process. Lett. 108(5), 308–312 (2008)
19. Laroussinie, F., Markey, N., Schnoebelen, P.: Temporal logic with forgettable past. In: LICS 2002, pp. 383–392. IEEE Computer Society Press, Los Alamitos (2002)
20. Lichtenstein, O., Pnueli, A., Zuck, L.D.: The glory of the past. In: Parikh, R. (ed.) Logic of Programs 1985. LNCS, vol. 193, pp. 196–218. Springer, Heidelberg (1985)
21. Markey, N.: Temporal logic with past is exponentially more succinct. Bulletin of the EATCS 79, 122–128 (2003)
22. Miyano, S., Hayashi, T.: Alternating finite automata on ω-words. Theoret. Comput. Sci. 32(3), 321–330 (1984)
23. Pnueli, A.: The temporal logic of programs. In: FOCS 1977, pp. 46–57. IEEE Computer Society Press, Los Alamitos (1977)
24. Pnueli, A., Zaks, A.: PSL model checking and run-time verification via testers. In: Misra, J., Nipkow, T., Sekerinski, E. (eds.) FM 2006. LNCS, vol. 4085, pp. 573–586. Springer, Heidelberg (2006)
25. Vardi, M.Y.: A note on the reduction of two-way automata to one-way automata. Inform. Process. Lett. 30(5), 261–264 (1989)
26. Vardi, M.Y.: An automata-theoretic approach to linear temporal logic. In: Moller, F., Birtwistle, G. (eds.) Logics for Concurrency. LNCS, vol. 1043, pp. 238–266. Springer, Heidelberg (1996)
27. Vardi, M.Y., Wolper, P.: An automata-theoretic approach to automatic program verification. In: LICS 1986, pp. 332–344. IEEE Computer Society Press, Los Alamitos (1986)
28. Wolper, P.: Temporal logic can be more expressive. Information and Control 56(1/2), 72–99 (1983)

Forward Analysis for WSTS, Part II: Complete WSTS

Alain Finkel[1] and Jean Goubault-Larrecq[1,2]

[1] LSV, ENS Cachan, CNRS, France
finkel@lsv.ens-cachan.fr
[2] INRIA Saclay, France
goubault@lsv.ens-cachan.fr

Abstract. We describe a simple, conceptual forward analysis procedure for ∞-complete WSTS $\mathfrak{S}$. This computes the *clover* of a state s_0, i.e., a finite description of the closure of the cover of s_0. When $\mathfrak{S}$ is the completion of a WSTS $\mathfrak{X}$, the clover in $\mathfrak{S}$ is a finite description of the cover in $\mathfrak{X}$. We show that this applies exactly when $\mathfrak{X}$ is an ω^2-*WSTS*, a new robust class of WSTS. We show that our procedure terminates in more cases than the generalized Karp-Miller procedure on extensions of Petri nets. We characterize the WSTS where our procedure terminates as those that are *clover-flattable*. Finally, we apply this to well-structured counter systems.

1 Introduction

Context. Well-structured transition systems (WSTS) are a general class of infinite-state systems where coverability—given states s, t, decide whether $s\ (\geq; \to^*; \geq)\ t$, i.e., whether $s \geq s_1 \to^* t_1 \geq t$ for some s_1, t_1—is decidable, using a simple backward algorithm [14,15,19,2].

The starting point of this paper and of its first part [17] is our desire to derive similar *forward* algorithms, namely algorithms computing the *cover* $\downarrow Post^*(\downarrow s)$ of s. While the cover allows one to decide coverability as well, by testing whether $t \in \downarrow Post^*(\downarrow s)$, it can also be used to decide U-boundedness, i.e., to decide whether there are only finitely many states t in the upward-closed set U and such that $s\ (\geq; \to^*)\ t$. No backward algorithm can decide this. In fact, U-boundedness is undecidable in general, e.g., on lossy channel systems [9]. So the reader should be warned that computing the cover is not possible for general WSTS. Despite this, the known forward algorithms are felt to be more efficient than backward procedures in general: e.g., for lossy channel systems, although the backward procedure always terminates, only the non-terminating forward procedure is implemented in the tool TREX [1].

State of the art. Karp and Miller [27] proposed an algorithm, for Petri nets, which computes a finite representation of the *cover*, i.e., of the downward closure of the reachability set of a Petri net. Finkel [14,15] introduced the WSTS framework and generalized the Karp-Miller procedure to a class of WSTS. This was achieved by building a non-effective completion of the set of states, and replacing ω-accelerations of increasing sequences of states (in Petri nets) by least upper bounds (lub). In [12,15] a variant of this generalization of the Karp-Miller procedure was studied; but no guarantee was given

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 188–199, 2009.

that the cover could be represented finitely. There were no effective finite representations of downward closed sets in [15]. Finkel [16] modified the Karp-Miller algorithm to reduce the size of the intermediate computed trees. [22] recently proposed a weaker acceleration, which avoid some possible underapproximations in [16]. Emerson and Namjoshi [12] took into account the labeling of WSTS for adapting the generalised Karp-Miller algorithm to model-checking. They assume the existence of a compatible cpo, and proved that for broadcast protocols (which are equivalent to transfer Petri nets), the Karp-Miller procedure can be generalized. However, termination is then not guaranteed [13], and in fact neither is the existence of a finite representation of the cover. Abdulla, Colomb-Annichini, Bouajjani and Jonsson proposed a forward procedure for lossy channel systems [3] using downward closed regular languages as symbolic representations. Ganty, Geeraerts, Raskin and Van Begin [21,20] proposed a forward procedure for solving the coverability problem for WSTS equipped with an effective adequate domain of limits, or equipped with a finite set D used as a parameter to tune the precision of an abstract domain. Both solutions insure that every downward closed set has a finite representation. Abdulla *et al.* [3] applied this framework to Petri nets and lossy channel systems. Abdulla, Deneux, Mahata and Nylén proposed a symbolic framework for dealing with downward closed sets for Timed Petri nets [4].

Our contribution. First, we define *complete WSTS* as WSTS whose well-ordering is also a continuous dcpo. This allows us to design a conceptual procedure $\mathbf{Clover}_{\mathfrak{S}}$ that looks for a finite representation of the downward closure of the reachability set, i.e., of the cover [15]. We call such a finite representation a *clover* (for *clo*sure of *cover*). This clearly separates the fundamental ideas from the data structures used in implementing Karp-Miller-like algorithms. Our procedure also terminates in more cases than the well-known (generalized) Karp-Miller procedure [12,15]. We establish the main properties of clovers in Section 3 and use them to prove $\mathbf{Clover}_{\mathfrak{S}}$ correct, notably, in Section 5.

Second, we characterize complete WSTS for which $\mathbf{Clover}_{\mathfrak{S}}$ terminates. These are the ones that have a (continuous) flattening with the same clover. This establishes a surprising relationship with the theory of flattening [8].

Third, and building on our theory of completions [17], we characterize those WSTS whose completion is a complete WSTS in the sense above. They are exactly the ω^2-*WSTS*, i.e., those whose state space is ω^2-wqo, as we show in Section 4.

Finally, we apply our framework of complete WSTS to counter systems in Section 6. We show that affine counter systems may be completed into ∞-complete WSTS iff the domains of the monotone affine functions are upward closed.

2 Preliminaries

Posets, dcpos. We borrow from theories of order, as used in model-checking [2,19], and also from domain theory [6,23]. A *quasi-ordering* $\leq$ is a reflexive and transitive relation on a set X. It is a (partial) *ordering* iff it is antisymmetric.

We write $\geq$ the converse quasi-ordering, $<$ the associated strict ordering $(\leq \setminus \geq)$, and $>$ the converse $(\geq \setminus \leq)$ of $<$. A set X with a partial ordering $\leq$ is a *poset* $(X, \leq)$, or just X when $\leq$ is clear. The *upward closure* $\uparrow E$ of a set E is $\{y \in X \mid \exists x \in E \cdot x \leq y\}$. The *downward closure* $\downarrow E$ is $\{y \in X \mid \exists x \in E \cdot y \leq x\}$. A subset E of X is *upward closed* if and only if $E = \uparrow E$. *Downward closed* sets are defined similarly. A downward closed (resp. upward closed) set E has a *basis* A iff $E = \downarrow A$ (resp. $E = \uparrow A$); E has a *finite basis* iff A can be chosen finite.

A quasi-ordering is *well-founded* iff it has no infinite strictly descending chain $x_0 > x_1 > \ldots > x_i > \ldots$. An *antichain* is a set of pairwise incomparable elements. A quasi-ordering is *well* iff it is well-founded and has no infinite antichain. We abbreviate well posets as *wpos*.

An *upper bound* $x \in X$ of $E \subseteq X$ is such that $y \leq x$ for every $y \in E$. The *least upper bound (lub)* of a set E, if it exists, is written $lub(E)$. An element x of E is *maximal* (resp. minimal) iff $\uparrow x \cap E = \{x\}$ (resp. $\downarrow x \cap E = \{x\}$). Write $\mathrm{Max}\, E$ (resp. $MinE$) the set of maximal (resp. minimal) elements of E.

A *directed subset* of X is any non-empty subset D such that every pair of elements of D has an upper bound in D. Chains, i.e., totally ordered subsets, and one-element set are examples of directed subsets. A *dcpo* is a poset in which every directed subset has a least upper bound. For any subset E of a dcpo X, let $\mathrm{Lub}(E) = \{lub(D) \mid D \text{ directed subset of } E\}$. Clearly, $E \subseteq \mathrm{Lub}(E)$; $\mathrm{Lub}(E)$ can be thought of E plus all limits from elements of E.

The *way below* relation $\ll$ on a dcpo X is defined by $x \ll y$ iff, for every directed subset D such that $lub(D) \leq y$, there is a $z \in D$ such that $x \leq z$. Write $\Downarrow E = \{y \in X \mid \exists x \in E \cdot y \ll x\}$. X is *continuous* iff, for every $x \in X$, $\Downarrow x$ is a directed subset, and has x as least upper bound.

When $\leq$ is a well partial ordering that also turns X into a dcpo, we say that X is a *directed complete well order*, or *dcwo*. If additionally X is continuous, we say that X is a *cdcwo*.

A subset U of a dcpo X is (Scott-)*open* iff U is upward-closed, and for any directed subset D of X such that $lub(D) \in U$, some element of D is already in U. A map $f : X \to X$ is (Scott-)*continuous* iff f is monotonic ($x \leq y$ implies $f(x) \leq f(y)$) and for every directed subset D of X, $lub(f(D)) = f(lub(D))$. Equivalently, f is continuous in the topological sense, i.e., $f^{-1}(U)$ is open for every open U.

A *closed* set is the complement of an open set. Every closed set is downward closed. The *closure* $cl(A)$ of $A \subseteq X$ is the smallest closed set containing A. This should not be confused with the *inductive closure* $\mathrm{Ind}(A)$ of A, which is obtained as the least set B containing A and such that $\mathrm{Lub}(B) = B$. In general, $\downarrow A \subseteq \mathrm{Lub}(\downarrow A) \subseteq \mathrm{Ind}(\downarrow A) \subseteq cl(A)$, and all inclusions can be strict. However, when X is a *continuous* dcpo, and A is downward closed in X, $\mathrm{Lub}(A) = \mathrm{Ind}(A) = cl(A)$. (See, e.g., [17, Proposition 3.5].)

Well-Structured Transition Systems. A *transition system* is a pair $\mathfrak{S} = (S, \to)$ of a set S, whose elements are called *states*, and a *transition relation* $\to \subseteq S \times S$. We write $s \to s'$ for $(s, s') \in \to$. Let $\xrightarrow{*}$ be the transitive and reflexive closure of the relation $\to$. We write $Post_{\mathfrak{S}}(s) = \{s' \in S \mid s \to s'\}$ for the set of immediate successors of the state s. The *reachability set* of a transition system $\mathfrak{S} = (S, \to)$ from an initial state s_0 is $Post^*_{\mathfrak{S}}(s_0) = \{s \in S \mid s_0 \xrightarrow{*} s\}$.

A transition system $(S, \to)$ is *effective* iff S is r.e., and for every state s, $Post_{\mathfrak{S}}(s)$ is finite and computable. An *ordered* transition system is a triple $\mathfrak{S} = (S, \to, \leq)$ where $(S, \to)$ is a transition system and $\leq$ is a quasi-ordering on S. We say that $(S, \to, \leq)$ is *effective* if $(S, \to)$ is effective and if $\leq$ is decidable.

$\mathfrak{S} = (S, \rightarrow, \leq)$ is *monotone* (resp. *strictly monotone*) iff for every $s, s', s_1 \in S$ such that $s \rightarrow s'$ and $s_1 \geq s$ (resp. $s_1 > s$), there exists an $s'_1 \in S$ such that $s_1 \xrightarrow{*} s'_1$ and $s'_1 \geq s'$ (resp. $s'_1 > s'$). $\mathfrak{S}$ is *strongly monotone* iff for every $s, s', s_1 \in S$ such that $s \rightarrow s'$ and $s_1 \geq s$, there exists an $s'_1 \in S$ such that $s_1 \rightarrow s'_1$ and $s'_1 \geq s'$.

Finite representations of $Post_{\mathfrak{S}}(s)$,e.g., as Presburger formulae or finite automata, usually don't exist even for monotone transition systems (not even speaking of being computable). The *cover* $Cover_{\mathfrak{S}}(s) = \downarrow Post^*_{\mathfrak{S}}(\downarrow s)$ ($= \downarrow Post^*_{\mathfrak{S}}(s)$ when $\mathfrak{S}$ is monotone) is better behaved. Note that being able to compute the cover allows one to decide *coverability*: $s\ (\geq; \rightarrow^*; \geq)\ t$ iff $t \in Cover_{\mathfrak{S}}(s)$. In most cases we shall encounter, it will also be decidable whether a finitely represented cover is finite, or whether it meets a given upward closed set U in only finitely many points. Therefore *boundedness* (is $Post^*_{\mathfrak{S}}(s)$ finite?) and U*-boundedness* (is $Post^*_{\mathfrak{S}}(s) \cap U$ finite?) will be decidable, too.

An ordered transition system $\mathfrak{S} = (S, \rightarrow, \leq)$ is a *Well Structured Transition System* (*WSTS*) iff $\mathfrak{S}$ is monotone and $(S, \leq)$ is wpo. This is our object of study.

For strictly monotone WSTS, it is also possible to decide the boundedness problem, with the help of the Finite Reachability Tree (FRT) [15]. However, the U-Boundedness problem (called the place-boundedness problem for Petri nets) remains undecidable for strictly monotone WSTS (for instance, for transfer Petri nets), but it is decidable for Petri nets. It is decided with the help of a richer structure than the FRT, the Karp-Miller tree. The set of labels of the Karp-Miller tree is a finite representation of the cover.

We will consider transition systems defined by a finite set of transition functions for simplicity. This is as in [17]. Formally, a *functional transition system* $(S, \xrightarrow{F})$ is a labeled transition system where the transition relation $\xrightarrow{F}$ is defined by a finite set F of partial functions $f : S \longrightarrow S$, in the sense that for every $s, s' \in S$, $s \xrightarrow{F} s'$ iff $s' = f(s)$ for some $f \in F$. A map $f : S \rightarrow S$ is *partial monotonic* iff $\operatorname{dom} f$ is upward-closed and for all $x, y \in \operatorname{dom} f$ with $x \leq y$, $f(x) \leq f(y)$. An *ordered functional transition system* is an ordered transition system $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ where F consists of partial monotonic functions. This is always strongly monotonic. A *functional WSTS* is an ordered functional transition system where $\leq$ is well.

A functional transition system $(S, \xrightarrow{F})$ is *effective* if every $f \in F$ is computable: given a state s and a function f, we may decide whether $s \in \operatorname{dom} f$ and in this case, one may also compute $f(s)$.

3 Clovers of Complete WSTS

Complete WSTS and their clovers. All forward procedures for WSTS rest on completing the given WSTS to one that includes all limits. E.g., the state space of Petri nets is $\mathbb{N}^k$, the set of all markings on k places, but the Karp-Miller algorithm works on $\mathbb{N}^k_\omega$, where $\mathbb{N}_\omega$ is $\mathbb{N}$ plus a new top element ω. We have defined general completions of wpos, serving as state spaces, and have briefly described completions of (functional) WSTS in [17]. We temporarily abstract away from this, and consider *complete* WSTS directly.

Generalizing the notion of continuity to partial maps, a *partial continuous* map $f : X \rightarrow X$, where $(X, \leq)$ is a dcpo, is such that $\operatorname{dom} f$ is open (not just upward-closed), and for every directed subset D *in* $\operatorname{dom} f$, $lub(f(D)) = f(lub(D))$.

Equivalently, $\operatorname{dom} f$ is open and $f^{-1}(U)$ is open for any open U. The composite of two partial continuous maps is again partial continuous.

Definition 1. *A* complete *WSTS is a (functional) WSTS* $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ *where* $(S, \leq)$ *is a cdcwo and every function in* F *is partial continuous.*

The point in complete WSTS is that one can *accelerate* loops:

Definition 2. *Let* $(X, \leq)$ *be a dcpo,* $f : X \to X$ *be partial continuous. The* lub-acceleration $f^\infty : X \to X$ *is defined by:* $\operatorname{dom} f^\infty = \operatorname{dom} f$*, and for any* $x \in \operatorname{dom} f$*, if* $x < f(x)$ *then* $f^\infty(x) = lub\{f^n(x) \mid n \in \mathbb{N}\}$*, else* $f^\infty(x) = f(x)$*.*

Note that if $x \leq f(x)$, then $f(x) \in \operatorname{dom} f$, and $f(x) \leq f^2(x)$. By induction, we can show that $\{f^n(x) \mid n \in \mathbb{N}\}$ is an increasing sequence, so that the definition makes sense.

Complete WSTS are strongly monotone. One may not decide, in general, whether a recursive function f is monotone [18] or continuous, whether an ordered set $(S, \leq)$ with a decidable ordering $\leq$, is a dcpo or whether it is a wpo. We may prove that given an effective ordered functional transition system, one cannot decide whether it is a WSTS, or a complete WSTS. However, the completion of *any* functional ω^2-WSTS is complete, as we shall see in Theorem 1.

In a complete WSTS, there is a *canonical* finite representation of the cover:

Definition 3 (Clover). *Let* $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ *be a complete WSTS. The* clover $Clover_{\mathfrak{S}}(s_0)$ *of the state* $s_0 \in S$ *is* $\operatorname{Max}\operatorname{Lub}(Cover_{\mathfrak{S}}(s_0))$*.*

Proposition 1. *Let* $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ *be a complete WSTS, and* $s_0 \in S$*. Then* $Clover_{\mathfrak{S}}(s_0)$ *is finite, and* $cl(Cover_{\mathfrak{S}}(s_0)) = \downarrow Clover_{\mathfrak{S}}(s_0)$*.*

Proof. $\operatorname{Lub}(Cover_{\mathfrak{S}}(s_0)) = cl(Cover_{\mathfrak{S}}(s_0))$ since $Cover_{\mathfrak{S}}(s_0)$ is downward closed, and S is a continuous dcpo. Since S is a wpo, it is Noetherian in its Scott topology [25, Proposition 3.1]. Since S is a continuous dcpo, S is also sober [6, Proposition 7.2.27], so Corollary 6.5 of [25] applies: every closed subset F of S is such that $\operatorname{Max} F$ is finite and $F = \downarrow \operatorname{Max} F$. Now let $F = \operatorname{Lub}(Cover_{\mathfrak{S}}(s_0))$. □

For any other representative, i.e., for any finite set R such that $\downarrow R = \downarrow Clover_{\mathfrak{S}}(s_0)$, $Clover_{\mathfrak{S}}(s_0) = \operatorname{Max} R$. Indeed, for any two finite sets $F, G \subseteq S$ such that $\downarrow F = \downarrow G$, $\operatorname{Max} F = \operatorname{Max} G$. So *Clover* is the *minimal representative* of the cover, i.e., there is no representative R with $|R| < |Clover_{\mathfrak{S}}(s_0)|$. The clover was called the minimal coverability set in [16].

Despite the fact that the clover is always finite, it is non-computable in general (see Proposition 4 below). Nonetheless, it is computable on *flat* complete WSTS, and even on the larger class of *clover-flattable* complete WSTS (Theorem 3 below).

Completions. There are numberous WSTS which are not complete: the set $\mathbb{N}^k$ of states of a Petri net with k places is not even a dcpo. The set of states of a lossy channel system with k channels, $(\Sigma^*)^k$, is not a dcpo for the subword ordering either. We have defined general completions of wpos, and of WSTS, in [17], which we recall quickly.

The *completion* $\widehat{X}$ of a wpo $(X, \leq)$ is defined in any of two equivalent ways. First, $\widehat{X}$ is the *ideal completion* $Idl(X)$ of X, i.e., the set of ideals of X, ordered by inclusion, where an *ideal* is a downward-closed directed subset of X. This can also be described as the sobrification $\mathcal{S}(X_a)$ of the Noetherian space X_a, but this is probably harder to understand (although it makes proofs simpler). We consider X as a subset of $\widehat{X}$, by equating each element $x \in X$ with $\downarrow x \in \mathit{Idl}(X)$. For instance, if $X = \mathbb{N}^k$, e.g., with $k = 3$, then $(1, 3, 2)$ is equated with the ideal $\downarrow(1, 3, 2)$, while $\{(1, m, n) \mid m, n \in \mathbb{N}\}$ is a *limit*, i.e. an element of $\widehat{X} \setminus X$; the latter is usually written $(1, \omega, \omega)$, and is the least upper bound of all $(1, m, n)$, $m, n \in \mathbb{N}$. The downward-closure of $(1, \omega, \omega)$ in $\widehat{X}$, intersected with X, gives back the set of non-limit elements $\{(1, m, n) \mid m, n \in \mathbb{N}\}$.

This is a general situation: one can always write $\widehat{X}$ as the disjoint union $X \cup L$, so that any downward closed subset D of X can be written as $X \cap \downarrow A$, where A is a *finite* subset of $X \cup L$. Then L, the set of limits, is a *weak adequate domain of limits* (WADL) for X—we slightly simplify Definition 3.1 of [17], itself a slight generalization of [21]. In fact, $\widehat{X}$ (minus X) is the *smallest* WADL [17, Theorem 3.4].

$\widehat{X} = \mathit{Idl}(X)$ is always a continuous dcpo. In fact, it is even algebraic [6, Proposition 2.2.22]. It may however fail to be well, hence to be a cdcwo, see Lemma 1 below.

We have also described a hierarchy of datatypes on which completions are effective [17, Section 5]. Notably, $\widehat{\mathbb{N}} = \mathbb{N}_\omega$, $\widehat{A} = A$ for any finite poset, and $\widehat{\prod_{i=1}^k X_i} = \prod_{i=1}^k \widehat{X}_i$. Also, $\widehat{X^*}$ is the space of *products* on X, as defined in [1], i.e., regular expressions that are products of *atomic expressions* A^* ($A \in \mathbb{P}_{\text{fin}}(\widehat{X})$, where $\mathbb{P}_{\text{fin}}$ denotes the set of *finite* subsets) or $a^?$ ($a \in \widehat{X}$). In any case, elements of completions $\widehat{X}$ have a finite description, and the ordering $\subseteq$ on elements of $\widehat{X}$ is decidable [17, Theorem 5.3].

Having defined the completion $\widehat{X}$ of a wpo X, we can define the completion $\mathfrak{S} = \widehat{\mathfrak{X}}$ of a (functional) WSTS $\mathfrak{X} = (X, \xrightarrow{F}, \leq)$ as $(\widehat{X}, \xrightarrow{\mathcal{S}F}, \subseteq)$, where $\mathcal{S}F = \{\mathcal{S}f \mid f \in F\}$ [17, Section 6]. For each partial monotonic map $f \in F$, the partial continuous map $\mathcal{S}f : \widehat{S} \to \widehat{S}$ is such that $\operatorname{dom} \mathcal{S}f = \{C \in \widehat{X} \mid C \cap \operatorname{dom} f \neq \emptyset\}$, and $\mathcal{S}f(C) = \downarrow f(C)$ for every $C \in \widehat{X}$. In the cases of Petri nets or functional-lossy channel systems, the completed WSTS is effective [17, Section 6].

The important fact, which assesses the importance of the clover, is the following:

Proposition 2. *Let $\mathfrak{S} = \widehat{\mathfrak{X}}$ be the completion of the functional WSTS $\mathfrak{X} = (X, \xrightarrow{F}, \leq)$. For every state $s_0 \in X$, $\mathit{Cover}_{\mathfrak{X}}(s_0) = \mathit{Cover}_{\mathfrak{S}}(s_0) \cap X = \downarrow \mathit{Clover}_{\mathfrak{S}}(s_0) \cap X$.*

$\mathit{Cover}_{\mathfrak{S}}(s_0)$ is contained, usually strictly, in $\downarrow \mathit{Clover}_{\mathfrak{S}}(s_0)$. The above states that, when restricted to non-limit elements (in X), both contain the same elements. Taking lub-accelerations $(\mathcal{S}f)^\infty$ of any composition f of maps in F leaves $\mathit{Cover}_{\mathfrak{S}}(s_0)$, but is always contained in $\downarrow \mathit{Clover}_{\mathfrak{S}}(s_0) = \mathit{cl}(\mathit{Cover}_{\mathfrak{S}}(s_0))$. So we can safely lub-accelerate in $\mathfrak{S} = \widehat{\mathfrak{X}}$ to compute the clover in $\mathfrak{S}$. While the clover is larger than the cover, taking the intersection back with X will produce exactly the cover $\mathit{Cover}_{\mathfrak{X}}(s_0)$.

4 A Robust Class of WSTS: ω^2-WSTS

The construction of the completion $\mathfrak{S} = \widehat{\mathfrak{X}}$ of a WSTS $\mathfrak{X} = (X, \xrightarrow{F}, \leq)$ is almost perfect: the only missing ingredient to show that $\mathfrak{S}$ is a complete WSTS is to check that $\widehat{X}$ is well-ordered by inclusion. We have indeed seen that $\widehat{X}$ is a continuous dcpo; and $\mathfrak{S}$ is strongly monotonic, because $\mathcal{S}f$ is continuous, hence monotonic, for every $f \in F$.

We show that, in some cases, $\widehat{X}$ is indeed *not* well-ordered. Take X to be Rado's structure X_{Rado} [29], i.e., $\{(m, n) \in \mathbb{N}^2 \mid m < n\}$, ordered by $\leq_{\text{Rado}}$: $(m, n) \leq_{\text{Rado}}$

(m', n') iff $m = m'$ and $n \leq n'$, or $n < m'$. It is well-known that $\leq_{\text{Rado}}$ is a well quasi-ordering, and that $\mathbb{P}(X_{\text{Rado}})$ is not well-quasi-ordered by $\leq^\sharp_{\text{Rado}}$, defined as $A \leq^\sharp_{\text{Rado}} B$ iff for every $y \in B$, there is a $x \in A$ such that $x \leq_{\text{Rado}} y$ [26]; see for example [5, Example 3.2] for a readable reference. One can show that $\widehat{X_{\text{Rado}}} = Idl(X_{\text{Rado}})$ is comprised of all elements of X_{Rado}, plus infinitely many elements $\omega_0, \omega_1, \ldots, \omega_i, \ldots$, and ω, so that $(i, n) \leq \omega_i$ for all $n \geq i+1$, $\omega_i \leq \omega$ for all $i \in \mathbb{N}$, and $\{\omega_i \mid i \in \mathbb{N}\}$ is an antichain. We note that the latter is infinite. So:

Lemma 1. *$\widehat{X_{Rado}}$ is not well-ordered by inclusion.*

A well-quasi-order X is ω^2*-wqo* if and only if it does not contain an (isomorphic copy of) X_{Rado}, see e.g. [26]. We show that the above is the only case that can go bad:

Proposition 3. *Let S be a well-quasi-order. Then $\widehat{S}$ is well-quasi-ordered by inclusion iff S is ω^2-wqo.*

Let an ω^2*-WSTS* be any WSTS whose underlying poset is ω^2-wqo. It follows:

Theorem 1. *Let $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ be a functional WSTS. Then $\widehat{\mathfrak{S}}$ is a (complete, functional) WSTS iff $\mathfrak{S}$ is an ω^2-WSTS.*

All wpos used in the literature, and in fact all wpos arising from the hierarchy of data types of [17, Section 5] are ω^2-wqo. This follows from the fact that they are even better-quasi-ordered—see [5] for a gentle introduction to the latter concept.

Effective complete WSTS. The completion $\widehat{\mathfrak{S}}$ of a WSTS $\mathfrak{S}$ is effective iff the completion $\widehat{S}$ of the set of states is effective and if $\mathcal{S}f$ is recursive for all $f \in F$. $\widehat{S}$ is effective for all the data types of [17, Section 5]. Also, $\mathcal{S}f$ is indeed recursive for all $f \in F$, whether in Petri nets, functional-lossy channel systems (a way of recasting lossy channel systems as functional WSTS [17, Section 6]), reset/transfer Petri nets notably. As promised, we can now show:

Proposition 4. *There are effective complete WSTS $\mathfrak{S}$ such that the map $Clover_{\mathfrak{S}} : S \to \mathbb{P}_{fin}(S)$ is not recursive.*

Proof. Let $\mathfrak{S}$ be the completion of a functional-lossy channel system [17, Section 6] on the message alphabet Σ. By Theorem 1, $\mathfrak{S}$ is a complete WSTS. It is effective, too, see op.cit., or [1, Lemma 6]. $Clover_{\mathfrak{S}}(s_0)$ can be written as a tuple of control states and of *simple regular expression* $P_1 + \ldots + P_n$ representing the contents of channels. Each P_i is a product of atomic expressions A^* ($A \in \mathbb{P}_{\text{fin}}(\Sigma)$) or $a^?$ ($a \in \Sigma$). Now $Post^*_{\mathfrak{S}}(s_0)$ is finite iff none of these atomic expressions is of the form A^*. So computing $Clover_{\mathfrak{S}}(s_0)$ would allow one to decide boundedness for functional-lossy channel systems. However functional-lossy channel systems are equivalent to lossy channel systems in this respect, and boundedness is undecidable for the latter [9]. The same argument also applies to reset Petri nets [11]. □

5 A Conceptual Karp-Miller Procedure

We say that an effective complete (functional) WSTS $\mathfrak{S} = (S, \xrightarrow{F}, \leq)$ is ∞*-effective* iff every function g^∞ is computable, for every $g \in F^*$, where F^* is the set of all compositions of map in F. E.g., the completion of a Petri net is ∞-effective: not only is $\mathbb{N}_\omega^k$ a wpo, but every composition of transitions $g \in F^*$ is of the form $g(\boldsymbol{x}) = \boldsymbol{x} + \delta$, where $\delta \in \mathbb{Z}^k$. If $\boldsymbol{x} < g(\boldsymbol{x})$ then $\delta \in \mathbb{N}^k \setminus \{0\}$. Write $\boldsymbol{x}_i$ the ith component of $\boldsymbol{x}$, it follows that $g^\infty(\boldsymbol{x})$ is the tuple whose ith component is $\boldsymbol{x}_i$ if $\delta_i = 0$, ω otherwise.

Let $\mathfrak{S}$ be an ∞-effective WSTS, and write $A \sqsubseteq B$ iff $\downarrow A \subseteq \downarrow B$, i.e., iff every element of A is below some element of B. The following is a simple procedure which computes the clover of its input $s_0 \in S$ (when it terminates):

```
Procedure Clover_𝔖(s_0) :
1. $A \leftarrow \{s_0\}$;
2. while $Post_{\mathfrak{S}}(A) \not\sqsubseteq A$ do
     (a) Choose fairly $(g, a) \in F^* \times A$
         such that $a \in \operatorname{dom} g$;
     (b) $A \leftarrow A \cup \{g^\infty(a)\}$;
3. return $\operatorname{Max} A$;
```

Fig. 1. The **Clover**$_{\mathfrak{S}}$ procedure

Note that **Clover**$_{\mathfrak{S}}$ is well-defined and all its lines are computable by assumption, provided we make clear what we mean by fair choice in line (a). Call A_m the value of A at the start of the $(m-1)$st turn of the loop at step 2 (so in particular $A_0 = \{s_0\}$). The choice at line (a) is *fair* iff, on every infinite execution, every pair $(g, a) \in F^* \times A_m$ will be picked at some later stage $n \geq m$.

Our procedure is more conceptual than the existing proposals, which generally build a tree [27,15,16,22] or a graph [12] for computing the clover. We shall see that termination of **Clover**$_{\mathfrak{S}}$ has strong ties with the theory of *flattening* [8]; but this paper requires one to enumerate sets of the form $g^*(\boldsymbol{x})$, which is sometimes harder than computing just the element $g^\infty(\boldsymbol{x})$. For example, if $g : \mathbb{N}^k \to \mathbb{N}^k$ is an affine map $g(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$ with $A \geq 0$ and $\boldsymbol{b} \geq 0$ then $g^\infty(\boldsymbol{x})$ is computable as a vector in $\mathbb{N}_\omega^k$ [18, Theorem 7.9], but $g^*(\boldsymbol{x})$ is not even definable by a Presburger formula.

Finally, we use a *fixpoint test* (line 2) that is not in the Karp-Miller algorithm; and this improvement allows **Clover**$_{\mathfrak{S}}$ to terminate in *more cases* than the Karp-Miller procedure when it is used for extended Petri nets (for reset Petri nets for instance, which are a special case of the affine maps above), as we shall see. To decide whether the current set A, which is always an under-approximation of $Clover_{\mathfrak{S}}(s_0)$, is the clover, it is enough to decide whether $Post_{\mathfrak{S}}(A) \sqsubseteq A$. The various Karp-Miller procedures only test each branch of a tree separately, to the partial exception of the minimal coverability tree algorithm [15] and the recent coverability algorithm [22], which compare nodes across branches. That the simple test $Post_{\mathfrak{S}}(A) \sqsubseteq A$ does all this at once does not seem to have been observed until now.

By Proposition 4, we cannot hope to have **Clover**$_{\mathfrak{S}}$ terminate on all inputs. But:

Theorem 2 (Correctness). *If* **Clover**$_{\mathfrak{S}}(s_0)$ *terminates, then it computes* $Clover_{\mathfrak{S}}(s_0)$.

If the generalized Karp-Miller Tree procedure [15] terminates then it has found a finite set $g_1, g_2, ..., g_n$ of maps to lub-accelerate. These lub-accelerations will also be found by **Clover**$_{\mathfrak{S}}$, by fairness. From the fixpoint test, **Clover**$_{\mathfrak{S}}$ will also stop. The reset Petri net of [11, Example 3], with an extra transition that adds a token to each place, is an example where the generalized Karp-Miller procedure does not terminate, while **Clover**$_{\mathfrak{S}}$ terminates. So:

Proposition 5. *The procedure* **Clover**$_{\mathfrak{S}}$ *terminates in more cases than the generalized Karp-Miller procedure.*

Termination is however undecidable, using Proposition 4 and Theorem 2.

Proposition 6. *There is an ∞-effective complete WSTS such that the termination of* **Clover**$_{\mathfrak{S}}$ *is undecidable.*

We now characterize those transition systems on which **Clover**$_{\mathfrak{S}}$ terminates.

A functional transition system $(\mathfrak{S}, \xrightarrow{F})$ with initial state s_0 is *flat* iff there are finitely many words $w_1, w_2, ..., w_k \in F^*$ such that any fireable sequence of transitions from s_0 is contained in the language $w_1^* w_2^* ... w_k^*$. (We equate functions in F with letters from the alphabet F, and understand words as the corresponding composition of maps.) Ginsburg and Spanier [24] call this a *bounded* language, and show that it is decidable whether any context-free language is flat.

Not all systems of interest are flat. For an arbitrary system S, *flattening* [8] consists in finding a flat system S', equivalent to S w.r.t. reachability, and computing on S' instead of S. We adapt the definition in [8] to functional transition systems (without an explicit finite control graph). A functional transition system $\mathfrak{S}_1 = (S_1, \xrightarrow{F_1})$, together with a map $\varphi : S_1 \to S_2$ and a map, also written φ, from F_1 to F_2, is a *flattening* of a functional transition system $\mathfrak{S}_2 = (S_2, \xrightarrow{F_2})$ iff (1) $\mathfrak{S}_1$ is flat and (2) for all $(s, s') \in S_1^2$, for all $f_1 \in F_1$ such that $s \in \operatorname{dom} f_1$ and $s' = f_1(s)$, $\varphi(s) \in \operatorname{dom} \varphi(f_1)$ and $\varphi(s') = \varphi(f_1)(\varphi(s))$. (I.e., φ is a morphism of transition systems.) Let us recall that $(\mathfrak{S}, s_0)$ is *$Post^*$-flattable* iff there is a flattening $\mathfrak{S}_1$ of $\mathfrak{S}$ and a state s_1 of $\mathfrak{S}_1$ such that $\varphi(s_1) = s_0$ and $Post^*_{\mathfrak{S}}(s_0) = \varphi(Post^*_{\mathfrak{S}_1}(s_1))$. A flattening is *continuous* iff $\mathfrak{S}_1$ is an complete transition system and $\varphi : S_1 \to S_2$ is continuous. Correspondingly, we say that $(\mathfrak{S}, s_0)$ is *clover-flattable* iff there is an continuous flattening $\mathfrak{S}_1$, φ of $\mathfrak{S}$ and a state s_1 of $\mathfrak{S}_1$ such that $\varphi(s_1) = s_0$ and $Clover_{\mathfrak{S}}(s_0) \sqsubseteq \varphi(Clover_{\mathfrak{S}_1}(s_1))$.

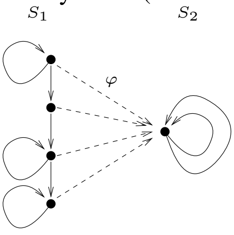


Fig. 2. Flattening

We obtain the following; the proof is non-trivial, and omitted for lack of space.

Theorem 3. *Let $\mathfrak{S}$ be an ∞-effective complete WSTS. The procedure* **Clover**$_{\mathfrak{S}}$ *terminates on s_0 iff $(\mathfrak{S}, s_0)$ is clover-flattable. Then we can even require that the continuous flattening has the same clover up to φ, i.e., $Clover_{\mathfrak{S}}(s_0) = \operatorname{Max} \varphi(Clover_{\mathfrak{S}_1}(s_1))$.*

6 Application: Well Structured Counter Systems

We now demonstrate how the fairly large class of counter systems fits with our theory. We show that counter systems composed of affine monotone functions with upward closed definition domains are complete (strongly monotonic) WSTS. This result is obtained by showing that every monotone affine function is continuous and its lub-acceleration f^∞ is computable. Moreover, we prove that it is possible to decide whether a general counter system (given by a finite set of Presburger relations) is a monotone affine counter system, but that one cannot decide whether it is a WSTS.

Definition 4. *A* relational counter system *(with n counters), for short an R*-counter system*, $\mathcal{C}$ is a tuple $\mathcal{C} = (Q, R, \to)$ where Q is a finite set of control states, $R = \{r_1, r_2, ...r_k\}$ is a finite set of Presburger relations $r_i \subseteq \mathbb{N}^n \times \mathbb{N}^n$ and $\to \subseteq Q \times R \times Q$.*

We will consider a special case of Presburger relations, those which allow to code the graph of affine functions. A (partial) function $f : \mathbb{N}^n \longrightarrow \mathbb{N}^n$ is *non-negative affine*, for short *affine* if there exist a matrix $A \in \mathbb{N}^{n \times n}$ with *non-negative coefficients* and a vector $b \in \mathbb{Z}^n$ such that for all $\boldsymbol{x} \in \operatorname{dom} f, f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$. When necessary, we will extend affine maps $f : \mathbb{N}^n \longrightarrow \mathbb{N}^n$ by continuity to $f : \mathbb{N}^n_\omega \longrightarrow \mathbb{N}^n_\omega$, by $f(lub_{i \in \mathbb{N}}(\boldsymbol{x}_i)) = lub_{i \in \mathbb{N}}(f(\boldsymbol{x}_i))$ for every countable chain $(\boldsymbol{x}_i)_{i \in \mathbb{N}}$ in $\mathbb{N}^n$.

Definition 5. *An* affine counter system *(with n counters) (ACS) $\mathcal{C} = (Q, R, \rightarrow)$ is a R-counter system where all relations r_i are (partial) affine functions.*

The domain of maps f in an affine counter system ACS are Presburger-definable. A reset/transfer Petri net is an ACS where every line or column of every matrix contains at most one non-zero coefficient equal to 1, and, all domains are upward closed sets. A *Petri net* is an ACS where all affine maps are translations with upward closed domains.

Theorem 4. *One can decide whether an effective relational counter system is an ACS.*

Proof. The formula expressing that a relation is a function is a Presburger formula, hence one can decide whether R is the graph of a function. One can also decide whether the graph G_f of a function f is monotone because monotonicity of a Presburger-definable function can be expressed as a Presburger formula. Finally, one can also decide whether a Presburger formula represents an affine function $f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$ with $A \in \mathbb{N}^{n \times n}$ and $\boldsymbol{b} \in \mathbb{Z}^n$ from [10]. □

For counter systems (which include Minsky machines), monotonicity is undecidable. Clearly, a counter system $\mathfrak{S}$ is well-structured iff $\mathfrak{S}$ is monotone: so there is no algorithm to decide whether a relational counter system is a WSTS. However, an ACS is strongly monotonic iff each map f is partial monotonic; this is equivalent to requiring that $\operatorname{dom} f$ is upward closed, since all matrices A have non-negative coefficients. This is easily cast as Presburger formula, and therefore decidable.

Proposition 7. *There is an algorithm to decide whether an ACS is a strongly monotonic WSTS.*

We have recalled that Petri net functions ($f(x) = x + b$, $b \in \mathbb{Z}^n$ and $\operatorname{dom}(f)$ upward closed) can be lub-accelerated effectively. This result was generalized to broadcast protocols (equivalent to transfer Petri nets) by Emerson and Namjoshi [12] and to a class of affine functions $f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$ such that $A \in \mathbb{N}^{n \times n}$, $b \in \mathbb{N}^n$ and $\operatorname{dom}(f)$ is upward closed [18]. Antonik recently extended this result to Presburger monotone affine functions: for every $f(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$ with $A \in \mathbb{N}^{n \times n}$, $b \in \mathbb{Z}^n$ and $\operatorname{dom}(f)$ Presburger-definable, the function f^∞ is recursive [7]. We deduce the following strong relationship between well-structured ACS and complete well-structured ACS.

Theorem 5. *The completion of an ACS S is an ∞-effective complete WSTS iff S is a strongly monotonic WSTS.*

Proof. Strong monotonicity reduces to partial monotonicity of each map f, as discussed above. Well-structured ACS are clearly effective, since $Post(\boldsymbol{s}) = \{\boldsymbol{t} \mid \exists f \in F \cdot f(\boldsymbol{t}) = \boldsymbol{s}\}$ is Presburger-definable. Note also that monotone affine function are continuous, and $\mathbb{N}^n_\omega$ is cdcwo. Finally, for every Presburger monotone affine function f, the function f^∞ is recursive, so the considered ACS is ∞-effective. □

Corollary 1. *One may decide whether the completion of an ACS is an ∞-effective complete WSTS.*

So the completions of reset/transfer Petri nets [11], broadcast protocols [13], self-modifying Petri nets [30] and affine well-structured nets [18] are ∞-effective complete WSTS.

7 Conclusion and Perspectives

We have provided a framework of *complete WSTS*, and of *completions* of WSTS, on which forward reachability analyses can be conducted, using natural finite representations for downward closed sets. The central element of this theory is the *clover*, i.e., the set of maximal elements of the closure of the cover. We have shown that, for complete WSTS, the clover is finite and describes the closure of the cover exactly. When the original WSTS is not complete, we have shown the the general completion of WSTS defined in [17] is still a WSTS, iff the original WSTS is an ω^2*-WSTS*. This charaterize a new, robust class of WSTS. We have also defined a simple procedure for computing the clover for ∞-effective complete WSTS, and we have shown that it terminates iff the WSTS is *clover-flattable*, iff it contains a flat subsystem having the same clover. We have also observed procedure terminates in more cases than the Karp-Miller procedure when applied to extensions of Petri nets.

In the future, we shall explore efficient strategies for choosing sequences $g \in F^*$ to lub-accelerate in the $\mathbf{Clover}_\mathfrak{S}$ procedure. We will also analyze whether $\mathbf{Clover}_\mathfrak{S}$ terminates in models such as BVASS [31], transfer Data nets [28], reconfigurable nets, timed Petri nets [4], post-self-modifying Petri nets [30] and strongly monotone affine well-structured nets [18]), i.e., whether they are clover-flattable.

References

1. Abdulla, P., Bouajjani, A., Jonsson, B.: On-the-fly analysis of systems with unbounded, lossy Fifo channels. In: Y. Vardi, M. (ed.) CAV 1998. LNCS, vol. 1427, pp. 305–318. Springer, Heidelberg (1998)
2. Abdulla, P.A., Čerāns, K., Jonsson, B., Tsay, Y.-K.: Algorithmic analysis of programs with well quasi-ordered domains. Information and Computation 160(1–2), 109–127 (2000)
3. Abdulla, P.A., Collomb-Annichini, A., Bouajjani, A., Jonsson, B.: Using forward reachability analysis for verification of lossy channel systems. Formal Methods in System Design 25(1), 39–65 (2004)
4. Abdulla, P.A., Deneux, J., Mahata, P., Nylén, A.: Forward reachability analysis of timed petri nets. In: Lakhnech, Y., Yovine, S. (eds.) FORMATS/FTRTFT 2004. LNCS, vol. 3253, pp. 343–362. Springer, Heidelberg (2004)
5. Abdulla, P.A., Nylén, A.: Better is better than well: On efficient verification of infinite-state systems. In: 14th LICS, pp. 132–140 (2000)
6. Abramsky, S., Jung, A.: Domain theory. In: Abramsky, S., Gabbay, D.M., Maibaum, T.S.E. (eds.) Handbook of Logic in Computer Science, vol. 3, pp. 1–168. Oxford University Press, Oxford (1994)
7. Antonik, A.: Presburger monotone affine functions can be lub-accelerated. Personal communication (2009)
8. Bardin, S., Finkel, A., Leroux, J., Schnoebelen, P.: Flat acceleration in symbolic model checking. In: Peled, D.A., Tsay, Y.-K. (eds.) ATVA 2005. LNCS, vol. 3707, pp. 474–488. Springer, Heidelberg (2005)

9. Cécé, G., Finkel, A., Purushothaman Iyer, S.: Unreliable channels are easier to verify than perfect channels. Information and Computation 124(1), 20–31 (1996)
10. Demri, S., Finkel, A., Goranko, V., van Drimmelen, G.: Towards a model-checker for counter systems. In: Graf, S., Zhang, W. (eds.) ATVA 2006. LNCS, vol. 4218, pp. 493–507. Springer, Heidelberg (2006)
11. Dufourd, C., Finkel, A., Schnoebelen, P.: Reset nets between decidability and undecidability. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 103–115. Springer, Heidelberg (1998)
12. Emerson, E.A., Namjoshi, K.S.: On model-checking for non-deterministic infinite-state systems. In: 13th LICS, pp. 70–80 (1998)
13. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: 14th LICS, pp. 352–359 (1999)
14. Finkel, A.: A generalization of the procedure of Karp and Miller to well structured transition systems. In: Ottmann, T. (ed.) ICALP 1987. LNCS, vol. 267, pp. 499–508. Springer, Heidelberg (1987)
15. Finkel, A.: Reduction and covering of infinite reachability trees. Information and Computation 89(2), 144–179 (1990)
16. Finkel, A.: The minimal coverability graph for Petri nets. In: Rozenberg, G. (ed.) APN 1993. LNCS, vol. 674, pp. 210–243. Springer, Heidelberg (1993)
17. Finkel, A., Goubault-Larrecq, J.: Forward analysis for WSTS, part I: Completions. In: 26th STACS, Freiburg, Germany. Springer, Heidelberg (to appear, 2009)
18. Finkel, A., McKenzie, P., Picaronny, C.: A well-structured framework for analysing Petri net extensions. Information and Computation 195(1-2), 1–29 (2004)
19. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theoretical Computer Science 256(1–2), 63–92 (2001)
20. Ganty, P., Raskin, J.-F., van Begin, L.: A complete abstract interpretation framework for coverability properties of WSTS. In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 49–64. Springer, Heidelberg (2005)
21. Geeraerts, G., Raskin, J.-F., van Begin, L.: Expand, enlarge and check: New algorithms for the coverability problem of WSTS. J. Comp. and System Sciences 72(1), 180–203 (2006)
22. Geeraerts, G., Raskin, J.-F., van Begin, L.: On the efficient computation of the minimal coverability set for Petri nets. In: Namjoshi, K.S., Yoneda, T., Higashino, T., Okamura, Y. (eds.) ATVA 2007. LNCS, vol. 4762, pp. 98–113. Springer, Heidelberg (2007)
23. Gierz, G., Hofmann, K.H., Keimel, K., Lawson, J.D., Mislove, M., Scott, D.S.: Continuous lattices and domains. In: Encyclopedia of Mathematics and its Applications, vol. 93. Cambridge University Press, Cambridge (2003)
24. Ginsburg, S., Spanier, E.H.: Bounded Algol-like languages. Trans. American Mathematical Society 113(2), 333–368 (1964)
25. Goubault-Larrecq, J.: On Noetherian spaces. In: 22nd LICS, Wrocław, Poland, pp. 453–462. IEEE Computer Society Press, Los Alamitos (2007)
26. Jančar, P.: A note on well quasi-orderings for powersets. Information Processing Letters 72(5–6), 155–160 (1999)
27. Karp, R.M., Miller, R.E.: Parallel program schemata. J. Comp. and System Sciences 3(2), 147–195 (1969)
28. Lazič, R., Newcomb, T., Ouaknine, J., Roscoe, A.W., Worrell, J.: Nets with tokens which carry data. Fundamenta Informaticae 88(3), 251–274 (2008)
29. Rado, R.: Partial well-ordering of sets of vectors. Mathematika 1, 89–95 (1954)
30. Valk, R.: Self-modidying nets, a natural extension of Petri nets. In: Ausiello, G., Böhm, C. (eds.) ICALP 1978, vol. 62, pp. 464–476. Springer, Heidelberg (1978)
31. Verma, K.N., Goubault-Larrecq, J.: Karp-Miller trees for a branching extension of VASS. Discrete Mathematics & Theoretical Computer Science 7(1), 217–230 (2005)

Qualitative Concurrent Stochastic Games with Imperfect Information*

Vincent Gripon and Olivier Serre

LIAFA (CNRS & Université Paris Diderot – Paris 7)

Abstract. We study a model of games that combines concurrency, imperfect information and stochastic aspects. Those are finite states games in which, at each round, the two players choose, *simultaneously* and *independently*, an action. Then a successor state is chosen accordingly to some fixed probability distribution depending on the previous state and on the pair of actions chosen by the players. Imperfect information is modeled as follows: both players have an equivalence relation over states and, instead of observing the exact state, they only know to which equivalence class it belongs. Therefore, if two partial plays are indistinguishable by some player, he should behave the same in both of them. We consider reachability (does the play eventually visit a final state?) and Büchi objective (does the play visit infinitely often a final state?).

Our main contribution is to prove that the following problem is complete for 2-ExpTime: decide whether the first player has a strategy that ensures her to almost-surely win against *any* possible strategy of her oponent. We also characterise those strategies needed by the first player to almost-surely win.

1 Introduction

Perfect information turn based two-player games on a graph [10] are widely studied in computer science. Indeed, they are a useful tool for both theoretical (for instance the modern proofs of Rabin's complementation lemma rely on the memoryless determinacy of parity games [11]) and more practical applications. On the practical side, a major application of games is for the verification of reactive open systems. Those are systems composed of both a program and some (possibly hostile) environment. The verification problem consists of deciding whether the program can be restricted so that the system meets some given specification whatever the environment does. Here, restricting the system means synthesizing some controller, which, in term of games, is equivalent to designing a winning strategy for the player modeling the program [14].

The perfect information turn-based model, even if it suffices in many situations, is somewhat weak for the following two reasons. First, it does not permit to capture the behavior of real concurrent models where, in each step, the program and its environment *independently* choose moves, whose *parallel* execution

* Supported by the ANR project JADE and by the ESF project GASICS.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 200–211, 2009.

determines the next state of the system. Second, in this model both players have, at each time, a perfect information on the current state of the play: this, for instance, forbids to model a system where the program and the environment share some public variables while having also their own private variables [15].

In this paper, we remove those two restrictions by considering concurrent stochastic games with imperfect information. Those are finite states games in which, at each round, the two players choose *simultaneously* and *independently* an action. Then a successor state is chosen accordingly to some fixed probability distribution depending on the previous state and on the pair of actions chosen by the players. Imperfect information is modeled as follows: both players have an equivalence relation over states and, instead of observing the exact state, they only see to which equivalence class it belongs. Therefore, if two partial plays are indistinguishable by some player, he should behave the same in both of them. Note that this model naturally captures several model studied in the literature [1,9,7,8]. The winning conditions we consider here are reachability (is there a final state eventually visited?), Büchi (is there a final state that is visited infinitely often?) and their dual versions, safety and co-Büchi.

We study qualitative properties of those games (note that quantitative properties — *e.g.* deciding whether the value of the game is above a given threshold — are already undecidable in much weaker models [13]). More precisely, we investigate the question of deciding whether some player can almost-surely win, that is whether he has a strategy that wins with probability 1 against any counter strategy of the oponent. Our main contributions is to prove that, for both reachability and Büchi objectives, one can decide, in doubly exponential time (which is proved to be optimal), whether the first player has an almost-surely winning strategy. Moreover, when it is the case, we are also able to construct such a finite-memory strategy. We also provide intermediate new results concerning positive winning in safety (and co-Büchi) $1\frac{1}{2}$-player games (*a.k.a* partial observation Markov decision process).

Related work. Concurrent games with perfect information have been deeply investigated in the last decade [2,1,7]. Games with imperfect information have been considered for turn-based model [15] as well as for concurrent models with only one imperfectly informed player [9,8]. To our knowledge, the present paper provides the first positive results on a model of games that combines concurrency, imperfect information (on both sides) and stochastic transition function. In a recent independent work [4], Bertrand, Genest and Gimbert obtain similar results than the one presented here for a closely related model. The main differences with our model are the following: Bertand *et al.* consider a slightly weaker model of games in which the players may observe their own actions, and they allow the players to use richer strategies where the players can randomly update their memory (note that those strategies when used in our model seem strictly more powerful than the one we consider [12]). Bertand *et al.* also discuss qualitative determinacy results and consider the case where a player is more informed than the other. We refer the reader to [4] for a detailed exposition.

2 Definitions

A **probability distribution** over a finite set X is a mapping $d : X \to [0,1]$ such that $\sum_{x \in X} d(x) = 1$. In the sequel we denote by $\mathcal{D}(X)$ the set of probability distributions over X.

Given some set X and some equivalence relation $\sim$ over X, $[x]_\sim$ stands for the equivalence class of x for $\sim$ and $X/_\sim = \{[x]_\sim \mid x \in X\}$ denotes the set of equivalence classes of $\sim$.

For some finite alphabet A, A^* (*resp.* A^ω) designates the set of finite (*resp.* infinite) words over A.

2.1 Arenas

A **concurrent arena with imperfect information** is a tuple $\mathcal{A} = \langle S, \Sigma_E, \Sigma_A, \delta, \sim_E, \sim_A \rangle$ where

- S is a finite set of control states;
- Σ_E (*resp.* Σ_A) is the (finite) set of actions for Eve (*resp.* Adam);
- $\delta : S \times \Sigma_E \times \Sigma_A \to \mathcal{D}(S)$ is the transition (total) function;
- $\sim_E$ and $\sim_A$ are two equivalence relations over states.

A play in a such an arena proceeds as follows. First it starts in some initial state s. Then Eve picks an action $\sigma_E \in \Sigma_E$ and, *simultaneously* and *independently*, Adam chooses an action $\sigma_A \in \Sigma_A$. Then a successor state is chosen accordingly to the probability distribution $\delta(s, \sigma_E, \sigma_A)$. Then the process restarts: the players choose a new pair of actions that induces, together with the current state, a new state and so on forever. Hence a **play** is an infinite sequence $s_0 s_1 s_2 \cdots$ in S^ω such that for every $i \geq 0$, there exists $(\sigma_E, \sigma_A) \in \Sigma_E \times \Sigma_A$ with $\delta(s_i, \sigma_E, \sigma_A)(s_{i+1}) > 0$. In the sequel we refer to a prefix of a play as a **partial play** and we denote by $Plays(\mathcal{A})$ the set of all plays in arena $\mathcal{A}$.

The intuitive meaning of $\sim_E$ (*resp.* $\sim_A$) is that two states s_1 and s_2 such that $s_1 \sim_E s_2$ (*resp.* $s_1 \sim_A s_2$) cannot be distinguished by Eve (*resp.* by Adam). We easily extend the relation $\sim_E$ to partial plays: let $\lambda = s_0 s_1 \cdots s_n$ and $\lambda' = s'_0 s'_1 \cdots s'_n$ be two partial plays, then $\lambda \sim_E \lambda'$ if and only if $s_i \sim_E s'_i$ for all $i = 0, \cdots, n$.

Note that perfect information concurrent arenas (in the sense of [2,1]) correspond to the special case where $\sim_E$ and $\sim_A$ are the equality relation over S.

2.2 Strategies

In order to choose their moves the players follow strategies, and, for this, they may use all the information they have about what was played so far. However, if two partial plays are equivalent for $\sim_E$, then Eve cannot distinguish them, and should therefore behave the same. This leads to the following notion.

An **observation-based strategy** for Eve is a function $\varphi_E : (S/_{\sim_E})^* \to \mathcal{D}(\Sigma_E)$, *i.e.*, to choose her next action, Eve considers the sequence of observations she got so far. In particular, a strategy φ_E is such that $\varphi_E(\lambda) = \varphi_E(\lambda')$ whenever $\lambda \sim_E \lambda'$. Observation-based strategies for Adam are defined similarly.

Of special interest are those strategies that does not require memory: a **memoryless observation-based strategies** for Eve is a function from $S/_{\sim_E} \to \mathcal{D}(\Sigma_E)$, that is to say these strategies only depend of the current equivalence class.

A **uniform strategy** for some player X is a strategy φ such that for all partial play λ, the probability measure $\varphi(\lambda)$ is uniform, *i.e.*, for all action $\sigma_X \in \Sigma_X$, either $\varphi(\lambda)(\sigma_X) = 0$ or $\varphi(\lambda)(\sigma_X) = \frac{1}{|\{\sigma_X \in \Sigma_X | \varphi(\lambda)(\sigma_X) \neq 0\}|}$. The set of memoryless uniform strategies for X is a finite set containing $(2^{|\Sigma_X|} - 1)^{|S|}$ elements. Equivalently those strategies can be seen as functions to (non-empty) sets of (authorised) actions.

A **finite-memory strategy** for Eve with memory M (M being a finite set) is some triple $\varphi = (Move, Up, m_0)$ where $m_0 \in M$ is the initial memory, $Move : M \to \mathcal{D}(\Sigma_E)$ associates a distribution of actions with any element in the memory M and $Up : M \times S/_{\sim_E} \to M$ is a mapping updating the memory with respect to some observation. One defines $\varphi(s_0) = Move(m_0)$ and $\varphi(s_0 \cdots s_n) = Move(Up(\cdots Up(Up(m_0, [s_1]/_{\sim_E}), [s_2]/_{\sim_E}), \cdots, [s_n]/_{\sim_E}) \cdots)$ for any $n \geq 1$. Hence, a finite-memory strategy is some observation-based strategy that can be implemented by a finite transducer whose set of control states is M.

Remark 1. Note that in our definition of a strategy (and more generally in the definition of a play) we implicitly assume that the players only observe the sequence of states and not the corresponding sequence of actions. While the fact that Eve does not observe what Adam played is rather fair (otherwise imperfect information on states would make less sense) one could object that Eve should observes the actions she played so far. Here, our view of a (randomised) strategy is the following: when Eve respects some strategy, it means that whenever she has to play, her strategy provides her a distribution that she sends to some scheduler that, together with the distribution chosen by Adam, picks the next state. Indeed, it permits for instance to model a system in which some agent does not have the resources to implement himself randomisation.

An alternative option would be to consider that Eve flips a coin to pick her action and then sends this action to the scheduler that, together with the action chosen by Adam, picks the next state. In this case, a strategy should depend on the sequence of states together with the associated sequence of actions played by Eve. We argue that this second approach can be simulated easily by the first one, hence justifying our initial choice. Indeed, one can always enrich the set of states to encode the last pair of actions played and then use the equivalence relations $\sim_E$ / $\sim_A$ to hide / show part of this information to the respective players.

2.3 Probability Space and Outcomes of Strategies

Let $\mathcal{A} = \langle S, \Sigma_E, \Sigma_A, \delta, \sim_E, \sim_A \rangle$ be a concurrent arena with imperfect information, let $s_0 \in S$ be an initial state, φ_E be a strategy for Eve and φ_A be a strategy for Adam. In the sequel we are interested in defining the probability of a (measurable) set of plays knowing that Eve (*resp.* Adam) plays accordingly φ_E (*resp.* φ_A). This is done in the classical way: first one defines the probability measure for basic sets of plays (called here *cones* and corresponding to plays having some initial common prefix) and then extends it in a unique way to all measurable sets.

First define $Outcomes(s_0, \varphi_E, \varphi_A)$ to be the set of all possible plays when the game starts on s_0 and when Eve and Adam plays respectively accordingly to φ_E and φ_A. More formally, an infinite play $\lambda = s_0 s_1 \cdots$ belongs to $Outcomes(s_0, \varphi_E, \varphi_A)$ if and only if, for every $i \geq 0$, there is a pair of actions $(\sigma_E, \sigma_A) \in \Sigma_E \times \Sigma_A$ with $\delta(s_i, \sigma_E, \sigma_A)(s_{i+1}) > 0$ and s.t. $\varphi_E(s_0 s_1 \cdots s_i)(\sigma_E) > 0$ and $\varphi_A(s_0 s_1 \cdots s_i)(\sigma_A) > 0$ (*i.e.* σ_X is possible accordingly to φ_X, for $X = E, A$).

Now, for any partial play λ, the **cone** for λ is the set $cone(\lambda) = \lambda \cdot S^\omega$ of all infinite plays with prefix λ. Denote by $Cones$ the set of all possible cones and let $\mathcal{F}$ be the Borel σ-field generated by $Cones$ considered as a set of basic open sets (*i.e.* $\mathcal{F}$ is the smallest set containing $Cones$ and closed under complementation, countable union and countable intersection). Then $(Plays(\mathcal{A}), \mathcal{F})$ is a σ-algebra.

A pair of strategies (φ_E, φ_A) induces a probability space over $(Plays(\mathcal{A}), \mathcal{F})$. Indeed one can define a measure $\mu_{s_0}^{\varphi_E, \varphi_A} : Cones \to [0,1]$ on cones (this task is easy as a cone is uniquely defined by a finite partial play) and then uniquely extend it to a probability measure on $\mathcal{F}$ using the Carathéodory Unique Extension Theorem. For this, one defines $\mu_{s_0}^{\varphi_E, \varphi_A}$ inductively on cones:

- $\mu_{s_0}^{\varphi_E, \varphi_A}(s) = 1$ if $s = s_0$ and $\mu_{s_0}^{\varphi_E, \varphi_A}(s) = 0$ otherwise.
- For every partial play λ ending in some vertex s,

$$\mu_{s_0}^{\varphi_E, \varphi_A}(\lambda \cdot s') = \mu_{s_0}^{\varphi_E, \varphi_A}(\lambda). \sum_{(\sigma_E, \sigma_A)} \varphi_E(\lambda)(\sigma_E).\varphi_A(\lambda)(\sigma_A).\delta(s, \sigma_E, \sigma_A)(s')$$

Denote by $\Pr_{s_0}^{\varphi_E, \varphi_A}$ the unique extension of $\mu_{s_0}^{\varphi_E, \varphi_A}$ to a probability measure on $\mathcal{F}$. Then $(Plays(\mathcal{A}), \mathcal{F}, \Pr_{s_0}^{\varphi_E, \varphi_A})$ is a probability space.

2.4 Objectives, Value of a Game

Fix a concurrent arena with imperfect information $\mathcal{A}$. An objective for Eve is a measurable set $\mathcal{O} \subseteq Plays(\mathcal{A})$: a play is won by her if it belongs to $\mathcal{O}$; otherwise it is won by Adam. A **concurrent game with imperfect information** is a triple $(\mathcal{A}, s_0, \mathcal{O})$ where $\mathcal{A}$ is a concurrent arena with imperfect information, s_0 is an initial state and $\mathcal{O}$ is an objective. In the sequel we focus on the following special classes of objectives (note that all of them are Borel sets hence measurable) that we define as means of a subset $F \subseteq S$ of **final states**.

- A **reachability objective** is of the form S^*FS^ω: a play is winning if it eventually goes through some final state.
- A **safety objective** is the dual of a reachability objective, *i.e.* is of the form $(S \setminus F)^\omega$: a play is winning if it never goes through a final state.
- A **Büchi objective** is of the form $\bigcap_{k\geq 0} S^k S^* F S^\omega$: a play is winning if it goes infinitely often through final states.
- A **co-Büchi objective** is the dual of a Büchi objective, *i.e.* is of the form $S^*(S \setminus F)^\omega$: a play is winning if it goes finitely often through final states.

A reachability (*resp.* safety, Büchi, co-Büchi) game is a game equipped with a reachability (*resp.* safety, Büchi, co-Büchi) objective. In the sequel we may replace $\mathcal{O}$ by F when it is clear from the context which winning condition we consider.

Fix a concurrent game with imperfect information $\mathbb{G} = (\mathcal{A}, s_0, \mathcal{O})$. A strategy φ_E for Eve is **almost-surely winning** if, for any counter-strategy φ_A for Adam, $\mathrm{Pr}_{s_0}^{\varphi_E,\varphi_A}(\mathcal{O}) = 1$. If such a strategy exists, we say that Eve **almost-surely wins** $\mathbb{G}$. A strategy φ_E for Eve is **positively winning** if, for any counter-strategy φ_A for Adam, $\mathrm{Pr}_{s_0}^{\varphi_E,\varphi_A}(\mathcal{O}) > 0$. If such a strategy exists, we say that Eve **positively wins** $\mathbb{G}$.

3 Knowledge Arena

For the rest of this section we let $\mathcal{A}$ be a concurrent arena with imperfect information with $\mathcal{A} = \langle S, \Sigma_E, \Sigma_A, \delta, \sim_E, \sim_A\rangle$ and let $s_0 \in S$ be some initial state.

Remark that in our model, the players do not observe the actions they play but they may know the distribution they have chosen. Therefore, one could consider a new arena in which the states have a component indicating the domain of the last distribution chosen by Eve, and that this component is visible only to her (it is hidden to Adam by the equivalence relantion $\sim_A$).

Even, if she does not see the precise control state, Eve can deduce information about it from previous information on the control state and from the set of possible actions she just played. We should refer to this as the **knowledge** of Eve, which formally is a set of states. Assume Eve knows that the current state belongs to some set $K \subseteq S$. After the next move Eve observes the equivalence class $[s]_{\sim_E}$ of the new control state and she also knows the subset D of actions she may have played (it is the domain of the distribution she chose): hence she can compute the set of possible states the play can be in. This is done using the function $\mathrm{UpKnow} : 2^S \times [S]_{/\sim_E} \times 2^{\Sigma_E} \to 2^{\Sigma_S}$ defined by letting

$$\mathrm{UpKnow}(K, [s]_{\sim_E}, D) = \\ \{t \sim_E s \mid \exists r \in K,\ \sigma'_E \in D,\ \sigma_A \in \Sigma_A \text{ s.t. } \delta(r, \sigma'_E, \sigma_A)(t) > 0\},$$

i.e. in order to update her current knowledge, observing in which equivalence class is the new control state, and knowing that she played an action in $D \subseteq \Sigma_E$, Eve computes the set of all states in this class that may be reached from a state in her former knowledge.

Based on our initial remark and on the notion of knowledge we define the **knowldege arena associated with $\mathcal{A}$**, denoted $\mathcal{A}^K$. The arena $\mathcal{A}^K$ is designed to make explicit the information Eve can collect on her moves (*i.e.* the domain of the distributions she plays) and on the possible current state. We define $\mathcal{A}^K = \langle S^K, \Sigma'_E, \Sigma_A, \delta^K, \sim_E^K, \sim_A^K \rangle$ as follows:

- $S^K = \{(s, K, D) \in S \times 2^S \times 2^{\Sigma_E} \mid K \subseteq [s]_{/\sim E}\}$: the first component is the real state, the second one is the current knowledge of Eve and the third one is the domain of the last distribution she played;
- $\Sigma'_E = \Sigma_E \times (2^{\Sigma_E} \setminus \emptyset)$: actions of Eve will now contain information on the domain of actions of the distributions she picks;
- $\delta^K((s, K, D), (\sigma_E, D'), \sigma_A)(s', K', D'') = 0$ if $D' \neq D''$ or $K' \neq \mathrm{UpKnow}(K, [s']_{\sim E}, D')$; and $\delta^K((s, K, D), (\sigma_E, D'), \sigma_A)(s', K', D') = \delta(s, \sigma_E, \sigma_A)(s')$ otherwise: δ^K behaves as δ on the first components and *deterministically* updates both the knowledge and the information on the domain;
- $(s, K, D) \sim_E^K (s', K', D')$ if and only if $K = K'$ (implying $s \sim_E s'$) and $D = D'$: Eve observes her knowledge and the domain of her last distribution;
- $(s, K, D) \sim_A^K (s', K', D')$ if and only if $s \sim_A s'$.

The intuitive meaning of the enriched alphabet Σ'_E of Eve is that instead of choosing a distribution $d : \Sigma_E \to [0, 1]$ Eve makes the domain $Dom = \{s \in S \mid d(s) > 0\}$ of d explicit by choosing the distribution d^K where $d^K(s, D) = d(s)$ if $D = Dom$ and $d^K(s, D) = 0$ otherwise. We call such a distribution d^K **well-formed** (*i.e.* d^K is obtained from some distribution d as just explained) and, in the sequel, whenever referring to strategies of Eve in $\mathcal{A}^K$, we will mean functions from sequences of observations into *well-formed* distributions.

Consider an observation-based strategy φ for Eve in the arena $\mathcal{A}$. Then it can be converted into an observation-based strategy on the associated knowledge arena. For this, remark that in the knowledge arena, those states reachable from the initial state $(s_0, \{s_0\}, \emptyset)$ are of the form (s, K, D) with all states in K being equivalent with s with respect to $\sim_E$. Then one can define $\varphi^K((s_0, K_0, D_0)(s_1, K_1, D_1) \cdots (s_n, K_n, D_n))$ as d^K where $d = \varphi([s_0]_{\sim E}[s_1]_{\sim E} \cdots [s_n]_{\sim E})$ is the corresponding distribution given by φ. Note that φ^K is observation-based as, for all $0 \leq h \leq n$, $[s_h]_{\sim E}$ is uniquely defined from the K_h, that are observed by Eve in the knowledge arena.

Conversely, any observation-based strategy in the knowledge arena can be converted into an observation-based strategy in the original arena. Indeed, consider some observation-based strategy φ^K in the knowledge arena: it is a mapping from $(2^S \times 2^\Sigma)^*$ into $\mathcal{D}(\Sigma'_E)$ (the equivalent classes of the relation $\sim_E^K$ are, by definition, isomorphic with $2^S \times 2^\Sigma$). Now, note that Eve can, while playing in $\mathcal{A}$, remember the domain of the distributions she played and compute on the fly her current knowledge (applying function UpKnow to her previous knowledge and to the domain of the last distribution played): hence along a play $s_0 s_1 \cdots s_n$ she can compute the corresponding sequence $(K_0, D_0)(K_1, D_1) \cdots (K_n, D_n)$ of knowledge / domain. Now it suffices to consider the observation-based strategy φ for Eve in the initial arena defined by:

$$\varphi(s_0 s_1 \cdots s_n) = \varphi^K((K_0, D_0)(K_1, D_1) \cdots (K_n, D_n))$$

Note that this last transformation (taking a strategy φ^K and producing a strategy φ) is the inverse of the first transformation (taking a strategy φ and producing a strategy φ^K). In particular, it proves that the observation-based strategies in both arena are in bijection. It should be clear that those strategies for Adam in both games are the same (as what he observes is identical).

Assume that $\mathcal{A}$ is equipped with a set F of final states. Then one defines the final states in $\mathcal{A}^K$ by letting $F^K = \{(f, K, D) \mid f \in F\} \cap S^K$: this allows to define an objective $\mathcal{O}^K$ in $\mathcal{A}^K$ from an objective $\mathcal{O}$ in $\mathcal{A}$. Based on the previous observations, we derive the following.

Proposition 1. *Let $\mathbb{G} = (\mathcal{A}, s_0, \mathcal{O})$ be some imperfect information game equipped with a reachability (*resp.* saftey, Büchi, co-Büchi) objective. Let $\mathbb{G}^K = (\mathcal{A}^K, (s_0, \{s_0\}, \emptyset), \mathcal{O}^K)$ be the associated game played on the knowledge arena. Then for any strategies φ_E, φ_A for Eve and Adam, the following holds:*
$\mathrm{Pr}_{s_0}^{\varphi_E,\varphi_A}(\mathcal{O}) = \mathrm{Pr}_{(s_0,\{s_0\},\emptyset)}^{\varphi_E^K,\varphi_A}(\mathcal{O}^K)$*. In particular, Eve has an almost-surely winning observation-based strategy in $\mathbb{G}$ if and only if she has one in $\mathbb{G}^K$.*

In the setting of the previous proposition, consider the special case where Eve has an almost-surely winning observation-based strategy φ^K in $\mathbb{G}^K$ that only depends on the current knowledge (in particular, it is *memoryless*). Then the corresponding almost-surely winning observation-based strategy φ in $\mathbb{G}$ is, in general, not memoryless, but can be implemented by a finite transducer whose set of control states is precisely the set of possible knowledges for Eve. More precisely the strategy consists in computing and updating on the fly (using a finite automaton) the value of the knowledge after the current partial play and to pick the next action by solely considering the knowledge. We may refer at such a strategy φ as a **knowledge-only strategy**.

4 Decidability Results

4.1 Reachability Objectives

The main result of this section is the following.

Theorem 1. *For any reachability concurrent game with imperfect information, one can decide, in doubly exponential time, whether Eve has an almost-surely winning strategy. If Eve has such a strategy then she has a knowledge-only uniform strategy, and such a strategy can be effectively constructed.*

Before proving Theorem 1 we first establish an intermediate result. A concurrent game (with imperfect information) in which one player has only a single available action is what we refer as a $1\frac{1}{2}$**-player game with imperfect information** (those games are also known in the literature as *partially observable Markov Decision Processes*). The following result is a key ingredient for the proofs of Proposition 2 and Theorem 1.

Lemma 1. *Consider an $1\frac{1}{2}$-player safety game with imperfect information. Assume that the player has an observation-based strategy that is positively winning. Then she also has an observation-based finite memory strategy that is positively winning. Moreover, both the strategy and the set of positively winning states can be computed in time $\mathcal{O}(2^{|S|})$.*

Proof (Sketch). Consider the knowledge arena and call a knowledge K *surely winning* if the player has a knowledge based strategy that is surely winning from any (s, K, D) with $s \in K$ and $D \subseteq \Sigma_E$. We prove, that if the player has a positively winning strategy, then the set of winning knowledges is non empty and that it comes with a memoryless surely winning strategy (that consists in staying in the surely winning component). This set also contains at least a singleton $\{s\}$ (meaning that if the player knows that she is in s then she can surely win): call such states *s surely winning.* Then, one proves that positively winning states are exactly those that are connected (in the graph sense) to some surely winning state by a path made of *non-final* states. Hence a positively winning strategy consists in playing some initial actions randomly (trying to reach a surely winning state) and then in mimicking a knowledge-only surely winning strategy. Complexity comes with a fixpoint definition of the previous objects. □

Fix, for the rest of this section, a concurrent game with imperfect information $\mathbb{G} = (\mathcal{A}, s_0, \mathcal{O})$ equipped with a reachability objective $\mathcal{O}$ defined from a set F of final states. We set $\mathcal{A} = \langle S, \Sigma_E, \Sigma_A, \delta, \sim_E, \sim_A \rangle$. We also consider $\mathbb{G}^K = (\mathcal{A}^K, (s_0, \{s_0\}, \emptyset), \mathcal{O}^K)$ to be the corresponding knowledge game.

To prove Theorem 1, one first defines (in a non constructive way) a knowledge-only uniform strategy φ for Eve as follows. We let

$$\mathcal{K}^{\mathrm{AS}} = \{K \in 2^S \mid \exists \varphi_E \text{ knowledge-based strategy for Eve s.t. } \varphi_E \text{ is almost-surely winning for Eve in } \mathbb{G}^K \text{ from any } (s, K, D) \text{ with } s \in K \text{ and} D \subseteq \Sigma_E\}$$

be the set of knowledges made only by almost-surely winning states for Eve (note here that we require that the almost-surely winning strategy is the same for all configurations with the same knowledge).

One can prove that, from a configuration with knowledge $K \in \mathcal{K}^{\mathrm{AS}}$, Eve always has at least one action which ensures that she remains in $\mathcal{K}^{\mathrm{AS}}$, and we define φ as the knowledge-only uniform strategy that chooses at random one of these safe actions. The next proposition shows that φ is almost-surely winning for Eve.

Proposition 2. *The strategy φ is almost-surely winning for Eve from states whose Eve's knowledge is in $\mathcal{K}^{\mathrm{AS}}$.*

Proof (sketch). To prove that φ is almost-surely winning, one needs to prove that it is almost surely-winning against any strategy of Adam. However, once φ is fixed (and as it is a knowledge-only strategy), one gets $1\frac{1}{2}$-player game in which only Adam is making choices. Proving that φ is almost surely winning is therefore equivalent to proving that Adam cannot positively wins in this new game (for a safety objective). For this we use Lemma 1 to argue that it suffices to prove that φ is winning against any finite-memory strategy of Adam. This fact permits us to conclude. □

Now one can prove Theorem 1. First Eve almost-surely wins in $\mathbb{G}$ if and only if she almost-surely wins in $\mathbb{G}^K$ if and only if $\{s_0\} \in \mathcal{K}^{\mathrm{AS}}$, *i.e.* (using Proposition 2) if and only if Eve has a knowledge-only uniform strategy in $\mathbb{G}^K$. Now, to decide whether Eve almost-surely wins $\mathbb{G}$, it suffices to check, for any possible knowledge-only uniform strategy φ for her, whether it is almost-surely winning. Once φ is fixed, it leads, from Adam's point of view, to a $1\frac{1}{2}$-player safety game $\mathbb{G}_\varphi$ where the player positively wins if and only if φ is not almost-surely winning. Hence Lemma 1 implies that deciding whether φ is almost-surely winning can be done in time exponential in the size of $\mathbb{G}_\varphi$, which itself is of exponential size in $|S|$. Hence deciding whether a knowledge-only uniform strategy for Eve is winning can be done in doubly exponential time (in the size of $|S|$). The set of knowledge-only uniform strategies for Eve is finite and its size is doubly exponential in the size of the game. Hence the overall procedure, that tests every possible such strategies, requires doubly exponential time. As effectivity is immediate, this concludes the proof of Theorem 1.

The naive underlying algorithm of Theorem 1 turns out to be optimal.

Theorem 2. *Deciding whether Eve almost-surely wins a concurrent game with imperfect information is a* 2-ExpTime*-complete problem.*

Proof (sketch). The proof is a generalisation of a similar result given in [8] showing ExpTime-hardness of concurrent games *only one* player is imperfectly informed. The idea is to simulate an alternating exponential space Turing machine (without input). We design a game where the players describe the run of such a machine: transitions from existential (*resp.* universal) states are chosen by Eve (*resp.* Adam) and Adam is also in charge of describing the successive configurations of the machine. To prevent him from cheating, Eve can secretly mark a cell of the tape, and latter check whether it was correctly updated (if not she wins). As she cannot store the exact index of the cell (it is of exponential size), she could cheat in the previous phase: hence Adam secretly marks some bit and one recall the value of the corresponding bit of the index of the marked cell: this bit is checked when Eve claims that Adam cheated (if it is wrong then she is loosing). Eve also wins if the described run is accepting. Eve can also restart the computation whenever she wants (this is useful when she cannot prove that Adam cheated): hence if the machine accepts the only option for Adam is to cheat, and Eve will eventually catch him with probability one. Now if the machine does not accept, the only option for Eve is to cheat, but it will be detected with positive probability. □

4.2 Büchi Objectives

We now consider the problem of deciding whether Eve almost-surely wins a Büchi game. The results and techniques are similar to the one for reachability games. In particular, we need to establish the following intermediate result (the proof is very similar to the one of Lemma 1 except that now the winning states are those connected by *any kind* of path to a surely winning state).

Lemma 2. *Consider an $1\frac{1}{2}$-player co-Büchi game with imperfect information. Assume that the player has an observation-based strategy that is positively winning. Then she also has an observation-based finite memory strategy that is positively winning. Moreover, both the strategy and the set of positively winning states can be computed in time $\mathcal{O}(2^{|S|})$.*

From Lemma 2 and extra intermediate results we derive our main result. Again, the key idea is to prove that the strategy that plays randomly inside the almost-surely winning region is an almost-surely winning strategy.

Theorem 3. *For any Büchi concurrent game with imperfect information, one can decide, in doubly exponential time, whether Eve has an almost-surely winning strategy. If Eve has such a strategy then she has a knowledge-based uniform memoryless strategy, and such a strategy can be effectively constructed. The doubly exponential time complexity bound is optimal.*

5 Discussion

The main contribution of this paper is to prove that one can decide whether Eve has an almost-surely winning strategy in a concurrent game with imperfect information equipped with a reachability objective or a Büchi objective.

A natural question is whether this result holds for other objectives, in particular for co-Büchi objectives. In a recent work [3], Baier *et al.* established undecidability of the emptiness problem for probabilistic Büchi automata on infinite words. Such an automaton can be simulated by a $1\frac{1}{2}$-player imperfect information game: the states of the game are the one of the automaton, they are all equivalent for the player, and therefore an observation based strategy is an infinite word. Hence a pure (*i.e.* non-randomised) strategy in such a game coincide with an input word for the automaton. From this fact, Baier *et al.* derived that it is undecidable whether, in a $1\frac{1}{2}$-player co-Büchi game with imperfect information, Eve has an almost-surely winning *pure* strategy.

One can also consider the stochastic-free version of this problem (an arena is **deterministic** iff $\delta(q, \sigma_E, \sigma_A)(q') \in \{0, 1\}$ for all $q, q', \sigma_E, \sigma_A$) and investigate whether one can decide if Eve has an almost-surely winning strategy in a *deterministic* game equipped with a co-Büchi objective. We believe that the $1\frac{1}{2}$-player setting can be reduced to this new one, hence allowing to transfer undecidability results [12]. An even weaker model to consider is the stochastic-free model in which Adam has perfect information about the play [8].

It may happen that Eve has no almost-surely winning strategy while having a family $(\varphi_\varepsilon)_{0<\varepsilon<1}$ of strategies such that φ_ε ensures to win with probability at least $1 - \varepsilon$. Such a family is called **limit-surely winning**. Deciding existence of such families is a very challenging problem: indeed, in many practical situations, it is satisfying enough if one can control the risk of failing. Even if those questions have been solved for perfect information games [1], as far as we know, there has not been yet any result obtained in the imperfect information setting.

Even if the algorithms provided in this paper are "optimal", they are rather naive (checking all strategies for Eve may cost a lot in practice). Hence, one should look for fixpoint-based algorithms as the one studied in [8]: it would be of great help for a symbolic implementation, and it could also be a useful step toward a solution of the problem of finding limit-surely winning strategies. Note that there are already efficient techniques and tools for finding *sure* winning strategies in subclasses of concurrent games with imperfect information [6,5].

Acknowledgements. The authors want to thank the anonymous reviewers as well as Florian Horn for their useful comments on preliminary versions of this work.

References

1. de Alfaro, L., Henzinger, T.A.: Concurrent omega-regular games. In: Proceedings of LICS 2000, pp. 141–154 (2000)
2. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. Theoretical Computer Science 386(3), 188–217 (2007)
3. Baier, C., Bertrand, N., Größer, M.: On decision problems for probabilistic büchi automata. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 287–301. Springer, Heidelberg (2008)
4. Bertrand, N., Genest, B., Gimbert, H.: Qualitative Determinacy and Decidability of Stochastic Games with Signals. In: Proceedings of LICS 2009 (to appear, 2009)
5. Berwanger, D., Chatterjee, K., De Wulf, M., Doyen, L., Henzinger, T.A.: Alpaga: A tool for solving parity games with imperfect information. In: TACAS 2009. LNCS, vol. 5505, pp. 58–61. Springer, Heidelberg (2009)
6. Berwanger, D., Chatterjee, K., Doyen, L., Henzinger, T.A., Raje, S.: Strategy construction for parity games with imperfect information. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 325–339. Springer, Heidelberg (2008)
7. Chatterjee, K.: Stochastic ω-Regular Games. PhD thesis, University of California (2007)
8. Chatterjee, K., Doyen, L., Henzinger, T.A., Raskin, J.-F.: Algorithms for omega-regular games with imperfect information. Logical Methods in Computer Science 3(3) (2007)
9. Chatterjee, K., Henzinger, T.A.: Semiperfect-information games. In: Sarukkai, S., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 1–18. Springer, Heidelberg (2005)
10. Grädel, E., Thomas, W., Wilke, T. (eds.): Automata, Logics, and Infinite Games. LNCS, vol. 2500. Springer, Heidelberg (2002)
11. Gurevich, Y., Harrington, L.: Trees, automata, and games. In: Proceedings of STOC 1982, pp. 60–65 (1982)
12. Horn, F.: Private communication (February 2009)
13. Paz, A.: Introduction to probabilistic automata. Academic Press, New York (1971)
14. Ramadge, P.J., Wonham, W.M.: Supervisory Control of a Class of Discrete Event Processes. SIAM Journal on Control and Optimization 25, 206 (1987)
15. Reif, J.H.: The complexity of two-player games of incomplete information. Journal of Computer and System Sciences 29(2), 274–301 (1984)

Diagrammatic Confluence and Completion

Jean-Pierre Jouannaud[1] and Vincent van Oostrom[2,⋆]

[1] INRIA-Rocquencourt, France and Tsinghua University, Beijing, China
[2] University of Utrecht, The Netherlands

Abstract. We give a new elegant proof that decreasing diagrams imply confluence based on a proof reduction technique, which is then the basis of a novel completion method which proof-reduction relation transforms arbitrary proofs into rewrite proofs even in presence of non-terminating reductions. Unlike previous methods, no ordering of the set of terms is required, but can be used if available. Unlike ordered completion, rewrite proofs are closed under instantiation. Examples are presented, including Kleene's and Huet's classical examples showing that non-terminating local-confluent relations may not be confluent.

Keywords: rewriting, confluence, completion, decreasing diagrams.

1 Introduction

Confluence of a binary relation on a set is defined as the existence of two derivations reaching the same element for any two derivations issuing from a same element. In case the relation describes some non-deterministic intentional computation, confluence expresses that the associated extensional relation is functional.

There are two radically different methods for showing confluence of a binary relation: based on Newman's lemma, the first applies to terminating relations [10]; based on Hindley-Rosen's lemma, the second applies to non-terminating relations [11]. Both aim at restricting the confluence check to the so-called local peaks.

In a series of paper starting with his PhD thesis and ending with [13], Van Oostrom succeeded to capture both methods within a single *complete* method, thanks to the notion of *decreasing diagram* of a *labelled relation*. The goal of this paper is to investigate a completion procedure for non-terminating relations based on decreasing diagrams.

Our contributions are three: we first reformulate the diagrammatic confluence result as an abstract Church-Rosser property under the existence of decreasing diagrams for all peaks, for which we give an elegant proof in Section 3 which prepares the investigation of abstract diagrammatic completion in Section 4, our second contribution. Our method is illustrated with Kleene's and Huet's famous examples showing that the termination assumption in Newman's lemma is necessary for obtaining confluence. In section 5, our third contribution is a critical diagram lemma generalizing both Huet's lemma and Tait's approach. By incorporating it to diagrammatic completion, we obtain a completion method for non-terminating rewrite relations on terms which transforms an arbitrary equational proof into a rewrite proof which instances are themselves rewrite proofs, unlike what happens with ordered completion [5,2].

[⋆] This work was initiated while the second author was visiting the first at École Polytechnique with support of Digiteo-Labs. The authors thank the refrerees for their suggestions.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 212–222, 2009.

2 Preliminaries

We assume a set of *objects* $\mathcal{O}$ and a set of labels $\mathcal{L}$ equipped with a partial quasi-ordering $\trianglerighteq$ which strict part $\triangleright$ is well-founded, and add for convenience a label 0 incomparable to other labels. We use the letters l, m, n for labels, and α, β, γ for words over $\mathcal{L}$. We write $M \triangleright l$ (resp., $l \triangleright M$) for $m \triangleright l$ (resp., $l \triangleright m$) for all m in the multiset of labels M.

We consider (*rewrite*) relations generated by *labelled rewrite steps* of the form $s \longrightarrow^l t$ for $s, t \in \mathcal{O}$ and $l \in \mathcal{L}$, and may omit l, s or t. Given an arbitrary labelled rewrite step $\longrightarrow^l$, we denote its inverse by ${}^l\!\longleftarrow$, its reflexive closure by $=\!\!\!\longrightarrow^l$, its symmetric closure by $\overset{l}{\longleftrightarrow}$, its reflexive transitive closure by $\twoheadrightarrow^\alpha$ for some word-label α, and its reflexive, symmetric transitive closure, called *conversion*, by $\overset{\alpha}{\twoheadleftarrow\!\!\twoheadrightarrow}$. A conversion $\overset{\beta}{\twoheadleftarrow\!\!\twoheadrightarrow}$ is called a *subproof* of the conversion $\overset{\alpha\beta\gamma}{\twoheadleftarrow\!\!\twoheadrightarrow} = \overset{\alpha}{\twoheadleftarrow\!\!\twoheadrightarrow} \overset{\beta}{\twoheadleftarrow\!\!\twoheadrightarrow} \overset{\gamma}{\twoheadleftarrow\!\!\twoheadrightarrow}$. We use $\mathcal{L}(P)$ for the multiset of labels occuring in a conversion P.

The triple v, u, w is said to be a *local peak* if $v \; {}^l\!\longleftarrow u \longrightarrow^m w$, a *peak* if $v \; {}^\alpha\!\twoheadleftarrow u \twoheadrightarrow^\beta w$, and a *rewrite proof* if $v \twoheadrightarrow^\alpha t \; {}^\beta\!\twoheadleftarrow w$. The pair v, w (resp., the peak v, u, w) is *convertible* if $v \overset{\alpha}{\twoheadleftarrow\!\!\twoheadrightarrow} w$ and *joinable* if $v \twoheadrightarrow^\alpha t \; {}^\beta\!\twoheadleftarrow w$ for some t. The relation $\longrightarrow$ is said to be *locally confluent* if every local peak is joinable, *confluent* if every peak is joinable, and *Church-Rosser* if every convertible pair is joinable.

We refer to [3] and [4] for missing notations and definitions.

3 Abstract Diagrammatic Church-Rosser Properties

Given an arbitrary rewrite relation on $\mathcal{O}$, we first consider specific conversions called *local diagrams* and define the important subclass of decreasing (local) diagrams [12,13].

Definition 1 (Local diagrams). *A* local diagram D *is a pair made of a* local peak $D_{peak} = v \longleftarrow u \longrightarrow w$ *and a* conversion $D_{conv} = v \twoheadleftarrow\!\!\twoheadrightarrow w$. *We call* diagram rewriting *the rewrite relation $\Longrightarrow_{\mathcal{D}}$ on conversions generated by a set $\mathcal{D}$ of local diagrams, in which a local peak in a conversion $PD_{peak}Q$ is replaced by its associated conversion to yield $PD_{conv}Q$.*

Definition 2 (Decreasing diagrams [13]). *A local diagram with local peak $v \; {}^l\!\longleftarrow u \longrightarrow^m w$ is* decreasing *whenever its conversion is of the form $v \overset{\alpha}{\twoheadleftarrow\!\!\twoheadrightarrow} s \overset{m}{=\!\!\!\longrightarrow} s' \overset{\gamma}{\twoheadleftarrow\!\!\twoheadrightarrow} t' \overset{l}{\longleftarrow\!\!\!=} t \overset{\beta}{\twoheadleftarrow\!\!\twoheadrightarrow} w$, with labels in α (resp. β) strictly smaller than m (resp. l), and labels in γ strictly smaller than l or m.*

Our first important result is that diagram rewriting is terminating.

Theorem 1. *$\Longrightarrow_{\mathcal{D}}$ terminates for any set of local decreasing diagrams.*

The difficulty here is to define a measure on conversions that decreases when replacing a local peak by one of its associated conversions.

Definition 3 (Filtered conversions). *We define the* filtered proof *of a conversion P as the sequence of elementary steps:*

$$\begin{array}{lll}
\llbracket \xleftrightarrow{l} P\rrbracket = \llbracket P\rrbracket & \mathit{if} & \llbracket P\rrbracket = Q \xleftarrow{r} R \mathit{\ with\ } l \lhd r \\
\llbracket \xleftarrow{l} P\rrbracket = \xleftarrow{l}\llbracket P\rrbracket & \mathit{if} & \llbracket P\rrbracket \neq Q \xleftarrow{r} R \mathit{\ with\ } l \lhd r \\
\llbracket \xrightarrow{l} P\rrbracket = \xrightarrow{l} (\llbracket P\rrbracket \setminus l) & \mathit{if} & \llbracket P\rrbracket \neq Q \xleftarrow{r} R \mathit{\ with\ } l \lhd r
\end{array}$$

with : $nil \setminus l = nil$; $(\xleftrightarrow{m} P) \setminus l = (P \setminus l)$ *if* $l \rhd m$, *else* $\xleftrightarrow{m}(P \setminus m)$
An elementary step in P *is* visible *if it belongs to* $\llbracket P\rrbracket$, *otherwise* hidden.

Filtered proofs generalize the lexicographic maximal measure [12].

Definition 4. *A local peak* $v \longleftarrow u \longrightarrow w$ *is* visible, semi-visible *or* hidden *in a conversion* P *if its elementary steps* $v \longleftarrow u$ *and* $u \longrightarrow w$ *are both visible, one visible and one hidden, or both hidden.*

Definition 5. *The* visible surface $\mathcal{V}s(P)$ *of a conversion* P *is the multiset containing for each visible step* $\longrightarrow^{l}$ *(resp.* $\longleftarrow$ *) of* P*, the multiset* M_l *of labels labelling the visible steps to its left (resp. right).*

Definition 6. *A* semi-visible derivation *of* length k *in a conversion* P *is a subproof of the form* $v_0 \xleftrightarrow{M_0} u_1 \longrightarrow v_1 \xleftrightarrow{M_1} u_2 \ldots \xleftrightarrow{M_{k-1}} u_k \longrightarrow v_k \xleftrightarrow{M_k} u_{k+1}$*, or of the form* $u_{k+1} \xleftrightarrow{M_k} v_k \longleftarrow u_k \xleftrightarrow{M_{k-1}} \ldots u_2 \xleftrightarrow{M_1} v_1 \longleftarrow u_1 \xleftrightarrow{M_0} v_0$*, which steps* $u_i \longleftrightarrow v_i$ *are visible and steps* $v_i \xleftrightarrow{M_i} u_{i+1}$ *are hidden. The* head *and* tail *of a maximal semi-visible derivation* P *are the subproofs* $v_0 \xleftrightarrow{M_0} u_1$ *and* $v_k \xleftrightarrow{M_k} u_{k+1}$*. The* hidden subproof *of the derivation is the* k*-tuple* $(\xleftrightarrow{M_0}, \ldots, \xleftrightarrow{M_{k-1}})$*. The* hidden proof $\mathcal{H}p(P)$ *of* P *is the multiset of its hidden subproofs. The* hidden tail $\mathcal{H}t(P)$ *of* P *is the multiset of its tails.*

We are now ready for the complexity measure of a conversion:

Definition 7. *We define the* complexity $\mathcal{C}(P)$ *of a conversion* P *to be the quadruple* $(\mathcal{L}(\llbracket P\rrbracket), \mathcal{V}s(P), \mathcal{C}(\mathcal{H}p(P)), \mathcal{C}(\mathcal{H}t(P)))$*, the complexity of conversions being naturally extended to multisets, and to tuples and multisets thereof. Complexities are compared in the* conversion ordering*:*
$\succ\!\!\succ = (\rhd_{mul}, (\rhd_{mul})_{mul}, ((\succ\!\!\succ)_{lex})_{mul}, (\succ\!\!\succ)_{mul})_{lex}$.

A simple induction shows that the conversion ordering is well-founded for any set $\mathcal{D}$ of local decreasing diagrams. An induction on the length of conversions shows that diagram rewriting decreases their complexity.

3.1 The Church-Rosser Property

Because conversions in normal form cannot have a peak, the existence of local decreasing diagrams for all local peaks implies the Church-Rosser property, which itself implies van Oostrom's theorem that the labelled relation is confluent under this assumption. While the result itself is not new, since confluence and Church-Rosser are equivalent in this abstract setting, this new elegant proof based on a conversion reduction technique is a key stepping stone to the coming completion procedure.

Theorem 2. *A labelled relation which all local peaks have a decreasing diagram is Church-Rosser.*

Termination of the conversion rewriting relation is actually too strong, weak termination would suffice by ensuring that a conversion in normal form is obtained provided a strategy is used that leads to a normal form.

The Church-Rosser property is therefore decidable when there are finitely many local peaks and a finite search space of decreasing diagrams for each of them.

4 Abstract Diagrammatic Completion

Diagrammatic completion is a novel completion procedure which transforms a labelled relation $\longrightarrow$ on $\mathcal{O}$ into a new one which all local diagrams are decreasing, defining the same convertibility relation.

4.1 Completion Inference Rules

The inference rules for diagrammatic completion given at Figure 1 operate on pairs (R, E) of labelled rewrites and equalities. The main operation of diagrammatic completion is to associate to each local peak $u \,{}^{l}\!\!\longleftarrow s \longrightarrow^{m} v$ between rules in R the labelled equality $(u, \{l\}, m) = (v, \{m\}, l)$, describing which rewrites are allowed on u and v so as to maintain decreasingness when orientations or reductions take place. We call *labelled object* a triple (u, M, n), where u is an object, M is a multiset of labels, and n is a label, and *labelled equality* a (symmetric) pair $(u, M, n) = (u', M', n')$ of labelled

Generate:

$$\frac{(E; R)}{(E \cup \{(v, \{m\}, n) = (w, \{n\}, m)\}; R)}$$

$$\texttt{if } v \xleftarrow{m} u \xrightarrow{n} w$$

Delete:

$$\frac{(E; R \cup \{s \xrightarrow{k} s\}) \text{ or } (E \cup \{(s, M, n) = (s, M', n')\}; R)}{(E; R)}$$

Decrease:

$$\frac{(E \cup (u, L, l) = (v, M, m); R)}{(E \cup (u', L', l') = (v, M, m); R)}$$

$$\texttt{if } \begin{cases} (u, L, l) \overset{k \lhd L}{\Longrightarrow} (u', L', l') \text{ and} \\ u' = v \text{ or } m \neq 0 \text{ or } l' \neq 0 \text{ or} \\ \quad \exists k \lhd L' \uplus M \end{cases}$$

Label:

$$\frac{(E \cup \{(u, L, l) = (v, M, m)\}; R)}{(E; R \cup \{u \xrightarrow{k} v\})}$$

$$\texttt{if } \begin{cases} l \neq 0 \text{ and } l \unrhd k \text{ or} \\ l = m = 0 \text{ and } L \rhd k \end{cases}$$

Simplify Left:

$$\frac{(E; R \cup \{u \xrightarrow{k} v\})}{(E; R \cup \{u' \xrightarrow{k' \lhd k} v\})}$$

$$\texttt{if } u \xrightarrow{l} u' \text{ and } k \rhd l$$

Simplify Right:

$$\frac{(E; R \cup \{u \xrightarrow{k} v\})}{(E; R \cup \{u \xrightarrow{k' \lhd k} v'\})}$$

$$\texttt{if } v \xrightarrow{l} v' \text{ and } \{q \mid w \xrightarrow{q} v\} \rhd l$$

Fig. 1. Diagrammatic completion

objects. Rewriting a labelled object (u, M, n) may use a step which label is strictly smaller than M without restriction, as well as at most *one* rewrite step labelled with n (using the label 0 in the definition of $\Longrightarrow$ for that purpose), before being possibly further reduced by rewrite steps strictly smaller than M or n:

$$\begin{array}{lll} (u, M, n) \Longrightarrow^k (v, M, n) & \text{if } u \xleftrightarrow{k} v \text{ with } M \rhd k \\ (u, M, n) \Longrightarrow^n (v, M \cup \{n\}, 0) & \text{if } u =\!\!\!\longrightarrow^n v \end{array}$$

Unlike Knuth-Bendix completion, diagrammatic completion cannot fail, since equations which cannot be deleted can permanently be oriented by **Label**: the property is true of generated equations, and is carefully maintained by **Decrease**. Note also that the label 0 is never used, since incomparable to other labels. Further, the fact that a rule labelled k can be simplified by another rule of a lesser label l implies that **Simplify** cannot be blocked by the lack of a label strictly smaller than k. On the other hand, large labels may prevent potential simplifications.

It is easy to see that diagrammatic completion *truly extends* KB completion, by labelling each rewrite step $s \longrightarrow t$ with the term s it originates from, comparing labels in the order used in KB-completion: rewrite proofs become then particular decreasing diagrams, hence diagrammatic completion terminates whenever KB-completion does.

We illustrate diagrammatic completion first with Kleene's example showing that local-confluence does not imply confluence when termination is not satisfied and then with Huet's version of Kleene's counter-example[6], in which there is no cycle, but an infinite derivation.

Example 1 (Kleene). Let $\mathcal{F} = \{a, b, c, d\}$ and $\mathcal{R} = \{a \xrightarrow{1} c, a \xrightarrow{2} b, b \xrightarrow{3} a, b \xrightarrow{4} d\}$, using natural numbers for labels and the ordering on natural numbers as the ordering on labels (any labelling would do).

R	E	Inference	Details
$a \to^1 c, a \to^1 b,$ $b \to^1 a, b \to^3 d$		Generate	3rd & 4th
$a \to^1 c, a \to^1 b,$ $b \to^1 a, b \to^3 d$	$(a, \{1\}, 3) = (d, \{3\}, 1)$	Decrease	by none
$a \to^1 c, a \to^1 b,$ $b \to^1 a, b \to^3 d$	$(a, \{1, 3\}, 0) = (d, \{3\}, 1)$	Decrease	by 1st
$a \to^1 c, a \to^1 b,$ $b \to^1 a, b \to^3 d$	$(c, \{1, 3\}, 0) = (d, \{3\}, 1)$	Label	
$a \to^1 c, a \to^1 b$ $b \to^1 a, b \to^3 d$ $c \to^2 d$		Simplify Left	by 3rd
$a \to^1 c, a \to^1 b$ $b \to^1 a, a \to^2 d$ $c \to^2 d$		Simplify Left	
$a \to^1 c, a \to^1 b$ $b \to^1 a, b \to^1 d$ $c \to^2 d$		Done	

Note the use of **Decrease** to move the label 3 inside the multiset $\{1\}$ at the second step without doing any reduction, therefore allowing us to reduce a by the first rule at a subsequent step.

Unlike KB-completion, simplification is not enough for completing a set of non-terminating ground rules into a confluent one. Generation steps are necessary that would be considered as simplification steps in KB-completion. This is so because reductions are controlled by the labelling, not by an ordering on terms. For the very same reason, the set of rules obtained is not fully reduced, as it is the case for KB-completion. Obtaining an inter-irreducible set of rules can of course only be achieved if the labelling schema ensures termination of the rewrite relation itself.

Example 2 (Huet). $\mathcal{F} = \{a_i, b_i\}_{i\geq 0} \cup \{c, d\}$,
$R = \{a_i \xrightarrow{2i+1} c, b_i \xrightarrow{2i} d, b_i \xrightarrow{2i+3} a_{i+1}, a_i \xrightarrow{2i+2} b_i\}$ which is locally confluent, but not confluent. Diagrammatic completion generates the two infinite families of rules: $\{c \xrightarrow{2i+2} b_i, d \xrightarrow{2i+1} a_i\}$ making it confluent.

4.2 Completion Sequences

We proceed as usual by associating rewrite rules on conversions to the inference rules and proving that peaks are eventually eliminated by the conversion rewrite rules under a fairness assumption [7,1].

Definition 8. *A* (diagrammatic) completion procedure *takes as input a set of rewrites R_0 and an empty set of equalities E_0, and computes a* diagrammatic completion sequence $\{(R_i, E_i)\}_i$ *made of a set of rewrites R_i and a set of equalities E_i such that $(E_i; R_i) \vdash_{DC} (E_{i+1}; R_{i+1})$.*

Given a completion sequence $(\emptyset; R_0) \ldots \vdash^*_{DC} (E_n, R_n) \vdash^*_{DC} \ldots$, we define its limit $(E^*; R^*)$ and inductive limit $(E^\infty; R^\infty)$ as follows:
$E* = \bigcup_i E_i \quad R* = \bigcup_i R_i \quad E^\infty = \bigcup_i \bigcap_{j\geq i} E_i \quad R^\infty = \bigcup_i \bigcap_{j\geq i} R_i$

Rewrites in R^∞ and equalities in E^∞ are called *persisting*.

4.3 Completion Rewrite Rules

During a completion sequence, the sets of rewrites and equalities at step i keep evolving until they possibly stabilize. Given two convertible objects s, t, the conversion $s \overset{R_i \cup E_i}{\twoheadleftarrow\!\!\twoheadrightarrow} t$ is therefore in a constant state of flux, which we describe by rewrite rules on conversions built from rewrites in $R*$ and equalities in $E*$, expressing the action of the inference rules at that level.

Conversions are now sequences of elementary steps of two kinds: $s \overset{(M,n),(m,N)}{\longleftrightarrow} t$, for which M, n, m, N describe the simplifications allowed on the equality $(s, M, n) = (t, N, m)$, and $s \xrightarrow{k} t$ or $s \xleftarrow{k} t$, for which k is, as before, the label of the rewrite step. We shall abuse notation by writing $u \longleftrightarrow v \longleftrightarrow w$ for the concatenation $u \longleftrightarrow v$ $v \longleftrightarrow w$. $nil(s)$ will denote the conversion of length zero between s and itself, an identity for concatenation which will be eliminated whenever possible.

Rewrite rules on conversions are given at Figure 2; as usual, they imply the soundness of the (diagrammatic) completion inference rules.

Generation	$v \xleftarrow{m} u \xrightarrow{n} w$	$\Rightarrow$	$v \overset{(\{m\},n),(m,\{n\})}{\longleftrightarrow} w$
DeletionE	$v \overset{(M,n),(n',M')}{\longleftrightarrow} v$	$\Rightarrow$	$nil(v)$
DeletionR	$v \xrightarrow{l} v$	$\Rightarrow$	$nil(v)$
Decreasingness	$u \overset{(M,n),(n',M')}{\longleftrightarrow} v$	$\Rightarrow$	$u \overset{(M\uplus\{n\},0),(n',M')}{\longleftrightarrow} v$
Decreasingness	$u \overset{(M,n),(n',M')}{\longleftrightarrow} v$	$\Rightarrow$	$u \overset{k\lhd M}{\longleftrightarrow} u' \overset{(M,n),(n',M')}{\longleftrightarrow} v$
Labelling	$u \overset{(M,n),(n',M')}{\longleftrightarrow} v$	$\Rightarrow$	$u \xrightarrow{k} v$
		if $n \unrhd k$	
Simplification Left	$u \xrightarrow{k} v$	$\Rightarrow$	$u \xrightarrow{l} u' \xrightarrow{k'} v$
		if $k \rhd l$ and $k \rhd k'$	
Simplification Right	$u \xrightarrow{k} v$	$\Rightarrow$	$u \xrightarrow{k'} v' \xleftarrow{r} v$
		if $k \rhd r$ and $k \rhd k'$	

Fig. 2. Diagrammatic completion rules on proofs

4.4 Completeness of Fair Completion Sequences

In Knuth-Bendix completion, the proof rewrite rules terminate, and end up in a proof without peak under a fairness assumption. In diagrammatic completion, the proof rewrite rules do not terminate because **Decrease** may not terminate. Restricting simplification to rewriting would not solve the problem when rewriting is non-terminating. However, the fairness assumption ensures that every local peak is eventually converted into a decreasing diagram, hence fair sequences will terminate and end up in a proof without peak, providing us with the argument for showing completeness of diagrammatic completion. As for KB-completion, fairness is of course undecidable. Our notion of fairness is not the most general possible, but simple enough to ease the proofs. We shall later describe a fair, refined diagrammatic completion schema.

Definition 9 (Fairness). *A reduction sequence is fair if for every persisting local peak, there is a step j, such that the local peak is completed into a decreasing diagram at step j.*

Theorem 3. *Fair reduction sequences are* complete*: any proof $s \overset{R_0}{\twoheadleftarrow\!\!\twoheadrightarrow} t$ is eventually rewritten into a persistent rewrite proof $s \overset{R^\infty}{\twoheadrightarrow} u \overset{R^\infty}{\twoheadleftarrow} t$.*

4.5 Refined Fair Search

Given a local peak $v \xleftarrow{l_0} u \xrightarrow{m_0} w$, the non-deterministic search for a decreasing diagrams can by reduced by looking for *strictly decreasing rewrite diagrams*:

$$\begin{array}{rcl}
v = v_0 \xrightarrow{l_1} v_1 \ldots v_{n-1} \xrightarrow{l_n} v_n \overset{m_0=l'_0}{\Longrightarrow} & v'_0 \ldots v'_{n'-1} \xrightarrow{l'_{n'}} v'_{n'} \\
& \qquad\qquad\qquad\quad = \\
w = w_0 \xrightarrow{m_1} w_1 \ldots w_{p-1} \xrightarrow{m_p} w_p \overset{r=m'_0}{\Longrightarrow} & w'_0 \ldots w'_{p'-1} \xrightarrow{m'_{p'}} w'_{p'}
\end{array}$$

with: $\begin{cases} \forall i \in [0, n-1],\ l_i \rhd l_{i+1}; & \forall i \in [0, n'-1],\ l'_i \rhd l'_{i+1}; \\ \forall i \in [0, p-1],\ m_i \rhd m_{i+1}; & \forall i \in [0, p'-1],\ m'_i \rhd m'_{i+1}. \end{cases}$

This can be achieved by a simple modification of the definition of $\Longrightarrow$:

$$\begin{array}{ll} (u, \{i\}, n) \Longrightarrow^k (v, \{k\}, n) & \text{if } u \longrightarrow^k v \text{ with } i \rhd k \\ (u, \{i\}, n) \Longrightarrow^k (v, \{n\}, 0) & \text{if } u \mathrel{=\!\!\!\longrightarrow}^n v \end{array}$$

The search uses the well-founded ordering on labels to force termination of a particular search, fairness is therefore partially built-in. Our new fairness assumption is the following:

Definition 10. *A completion sequence searching for strictly decreasing rewrite diagrams is* strongly fair *if* **Generate** *eventually applies to every persisting local peak.*

Strongly fair completion sequences searching for strictly decreasing rewrite diagrams are fair, hence complete, because **Simplify** and **Decrease** terminate. The search can be constrained even further by looking for rewrite proofs, as in Knuth-Bendix completion, which appears as a particular instance of diagrammatic completion.

5 Labelled Rewriting on Terms

Abstract rewriting on a set of objects $\mathcal{O}$ uses labels in an abstract set $\mathcal{L}$. When rewriting terms on a signature $\mathcal{F}$, the **Generate** inference rule should restrict to the so-called *critical peaks* obtained by unifiying lefthand sides of rules. Non-critical diagrams should then be obtained from critical ones by instantiation and context application: these operations defined on terms must be lifted to labels which set must be an $\mathcal{F}$-algebra.

5.1 Algebras of Terms and Labels

We denote by $\mathcal{T}(\mathcal{F}, \mathcal{X})$ the free algebra of *terms* generated from $\mathcal{F}$ and a denumerable set $\mathcal{X}$ of variables. We use $\mathcal{V}ar(t)$ for the set of variables of the term t, $\mathcal{P}os(t)$ for the set of positions in t, and $\mathcal{FP}os(t)$ for its set of non-variable positions. The *subterm* of t at position p is denoted by $t|_p$, and we write $t[u]_p$ for the result of replacing $t|_p$ at position p in t by u.

Definition 11. *We assume given a homomorphism $\llbracket \_ \rrbracket$ from the initial algebra $\mathcal{T}(\mathcal{F})$ into an $\mathcal{F}$-algebra L which elements in the $\rhd$-well-founded carrier $\mathcal{L}$ are called* labels. *Given term v and context $u[\ ]_p$, we denote by $u[l]_p$ the (unique) label $\llbracket u[v]_p \rrbracket$ for some v such that $l = \llbracket v \rrbracket$. A* labelling *of a given rewrite relation $\longrightarrow$ on ground terms is a mapping ϕ from $\longrightarrow$ to $\mathcal{L}$ such that $\phi(u[s] \longrightarrow u[t])$ has label $u[l]$ if $\phi(s \longrightarrow t)$ has label l.*

The homomorphism $\llbracket \_ \rrbracket$ is extended to terms with variables, in such a way that it respects substitutions: variables in $\mathcal{X}$ are added to the carrier $\mathcal{L}$ which is then closed by the operations in the algebra that interpret the function symbols in $\mathcal{F}$, making it a $\mathcal{F} \cup X$-algebra, and the homomorphism itself is then defined as a function from $\mathcal{L}^n$ to $\mathcal{L}$ which acts as the identity on variables. It is then folklore that $\llbracket u\sigma \rrbracket$ is obtained by evaluating the application of $\llbracket u \rrbracket$ to $\llbracket \sigma \rrbracket$.

5.2 Labelled Rewrite Systems

Definition 12. *Given a set of labelled rewrite rules $R = \{l_i \xrightarrow{m_i} r_i\}_{i \in I}$, where l_i, r_i, m_i are respectively the lefthand side, the righthand side and the label of the ith rule in R, the term u* rewrites *to the term v* with label *m if $u|_p = l_i\sigma$ for some position $p \in \mathcal{P}os(u)$ and index $i \in I$, $v = u[r_i\sigma]$ and $m = u[m_i\sigma]$, which we write $u \xrightarrow{p,m} v$. We may sometimes omit p, m.*

Definition 13. *A labelled rewrite system R is*

1. self-labelling *[13] if $s \longrightarrow^l t \longrightarrow^m u$ implies $l \rhd m$;*
2. rule-labelling *if labels are natural numbers.*

Both labellings can be equipped with a structure of $\mathcal{F}$-algebra. A rewrite relation can be made into a self-labelling relation iff it is terminating, showing that the *canonical* self-labelling of a step $u \longrightarrow v$ is by the rewritten term u. Rule-labelling targets non-terminating rewrite relations generated by ordered sets of rewrite rules. Be aware that different rules may have the same number as label.

5.3 Critical Diagrams

Our goal now is to restrict the set of local peaks to be checked in order to ensure confluence to those which are minimal with respect to instantiation and context application. The principles of such an analysis are well-known: there are three kinds of peaks to be distinguished, disjoint peaks when two rules apply at disjoint positions, ancestor peaks when a rule applies above the other without overlapping it, and critical peaks when both rules overlap. Unfortunately, there is no hope to have results that hold for arbitrary labellings. Disjoint peaks commute for self- and rule-labelling. Ancestor peaks, unfortunately, do not commute for rule-labelling unless rules are both left- and right-linear.

We now recall the usual notion of critical pair:

Definition 14. *Given a labelled rewrite system R, a labelled rule $l \xrightarrow{m} r$, a labelled rule $g \xrightarrow{n} d$ and a position $p \in \mathcal{FP}os(l)$ such that the equational problem $l|_p = g$ has a most general unifier σ, then the proof $r\sigma \xleftarrow{m} l\sigma \xrightarrow{n} l\sigma[d\sigma]_p$ is called a* critical peak *of the rule $g \xrightarrow{n} d$ onto the rule $l \xrightarrow{m} r$ at positions p, which associated* critical pair *is $(r\sigma, l\sigma[d\sigma]_p)$.*

We now come to the main result of this section, showing that the search for decreasing diagrams can be restricted to critical peaks of various kinds. To this end, we assume that the set of rewrite rules R is the union of two subsets R_1 and R_2. We assume that R_1 is a self-labelling set of rewrite rules, that R_2 is a set of order-labelling linear rewrite rules, and consider the relation $\longrightarrow^{R_1} \cup \longrightarrow^{R_2}$.

To work with the relation $\longrightarrow^{R_1} \cup \longrightarrow^{R_2}$, we blend the labellings $\rhd_1$ and $\rhd_2$ used for R_1 and R_2 respectively to proofs using steps with R_1 and steps with R_2 as follows: a rewrite step $s \longrightarrow^l t$ at position p with R_i is labelled by the pair $(i, l)\}$. Pairs are compared in the well-founded ordering $\succ := (2 > 1, \rhd_1 \cup \rhd_2)_{lex}$. Tedious checking yields:

Theorem 4 (Critical diagrammatic lemma). *Let $R = R_1 \uplus R_2$ be a rewrite system satisfying the assumptions above. Assume that critical peaks of rules in R have decreasing diagrams for $\succ$. Then, local peaks of rules in R have decreasing diagrams for $\succ$.*

Note that the ordering on triples reduces to the component orderings when checking the diagrams for local peaks in R_1 or in R_2.

As a corollary of theorems 2 and 4, we get:

Corollary 1. *Under our assumptions, $\longrightarrow^{R_1} \cup \longrightarrow^{R_2}$ is Church-Rosser.*

Generalizing the above results to left-linear (but right non-linear) rules in R_2 is a non-trivial matter. It is of course well-known that right-linearity can be taken care of by using parallel rewriting, to the price of more complex technicalities. We have not succeeded yet to accomodate these technicalities to the present framework.

6 Conclusion

We have described a novel extension of Knuth and Bendix completion procedure, *diagrammatic completion*, which accepts non-terminating relations as input, and returns as output a relation which has the Church-Rosser property. Unlike ordered completion, diagrammatic completion is a true generalization of Knuth-Bendix completion. In particular, they coincide for self-labelling rewrite relations, assuming **Decrease** is constrained so as to search for rewrite proofs. The properties of diagrammatic completion are based on the critical diagrammatic lemma, which appears to be a powerful tool for checking a relation for local confluence, a statement that is to be confirmed: first, by a generalization to the case where R_2 contains rules which are not right-linear; second by an automatic confluence checker. Generalizations to rewriting modulo [9] and normal rewriting [8] remain to be investigated.

References

1. Bachmair, L., Dershowitz, N., Hsiang, J.: Orderings for equational proofs. In: Proc. 1st IEEE Symp. Logic in Computer Science, Camb., Mass., June 1986, pp. 346–357 (1986)
2. Bachmair, L., Dershowitz, N., Plaisted, D.A.: Completion without failure. In: Kaci, A.H., Nivat, M. (eds.) Resolution of Equations in Algebraic Structures. Rewriting Techniques, vol. 2, pp. 1–30. Academic Press, New York (1989)
3. Dershowitz, N., Jouannaud, J.-P.: Rewrite systems. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 243–309. North-Holland, Amsterdam (1990)
4. Klop, J.W., et al.: Term rewriting systems. Cambridge Tracts in Theoretical Computer Science, vol. 55. Cambridge University Press, Cambridge (2003)
5. Hsiang, J., Rusinowitch, M.: On word problems in equational theories. In: 14th International Colloquium on Automata, languages and programming, London, UK, pp. 54–71. Springer, London (1987)
6. Huet, G.: Confluent reductions: abstract properties and applications to term rewriting systems. Journal of the ACM 27(4), 797–821 (1980)

7. Huet, G.: A complete proof of correctness of the Knuth-Bendix completion algorithm. Journal of Computer and System Sciences 23, 11–21 (1981)
8. Jouannaud, J.-P.: Church-Rosser properties of normal rewriting, draft (2009)
9. Jouannaud, J.-P., Kirchner, H.: Completion of a set of rules modulo a set of equations. SIAM Journal on Computing 15(4), 1155–1194 (1986)
10. Newman, M.H.A.: On theories with a combinatorial definition of 'equivalence'. Ann. Math. 43(2), 223–243 (1942)
11. Barry, K.R.: Tree-manipulating systems and Church-Rosser theorems. J. of the Association for Computing Machinery 20(1), 160–187 (1973)
12. van Oostrom, V.: Confluence for Abstract and Higher-Order Rewriting. PhD thesis, Vrije Universiteit, Amsterdam (1994)
13. van Oostrom, V.: Confluence by decreasing diagrams. In: Voronkov, A. (ed.) RTA 2008. LNCS, vol. 5117, pp. 306–320. Springer, Heidelberg (2008)

Complexity of Model Checking Recursion Schemes for Fragments of the Modal Mu-Calculus

Naoki Kobayashi[1] and C.-H. Luke Ong[2]

[1] Tohoku University
[2] University of Oxford

Abstract. Ong has shown that the modal mu-calculus model checking problem (equivalently, the alternating parity tree automaton (APT) acceptance problem) of possibly-infinite ranked trees generated by order-n recursion schemes is n-EXPTIME complete. We consider two subclasses of APT and investigate the complexity of the respective acceptance problems. The main results are that, for APT with a single priority, the problem is still n-EXPTIME complete; whereas, for APT with a disjunctive transition function, the problem is $(n-1)$-EXPTIME complete. This study was motivated by Kobayashi's recent work showing that the resource usage verification for functional programs can be reduced to the model checking of recursion schemes. As an application, we show that the resource usage verification problem is $(n-1)$-EXPTIME complete.

1 Introduction

The model checking problem for higher-order recursion schemes has been a topic of active research in recent years (for motivation as to why the problem is interesting, see e.g. the introduction of Ong's paper [1]). This paper studies the complexity of the problem with respect to certain fragments of the modal μ-calculus. A higher-order recursion scheme (recursion scheme, for short) is a kind of (deterministic) grammar for generating a possibly-infinite ranked tree. The model checking problem for recursion schemes is to decide, given an order-n recursion scheme $\mathcal{G}$ and a specification ψ for infinite trees, whether the tree generated by $\mathcal{G}$ satisfies ψ. Ong [1] has shown that if ψ is a modal μ-calculus formula (or equivalently, an alternating parity tree automaton), then the model checking problem is n-EXPTIME complete.

Following Ong's work, Kobayashi [2] has recently applied the decidability result to the model checking of higher-order functional programs (precisely, programs of the simply-typed λ-calculus with recursion and resource creation/access primitives). He considered the *resource usage verification problem* [3]—the problem of whether programs access dynamically created resources in a valid manner (e.g. whether every opened file will eventually be closed, and thereafter never read from or written to before it is reopened). He showed that the resource usage verification problem reduces to a model checking problem for recursion schemes

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 223–234, 2009.

by giving a transformation that, given a functional program, constructs a recursion scheme that generates all possible resource access sequences of the program. From Ong's result, it follows that the resource usage verification problem is in n-EXPTIME (where, roughly, n is the highest order of types in the program). This result also implies that various other verification problems, including (the precise verification of) reachability ("Given a closed program, does it reach the fail command?") and flow analysis ("Does a sub-term e evaluate to a value generated at program point l?"), are also in n-EXPTIME, as they can be easily recast as resource usage verification problems.

It was however unknown whether n-EXPTIME is the tightest upper-bound of the resource usage verification problem. Although the model checking of recursion schemes is n-EXPTIME-hard for the full modal μ-calculus, only a certain fragment of the modal μ-calculus is used in Kobayashi's approach to the resource usage verification problem. First, specifications are restricted to safety properties, which can be described by Büchi tree automata with a trivial acceptance condition (the class called "trivial automata" by Aehlig [4]). Secondly, specifications are also restricted to linear-time properties—the branching structure of trees is ignored, and only the path languages of trees are of interest. Thus, one may reasonably hope that there is a more tractable model checking algorithm than the n-EXPTIME algorithm.

The goal of this paper is, therefore, to study the complexity of the model checking of recursion schemes for various fragments of the modal μ-calculus (or, alternating parity tree automata) and to apply the result to obtain tighter bounds of the complexity of the resource usage verification problem.

The main results of this paper are as follows:

– The problem of whether a given Büchi automaton with a trivial acceptance condition (or, equivalently, alternating parity tree automaton with a single priority 0) accepts the tree generated by an order-n recursion scheme is still n-EXPTIME-hard. This follows from the n-EXPTIME-completeness of the word acceptance problem for higher-order alternating pushdown automata[1] [5].

– We introduce a new subclass of alternating parity tree automata (APT) called *disjunctive APT*, and show that its acceptance problem for trees generated by order-n recursion schemes is $(n-1)$-EXPTIME complete. From this general result, it follows that both the linear-time properties (including reachability, which is actually $(n-1)$-EXPTIME complete) and finiteness of the tree generated by a recursion scheme are $(n-1)$-EXPTIME.

– As an application, we show that the resource usage verification problem [2] is also $(n-1)$-EXPTIME-complete, where n is the highest order of types used in the source program (written in an appropriate language [2]).

Related Work. For the class of Büchi automata with a trivial acceptance condition, Kobayashi [2] showed that the complexity is linear in the size of recursion schemes, if the sizes of types and automata are bounded above by a

[1] Engelfriet's proof [5] is for a somewhat different—but equivalent—machine which is called *iterated pushdown automaton*.

constant. For the full modal μ-calculus, Kobayashi and Ong [6] have shown that the complexity is polynomial-time in the size of the recursion scheme, assuming that the size of types and the formula are bounded above by a constant.

2 Preliminaries

We assume the standard notions of (ranked/unranked) infinite trees [1].

Higher-Order Recursion Schemes. The set of *types* is defined by: $\kappa ::= \mathsf{o} \mid \kappa_1 \to \kappa_2$, where o describes trees. The *order* of κ, written $\mathit{order}(\kappa)$, is defined by: $\mathit{order}(\mathsf{o}) := 0$ and $\mathit{order}(\kappa_1 \to \kappa_2) := \mathit{max}(\mathit{order}(\kappa_1) + 1, \mathit{order}(\kappa_2))$. A (deterministic) higher-order recursion scheme (recursion scheme, for short) is a quadruple $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, where (i) Σ is a *ranked alphabet* i.e. a map from a finite set of symbols called *terminals* to types of order 0 or 1. (ii) $\mathcal{N}$ is a map from a finite set of symbols called *non-terminals* to types. (iii) $\mathcal{R}$ is a set of rewrite rules $F\,\widetilde{x} \to t$. Here, $\widetilde{x}$ abbreviates a sequence of variables, and t is an applicative term constructed from non-terminals, terminals, and variables. (iv) S is a *start symbol.* We require that $\mathcal{N}(S) = \mathsf{o}$. The set of (typed) terms is defined in the standard manner: A symbol (i.e., a terminal, non-terminal, or variable) of type κ is a term of type κ. If terms t_1 and t_2 have types $\kappa_1 \to \kappa_2$ and κ_1 respectively, then $t_1\ t_2$ is a term of type κ_2. For each rule $F\,\widetilde{x} \to t$, $F\,\widetilde{x}$ and t must be terms of type o. There must be exactly one rewrite rule for each non-terminal. The *order* of a recursion scheme is the highest order of its non-terminals.

A rewrite relation on terms is defined inductively by: (i) If $F\,\widetilde{x} \to t \in \mathcal{R}$, then $F\,\widetilde{s} \longrightarrow_{\mathcal{G}} [\widetilde{s}/\widetilde{x}]t$. (ii) If $t \longrightarrow_{\mathcal{G}} t'$, then $t\,s \longrightarrow_{\mathcal{G}} t'\,s$ and $s\,t \longrightarrow_{\mathcal{G}} s\,t'$. The *value tree* of a recursion scheme $\mathcal{G}$, written $[\![\mathcal{G}]\!]$, is the (possibly infinite) tree obtained by infinite rewriting of the start symbol S: See [1] for a precise definition.

Alternating parity tree automata. Given a finite set X, the set $\mathsf{B}^+(X)$ of *positive Boolean formulas* over X is defined as follows:

$$\mathsf{B}^+(X) \ni \theta\ ::=\ \mathsf{t} \mid \mathsf{f} \mid x \mid \theta \wedge \theta \mid \theta \vee \theta$$

where x ranges over X. We say that a subset Y of X *satisfies* θ just if assigning true to elements in Y and false to elements in $X \setminus Y$ makes θ true.

An *alternating parity tree automaton* (or APT for short) over Σ-labelled trees is a tuple $\mathcal{A} = (\Sigma, Q, \delta, q_I, \Omega)$ where (i) Σ is a ranked alphabet; let m be the largest arity of the terminal symbols; (ii) Q is a finite set of states, and $q_I \in Q$ is the initial state; (iii) $\delta : Q \times \Sigma \longrightarrow \mathsf{B}^+(\{1, \ldots, m\} \times Q)$ is the transition function where, for each $f \in \Sigma$ and $q \in Q$, we have $\delta(q, f) \in \mathsf{B}^+(\{1, \ldots, \mathit{arity}(f)\} \times Q)$; and (iv) $\Omega : Q \longrightarrow \{0, \cdots, M-1\}$ is the priority function.

A *run-tree* of an APT $\mathcal{A}$ over a Σ-labelled ranked tree T is a $(\mathit{dom}(T) \times Q)$-labelled unranked tree r satisfying: (i) $\epsilon \in \mathit{dom}(r)$ and $r(\epsilon) = (\epsilon, q_I)$; and (ii) for every $\beta \in \mathit{dom}(r)$ with $r(\beta) = (\alpha, q)$, there is a set S that satisfies $\delta(q, T(\alpha))$; and for each $(i, q') \in S$, there is some j such that $\beta\,j \in \mathit{dom}(r)$ and $r(\beta\,j) = (\alpha\,i, q')$.

Let $\pi = \pi_1\,\pi_2 \cdots$ be an infinite path in r; for each $i \geq 0$, let the state label of the node $\pi_1 \cdots \pi_i$ be q_{n_i} where q_{n_0}, the state label of ϵ, is q_I. We say that

π satisfies the *parity* condition just if the largest priority that occurs infinitely often in $\Omega(q_{n_0})\,\Omega(q_{n_1})\,\Omega(q_{n_2})\cdots$ is even. A run-tree r is *accepting* if every infinite path in it satisfies the parity condition. An APT $\mathcal{A}$ accepts a (possibly infinite) ranked tree T if there is an accepting run-tree of $\mathcal{A}$ over T.

Ong [1] has shown that there is a procedure that, given a recursion scheme $\mathcal{G}$ and an APT $\mathcal{A}$, decides whether $\mathcal{A}$ accepts the value tree of $\mathcal{G}$.

Theorem 1 (Ong). *Let $\mathcal{G}$ be a recursion scheme of order n, and $\mathcal{A}$ be an APT. The problem of deciding whether $\mathcal{A}$ accepts $[\![\mathcal{G}]\!]$ is n-EXPTIME-complete.*

3 Trivial APT and the Complexity of Model Checking

APT with a trivial acceptance condition, or *trivial APT* (for short), is an APT that has exactly one priority which is even. Note that trivial APT are equivalent to Aehlig's "trivial automata" [4] (for defining languages of ranked trees).

The first result of this paper is a logical characterization of the class of ranked trees accepted by trivial APT. Call $\mathcal{S}$ the following "safety fragment" of the modal mu-calculus:

$$\phi, \psi \ ::= \ P_f \mid Z \mid \phi \wedge \psi \mid \phi \vee \psi \mid \langle i \rangle \phi \mid \nu Z.\phi$$

where f ranges over symbols in Σ, and i ranges over $\{1, \cdots, \mathit{arity}(\Sigma)\}$.

Proposition 1 (Equi-Expressivity). *The logic $\mathcal{S}$ and trivial APT are equivalent for defining possibly-infinite ranked trees. I.e. for every closed $\mathcal{S}$-formula, there is a trivial APT that defines the same tree language, and vice versa.*

3.1 n-EXPTIME Completeness

We show that the model checking problem for recursion schemes is n-EXPTIME complete for trivial APT. The upper-bound of n-EXPTIME follows immediately from Ong's result [1]. To show the lower-bound, we reduce the decision problem of $w \overset{?}{\in} \mathcal{L}(\mathcal{A})$, where w is a word and $\mathcal{A}$ is an order-n alternating PDA, to the model checking problem for recursion schemes. n-EXPTIME hardness follows from the reduction, since the problem of $w \overset{?}{\in} \mathcal{L}(\mathcal{A})$ is n-EXPTIME hard [5].

Definition 1. An *order-n alternating PDA* (order-n APDA, for short) for finite words is a 7-tuple:

$$\mathcal{A} \ = \ \langle P,\ \lambda,\ p_0 \in P,\ \Gamma,\ \Sigma,\ \Delta \subseteq P \times \Gamma \times (\Sigma \cup \{\epsilon\}) \times P \times \mathit{Op}_n,\ F \subseteq P \rangle$$

where P is a set of states, $\lambda \in P \to \{\mathtt{A}, \mathtt{E}\}$, p_0 is the initial state, Γ is the set of stack symbols, Σ is an input alphabet, F is the set of final states, and Δ is a transition relation that satisfies: for every p, γ, if $(p, \gamma, \epsilon, p', \theta) \in \Delta$ for some p', θ, then $(p, \gamma, a, p', \theta) \notin \Delta$ for every $a \in \Sigma, p'$ and θ. A *configuration* of an order-n APDA is of the form (p, s) where s is an order-n stack: an order-1 stack is an ordinary stack, and an order-$(k+1)$ stack is a stack of order-k stacks. The (induced) transition relation on configurations is defined by the rule:

$$\text{if } (p, top_1(s), \alpha, p', \theta) \in \Delta, \text{ then } (p, s) \longrightarrow_\alpha (p', \theta(s)).$$

Here, $\theta \in Op_n$ is an order-n stack operation and $top_1(s)$ is the stack top of s. The definition of the stack operations Op_n is omitted since it is not important for understanding the encodings below; interested readers may wish to consult, for example, the paper [8] by Knapik et al.

Let w be a word over Σ. We write w_i $(0 \le i < |w|)$ for the i-th element of w. A *run tree* of an order-n APDA over a word w is a *finite*, unranked tree such that (i) The root is labelled by $(p_0, \perp_n, 0)$, where $\perp_n$ is the initial stack. (ii) If a node is labelled by (p, s, i) and $\lambda(p) = \mathtt{A}$, then either $p \in F$ and $i = |w|$, or the set of labels of the child nodes is exactly $\{(p', \theta(s), i{+}1) \mid (p, top_1(s), w_i, p', \theta) \in \Delta \wedge i < |w|\} \cup \{(p', \theta(s), i) \mid (p, top_1(s), \epsilon, p', \theta) \in \Delta\}$. (Thus, if the set is empty, the node has no child.) (iii) If a node is labelled by (p, s, i) and $\lambda(p) = \mathtt{E}$, then either $p \in F$ and $i = |w|$, or there exists exactly one child node which is labelled by an element of the set: $\{(p', \theta(s), i+1) \mid (p, top_1(s), w_i, p', \theta) \in \Delta \wedge i < |w|\} \cup \{(p', \theta(s), i) \mid (p, top_1(s), \epsilon, p', \theta) \in \Delta\}$. An order-$n$ APDA $\mathcal{A}$ *accepts* w if there exists a run tree of $\mathcal{A}$ over w.

Engelfriet [5] has shown that the word acceptance problem for order-n APDA is n-EXPTIME complete.

Theorem 2 (Engelfriet). *Let $\mathcal{A}$ be an order-n APDA and w a finite word over Σ. The problem of $w \overset{?}{\in} \mathcal{L}(\mathcal{A})$ is n-EXPTIME complete.*

To reduce the word acceptance problem of order-n APDA to the model checking problem for recursion schemes, we use the equivalence [8] between order-n *safe* recursion schemes and order-n PDA as (deterministic) devices for generating trees.

Definition 2. An *order-n tree-generating (deterministic) PDA* is a 5-tuple $\langle \Sigma, \Gamma, Q, \delta, q_0 \rangle$ where Σ is a ranked alphabet, Γ is a finite stack alphabet, Q is a finite state-set, $\delta : Q \times \Gamma \longrightarrow (Q \times Op_n + \{(f; q_1, \cdots, q_{arity(f)}) : f \in \Sigma, q_i \in Q\})$ is the transition function, and $q_0 \in Q$ is the initial state. A *generalized configuration* is either a configuration (which has the shape (q, s) where s is an order-n stack over Γ) or a triple of the form $(f; q_1, \cdots, q_{arity(f)}; s)$. We define $\overset{\ell}{>}$, a labelled transition relation over generalized configurations, as follows:

- $(q, s) \overset{(q',\theta)}{>} (q', \theta(s))$ if $\delta(q, top_1(s)) = (q', \theta)$
- $(q, s) \overset{f\,\widetilde{q}}{>} (f; \widetilde{q}; s)$ if $\delta(q, top_1\, s) = (f; \widetilde{q})$
- $(f; \widetilde{q}; s) \overset{(f,i)}{>} (q_i, s)$ for each $1 \le i \le arity(f)$.

A *computation path* of an order-n PDA $\mathcal{A}$ is a finite or infinite transition sequence $\rho = c_0 \overset{\ell_0}{>} c_1 \overset{\ell_1}{>} c_2 \overset{\ell_2}{>} \cdots$ where each c_i is a generalized configuration, and

$c_0 = (q_0, \bot_n)$ is the initial configuration. The *Σ-projection* of ρ is the subsequence $\ell_{r_1}\,\ell_{r_2}\,\ell_{r_3}\,\cdots$ of labels of the shape (f, i) (in which case $arity(f) > 0$) or f (i.e. $f\,\epsilon$, in which case $arity(f) = 0$, and the label marks the end of the Σ-projection). We say the PDA $\mathcal{A}$ *generates* the Σ-labelled tree t just if the *branch language*[2] of t coincides with the set of Σ-projection of computation paths of $\mathcal{A}$.

Theorem 3 (Knapik et al. [8]). *There exists a reduction of an order-n tree-generating PDA $\mathcal{M}$ to an order-n safe recursion scheme $\mathcal{G}$ that generates the same tree as $\mathcal{M}$. Moreover, both the running time of the reduction algorithm and the size of $\mathcal{G}$ are polynomial in the size of $\mathcal{M}$.*

By Theorems 2 and 3, it suffices to show that, given a word w and an order-n APDA $\mathcal{A}$, one can construct an order-n tree-generating PDA $\mathcal{M}_{\mathcal{A},w}$ and a trivial APT $\mathcal{B}_{\mathcal{A}}$ such that w is accepted by $\mathcal{A}$ if, and only if, the tree generated by $\mathcal{M}_{\mathcal{A},w}$ is accepted by $\mathcal{B}_{\mathcal{A}}$.

Let w be a word over Σ. We write w_i ($i \in \{0, \ldots, |w|-1\}$) for the i-th element of w. From w and $\mathcal{A} = \langle P, \lambda, p_0, \Gamma, \Sigma, \Delta, F\rangle$ above, we construct an order-k PDA $\mathcal{M}_{\mathcal{A},w}$ for generating a $\{\mathtt{A}, \mathtt{E}, \mathtt{R}, \mathtt{T}\}$-labelled tree, which expresses a kind of run tree of $\mathcal{A}$ over the input word w. The node label $\mathtt{A}$ ($\mathtt{E}$, resp) means that $\mathcal{A}$ is in a universal (existential, resp.) state; $\mathtt{T}$ means that $\mathcal{A}$ has accepted the word, and $\mathtt{R}$ means that $\mathcal{A}$ is stuck (having no outgoing transition).

Let $N := max_{q\in P, a\in\Sigma\cup\{\epsilon\}, \gamma\in\Gamma}|\{q', \theta) \mid (q, \gamma, a, q', \theta) \in \Delta\}|$. I.e. N is the degree of non-determinacy of $\mathcal{A}$. We define $\mathcal{M}_{\mathcal{A},w} := \langle\{\mathtt{A}, \mathtt{E}, \mathtt{T}, \mathtt{R}\}, \Gamma, Q, \delta, (p_0, 0)\rangle$ where:

– The ranked alphabet is $\{\mathtt{A}, \mathtt{E}, \mathtt{T}, \mathtt{R}\}$, where the arities of $\mathtt{A}$ and $\mathtt{E}$ are N, and those of $\mathtt{T}$ and $\mathtt{R}$ are 0.

– $Q = (P \times \{0, \ldots, |w|\}) \cup \{q_\top, q_\bot\} \cup (P \times \{0, \ldots, |w|\} \times Op_n)$

– $\delta : Q \times \Gamma \longrightarrow (Q \times Op_n + \{(g; \widetilde{q}) : g \in \{\mathtt{A}, \mathtt{E}, \mathtt{T}, \mathtt{R}\}, q_i \in Q\})$ is given by:

(1) $\delta((p, |w|), \gamma) = (\mathtt{T}; \epsilon)$, if $p \in F$

(2) $\delta((p, i), \gamma) = (\mathtt{A}; (p_1, j_1, \theta_1), \ldots, (p_m, j_m, \theta_m), \underbrace{q_\top, \ldots, q_\top}_{N-m})$

if $\lambda(p) = \mathtt{A}$ and $\{(p_1, j_1, \theta_1), \ldots, (p_m, j_m, \theta_m)\}$ is:
$\{(p', i+1, \theta) \mid (p, \gamma, w_i, p', \theta) \in \Delta \wedge i < |w|\} \cup \{(p', i, \theta) \mid (p, \gamma, \epsilon, p', \theta) \in \Delta\}$

(3) $\delta((p, i), \gamma) = (\mathtt{E}; (p_1, j_1, \theta_1), \ldots, (p_m, j_m, \theta_m), q_\bot, \ldots, q_\bot)$

if $\lambda(p) = \mathtt{E}$ and $\{(p_1, j_1, \theta_1), \ldots, (p_m, j_m, \theta_m)\}$ is:
$\{(p', i+1, \theta) \mid (p, \gamma, w_i, p', \theta) \in \Delta \wedge i < |w|\} \cup \{(p', i, \theta) \mid (p, \gamma, \epsilon, p', \theta) \in \Delta\}$

(4) $\delta((p, i, \theta), \gamma) = ((p, i), \theta)$

(5) $\delta(q_\top, \gamma) = (\mathtt{T}; \epsilon)$

(6) $\delta(q_\bot, \gamma) = (\mathtt{R}; \epsilon)$

Rules (2) and (3) are applied only when rule (1) is inapplicable. $\mathcal{M}_{\mathcal{A},w}$ simulates $\mathcal{A}$ over the word w, and constructs a tree representing the computation

[2] The *branch language* of $t : dom(t) \longrightarrow \Sigma$ consists of (i) infinite words $(f_1, d_1)(f_2, d_2)\cdots$ just if there exists $d_1\,d_2\cdots \in \{1, 2, \cdots, m\}^\omega$ (where m is the maximum arity of the Σ-symbols) such that $t(d_1 \cdots d_i) = f_{i+1}$ for every $i \geq 0$; and (ii) finite words $(f_1, d_1)\cdots(f_n, d_n)\,f_{n+1}$ just if there exists $d_1 \cdots d_n \in \{1, \cdots, m\}^*$ such that $t(d_1 \cdots d_i) = f_{i+1}$ for $0 \leq i \leq n$, and the arity of f_{n+1} is 0.

of $\mathcal{A}$. A state $(p, i) \in P \times \{0, \ldots, |w| - 1\}$ simulates $\mathcal{A}$ in state p reading the letter w_i. A state (p, i, θ) simulates an intermediate transition state of $\mathcal{A}$, where θ is the stack operation to be applied. The states $q_\top$ and $q_\perp$ are for creating dummy subtrees of nodes labelled with $\mathtt{A}$ or $\mathtt{E}$, so that the number of children of these nodes adds up to N, the arity of $\mathtt{A}$ and $\mathtt{E}$. Rule (1) ensures that when $\mathcal{A}$ has read the input word and reached a final state, $\mathcal{M}_{\mathcal{A},w}$ stops simulating $\mathcal{A}$ and outputs $\mathtt{T}$. Rule (2) is used to simulate transitions of $\mathcal{A}$ in a universal state, reading the i-th input: $\mathcal{M}_{\mathcal{A},w}$ constructs a node labelled $\mathtt{A}$ (to record that $\mathcal{A}$ was in a universal state) and spawns threads to simulate all possible transitions of $\mathcal{A}$. Rule (3) is for simulating $\mathcal{A}$ in an existential state. Note that, if $\mathcal{A}$ gets stuck (i.e. if there is no outgoing transition), all children of the $\mathtt{E}$-node are labelled $\mathtt{R}$; thus failure of the computation can be recognized by the trivial APT given in the following. Rule (4) is just for intermediate transitions. Note that a transition of $\mathcal{A}$ is simulated by $\mathcal{M}_{\mathcal{A},w}$ in two steps: the first for outputting $\mathtt{A}$ or $\mathtt{E}$, and the second for changing the stack.

Now, we construct a trivial APT that accepts the tree generated by $\mathcal{M}_{\mathcal{A},w}$ if, and only if, w is *not* accepted by $\mathcal{A}$. Let $\mathcal{B}_\mathcal{A}$ be $(\{q_0\}, \{\mathtt{A}, \mathtt{E}, \mathtt{T}, \mathtt{R}\}, q_0, \delta, \{q_0 \mapsto 0\})$ where:

$$\delta(q_0, \mathtt{A}) = \bigvee_{i=1}^{N} (i, q_0) \quad \delta(q_0, \mathtt{E}) = \bigwedge_{i=1}^{N} (i, q_0) \quad \delta(q_0, \mathtt{T}) = \mathsf{f} \quad \delta(q_0, \mathtt{R}) = \mathsf{t}$$

Intuitively, $\mathcal{B}_\mathcal{A}$ accepts all trees representing a failure computation tree of $\mathcal{A}$. If the automaton in state q_0 reads $\mathtt{T}$ (which corresponds to an accepting state of $\mathcal{A}$), it gets stuck. Upon reading $\mathtt{A}$, the automaton non-deterministically chooses one of the subtrees, and checks whether the subtree represents a failure computation of $\mathcal{A}$. On the other hand, upon reading $\mathtt{E}$, the automaton checks that all subtrees represent failure computation trees of $\mathcal{A}$.

Based on the above intuition, we can prove the following result.

Theorem 4. *Let w be a word, and $\mathcal{A}$ an order-n APDA. Then w is* not *accepted by $\mathcal{A}$ if, and only if, the tree generated by $\mathcal{M}_{\mathcal{A},w}$ is accepted by $\mathcal{B}_\mathcal{A}$.*

Corollary 1. *The model checking of an order-n recursion scheme with respect to a trivial APT is n-EXPTIME-hard in the size of the recursion scheme.*

By modifying the encoding, we can also show that the model checking problem is n-EXPTIME-hard in the size of APT. The idea is to modify $\mathcal{M}_{\mathcal{A},w}$ so that it generates a tree representing computation of $\mathcal{A}$ over not just w but all possible input words, and let a trivial APT check the part of the tree corresponding to the input word w. As a result, the trivial APT depends on the input word w, but the tree-generating PDA does not. See [7] for more details. To our knowledge, the lower-bound (of the complexity of model-checking recursion schemes) in terms of the size of APT has been unknown even for the entire class of APT.

4 Disjunctive APT and Complexity of Model Checking

A *disjunctive APT* is an APT whose transition function δ is disjunctive, i.e. δ maps each state to a positive boolean formula θ that contains only disjunctions and no conjunctions, as given by the grammar $\theta \ ::= \ \mathtt{t} \mid \mathtt{f} \mid (i,q) \mid \theta \vee \theta$. Disjunctive APT can be used to describe path (or linear-time) properties of trees.

First we give a logical characterization of disjunctive APT as follows. Call $\mathcal{D}$ the following "disjunctive fragment" of the modal mu-calculus:

$$\phi, \psi \ ::= \ P_f \wedge \phi \mid Z \mid \phi \vee \psi \mid \langle i \rangle \phi \mid \nu Z.\phi \mid \mu Z.\phi$$

where f ranges over symbols in Σ, and i ranges over $\{1, \cdots, \mathit{arity}(\Sigma)\}$.

Proposition 2 (Equi-Expressivity). *The logic $\mathcal{D}$ and disjunctive APT are equivalent for defining possibly-infinite ranked trees. I.e. for every closed $\mathcal{D}$-formula, there is a disjunctive APT that defines the same tree language, and vice versa.*

Remark 1. (i) A disjunctive APT is a non-deterministic parity tree automaton. (ii) For defining languages of ranked trees, disjunctive APT are a proper subset of the *disjunctive formulas* in the sense of Walukiewicz and Janin [9]. For example, the disjunctive formula $(1 \rightarrow \{\mathtt{t}\}) \wedge (2 \rightarrow \{\mathtt{t}\})$ is not equivalent to any disjunctive APT.

In the rest of the section, we show that the model checking problem for order-n recursion schemes is $(n-1)$-EXPTIME complete for disjunctive APT.

4.1 Upper Bound

We sketch a proof of the following theorem, based on Kobayashi and Ong's type system for recursion schemes [6]. An alternative proof, based on variable profiles [1], will be given in a forthcoming journal version of this paper [7].

Theorem 5. *Let $\mathcal{G}$ be an order-n recursion scheme and $\mathcal{B}$ a disjunctive APT. It is decidable in $(n-1)$-EXPTIME whether $\mathcal{B}$ accepts the value tree $[\![\mathcal{G}]\!]$ from its root.*

In a recent paper [6], we constructed an intersection type system equivalent to the modal mu-calculus model checking of recursion schemes, in the sense that for every APT, there is a type system such that the tree generated by a recursion scheme is accepted by the APT if, and only if, the recursion scheme is typable in the type system. Thus, the model checking problem is reduced to a type checking problem. The main idea of the type system is to refine the tree type o by the states and priorities of an APT: the type q describes a tree that is accepted by the APT with q as the start state. The intersection type $(\theta_1, m_1) \wedge (\theta_2, m_2) \rightarrow q$, which refines the type $\mathsf{o} \rightarrow \mathsf{o}$, describes a tree function that takes an argument which has types θ_1 and θ_2, and returns a tree of type q.

The type checking algorithm presented in *ibid.* is n-EXPTIME in the combined size of the order-n recursion scheme and APT (precisely the complexity is $O(r^{1+\lfloor m/2 \rfloor} \mathbf{exp}_n((a\,|Q|\,m)^{1+\epsilon}))$ for $n \geq 2$, where r is the number of rules and a the largest prioirty of symbols in the scheme, m is the largest priority, $|Q|$ is the number of states) The bottleneck of the algorithm is the number of (atomic) intersection types, where the set $\mathcal{T}(\kappa)$ of atomic types refining a simple type κ is inductively defined by: $\mathcal{T}(\mathsf{o}) = Q$ and $\mathcal{T}(\kappa_1 \to \kappa_2) = \{\bigwedge S \to \theta \mid \theta \in \mathcal{T}(\kappa_2), S \subseteq \mathcal{T}(\kappa_1) \times P\}$. where Q and P are the sets of states and priorities respectively.

According to the syntax of atomic types above, the number of atomic types refining a simple type of order-n is n-exponential in general. In the case of disjunctive APT, however, for each type of the form $\mathsf{o} \to \cdots \to \mathsf{o} \to \mathsf{o}$, we need to consider only atomic types of the form $\bigwedge S_1 \to \cdots \to \bigwedge S_k \to q$, where at most one of the S_i's is a singleton set and the other S_j's are empty. Intuitively, this is because a run-tree of a disjunctive APT consists of a single path, so that the run-tree visits only one of the arguments, at most once. In fact, we can show that, if a recursion scheme is typable in the type system for a disjunctive APT, the recursion scheme is typable in a restricted type system in which order-1 types are constrained as described above: this follows from the proof of completeness of the type system [6], along with the property of the accepting run-tree mentioned above. Thus, the number of atomic types is $k \times |Q| \times |P| \times |Q|$ (whereas for general APT, it is exponential). Therefore, the number of atomic types possibly assigned to a symbol of order n is $(n-1)$-exponential. By running the same type checking algorithm as *ibid.* (but with order-1 types constrained as above), order-n recursion schemes can be type-checked (i.e. model-checked) in $(n-1)$-EXPTIME.

4.2 Lower Bound

We show the lower bound by a reduction of the emptiness problem of the finite-word language accepted by an order-n (deterministic) PDA (which is $(n-1)$-EXPTIME complete [5]). From an order-n PDA $\mathcal{A}$, we can construct an order-n tree-generating PDA $\mathcal{M}_{\mathcal{A}}$, which simulates all possible input and ϵ-transitions of $\mathcal{A}$, and outputs e only when $\mathcal{A}$ reaches a final state: See the long version [7]. By a result of Knapik et al. [8], we can construct an equi-expressive order-n safe recursion scheme $\mathcal{G}$. By the construction, the finite word-language accepted by $\mathcal{A}$ is non-empty if, and only if, the value tree of $\mathcal{G}$ has a node labelled e. Since the latter property can be expressed by a disjunctive APT, the problem of model-checking recursion schemes for disjunctive APT is $(n-1)$-EXPTIME hard. The problem is $(n-1)$-EXPTIME hard also in the size of the disjunctive APT: See [7] for more details.

4.3 Path Properties

The *path language* of a Σ-labelled tree t is the image of the map F, which acts on the elements of the branch language of t by "forgetting the argument positions" i.e. $F : (f_1, d_1)\,(f_2, d_2) \cdots \mapsto f_1\, f_2 \cdots$ and $F : (f_1, d_1) \cdots (f_n, d_n)\, f_{n+1} \mapsto f_1 \cdots f_n\, f_{n+1}^{\omega}$. For example, the path language of the tree $f\,a\,(f\,a\,b)$ is $\{f\,a^{\omega}, f\,f\,a^{\omega}, f\,f\,b^{\omega}\}$. Let $\mathcal{G}$ be a recursion scheme. We write $\mathcal{W}(\mathcal{G})$ for the *path language* of $[\![\mathcal{G}]\!]$. Thus elements of $\mathcal{W}(\mathcal{G})$ are infinite words over the alphabet Σ which is now considered unranked (i.e. arities of the symbols are forgotten).

Lemma 1. *Let $\mathcal{G}$ be an order-n recursion scheme. The following problems are $(n-1)$-EXPTIME complete.*

(i) $\mathcal{W}(\mathcal{G}) \cap \mathcal{L}(\mathcal{C}) \overset{?}{=} \emptyset$*, where* $\mathcal{C}$ *is a* non-deterministic *parity word automaton.*

(ii) $\mathcal{W}(\mathcal{G}) \overset{?}{\subseteq} \mathcal{L}(\mathcal{C})$*, where* $\mathcal{C}$ *is a* deterministic *parity word automaton.*

The decision problems Reachability (i.e. whether $[\![\mathcal{G}]\!]$ has a node labelled by a given symbol `e`) and Finiteness (i.e. whether $[\![\mathcal{G}]\!]$ is finite) are instances of Problem (i) of Lemma 1; hence they are in $(n-1)$-EXPTIME (the former is $(n-1)$-EXPTIME complete, by the proof of Section 4.2).

5 Application to Resource Usage Verification

Now we apply the result of the previous section to show that the resource usage verification problem is $(n-1)$-EXPTIME complete. The aim of resource usage verification is to check whether a program accesses each resource according to the resource specification. For example, consider the following program.

```
let rec g x = if b then close(x) else read(x); g(x) in
let r = open_in "foo" in g(r)
```

It opens a read-only file “`foo`”, reads and closes it. For this program, the goal of the verification is to statically check that the file is eventually closed before the program terminates, and after it is closed, it is never read from or written to.

The resource usage verification problem was formalized by Igarashi and Kobayashi [3]. Kobayashi [2] recently showed that the problem is decidable for the simply-typed λ-calculus with recursion, generated from a base type of booleans, and augmented by resource creation/access primitives, by reduction to the model checking problem for recursion schemes.

Kobayashi [2] considered a language in CPS (continuation passing style), with only top-level function definitions of the form $F\,\widetilde{x} = e$, where e is given by:

$$e ::= \star \mid x \mid F \mid e_1 e_2 \mid \mathbf{If}{*}\ e_1\ e_2 \mid \mathbf{New}^L\ e \mid \mathbf{Acc}_a\ e_1\ e_2$$

The term $\star$ is the unit value. The term $\mathbf{If}{*}\ e_1\ e_2$ is a non-deterministic branch between e_1 and e_2. The term $\mathbf{New}^L\ e$ creates a fresh resource that should be used according to L, and passes it to e (thus, e is a function that takes a resource as an argument). Here, L is a regular language. The term $\mathbf{Acc}_a\ e_1\ e_2$ accesses the resource e_1 (where a is the name of the access primitive), and then executes e_2. For example, in the above program, the last line is expressed by $\mathbf{New}^{r*c}\ G$, and `close(x)` is expressed by $\mathbf{Acc}_c\ x\ k$ (where k is the continuation).

The language is simply typed; the two base types are **unit** for unit values and **R** for resources. The body of each definition must have type **unit** (in other words, resources cannot be used as return values; in that sense, programs are in CPS). The constants $\mathbf{If}{*}$, $\mathbf{New}^L$, and $\mathbf{Acc}_a$ are given the following types.

$$\mathbf{If}{*}{:}\mathbf{unit} \to \mathbf{unit} \to \mathbf{unit}, \mathbf{New}^L{:}(\mathbf{R} \to \mathbf{unit}) \to \mathbf{unit}, \mathbf{Acc}_a{:}\mathbf{R} \to \mathbf{unit} \to \mathbf{unit}$$

A program can be transformed to a recursion scheme that generates a tree representing all possible (resource-wise) access sequences of the program [2]. We just need to replace each function definition $F\,\widetilde{x} = e$ with the rewrite rule $F\,\widetilde{x} \rightarrow e$, and add the following rules

$$\begin{array}{lll}
\mathbf{If}{*}\ x\ y \rightarrow \mathtt{br}\,x\,y & \mathbf{Acc}_a\ x\ k \rightarrow x\,a\,k & \star \rightarrow \mathtt{t}(\star) \\
\mathbf{New}^L k \rightarrow \mathtt{br}\,(\nu^L(kI))\,(kK) & I\,x\,k \rightarrow x\,k & K\,x\,k \rightarrow k
\end{array}$$

Here, $\mathtt{br}$ is a terminal for representing non-deterministic choice. In the rule for $\mathbf{New}^L$, a fresh resource is instantiated to either I or K. This is a trick used to extract resource-wise access sequences, by tracing or ignoring the new resource in a non-deterministic manner: see Kobayashi's paper [2] for more explanation. The above transformation preserves types, except that **unit** and **R** are replaced by $\mathtt{o}$ and $(\mathtt{o} \rightarrow \mathtt{o}) \rightarrow \mathtt{o} \rightarrow \mathtt{o}$ respectively.

Along with the transformation above, a tree automaton can be constructed that accepts the trees containing an *invalid* access sequence. The automaton just needs to focus on paths that contain a single occurrence of $\mathbf{New}^L$, and check whether, for every sequence s below $\mathbf{New}^L$, all prefixes of s are elements of $L \cdot \mathtt{t}^*$ (with $\mathtt{br}$ ignored). Thus, the automaton belongs to the class of disjunctive APT. (On the other hand, an automaton that accepts the complement, i.e. the set of trees containing only valid sequences, belongs to the class of trivial APT.)

We now show that the resource usage verification is $(n-1)$-EXPTIME complete, where n is the largest order of types in the source program. The base types **unit** and **R** have orders 0 and 1 respectively. We assume that each resource specification in the program is given as a deterministic finite state automaton. The lower-bound can be shown by reduction of the reachability problem of recursion schemes to the resource usage verification problem: we just transform each rule $F\,\widetilde{x} \rightarrow t$ into the function definition $F\,\widetilde{x} = t$, and replace the terminal $\mathtt{e}$ with $\mathbf{New}^{\{\epsilon\}} \mathit{Fail}$, where Fail is defined by $\mathit{Fail}\,x = \mathbf{Acc}_{\mathtt{fail}}\ x\ \star$. Since resource primitives occur only in the transformation of $\mathtt{e}$, the order of the resulting resource usage program is the maximum of 3 and the order of the recursion scheme. Thus, the resource usage verification is $(n-1)$-EXPTIME hard for $n \geq 3$ (note that 3 is the lowest order of a closed program that creates a resource, since $\mathbf{New}^L$ has order 3).

Showing the upper-bound is a little tricky: since the resource type **R** of order 1 is transformed into the type $(\mathtt{o} \rightarrow \mathtt{o}) \rightarrow \mathtt{o} \rightarrow \mathtt{o}$ of order 2, a source program of order-n may be transformed into a recursion scheme of order $n+1$. For the image of the resource type, however, it is sufficient to consider only two atomic types σ_I and σ_K, where $\sigma_I = \bigwedge_{q_1,q_2}((q_1 \rightarrow q_2) \rightarrow q_1 \rightarrow q_2)$ and $\sigma_K = \bigwedge_q((\bigwedge \emptyset) \rightarrow q \rightarrow q)$. Here, we have omitted priorities. Thus, although, for example, a type $\mathbf{R} \rightarrow \cdots \rightarrow \mathbf{R} \rightarrow \mathbf{unit}$ of order 2 is transformed into an order-3 type, the number of atomic types that should be considered is single-exponential. Since the APT for recognizing the value tree is disjunctive, we can apply the argument in Section 4 to conclude that the recursion scheme can be model-checked in $(n-1)$-EXPTIME.

Acknowledgments. We would like to thank the anonymous reviewers for useful comments. This work was partially supported by Kakenhi 20240001 and EPSRC EP/F036361.

References

1. Ong, C.-H.L.: On model-checking trees generated by higher-order recursion schemes. In: LICS 2006, pp. 81–90. IEEE Computer Society Press, Los Alamitos (2006)
2. Kobayashi, N.: Types and higher-order recursion schemes for verification of higher-order programs. In: Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages, pp. 416–428 (2009)
3. Igarashi, A., Kobayashi, N.: Resource usage analysis. ACM Transactions on Programming Languages and Systems 27(2), 264–313 (2005)
4. Aehlig, K.: A finite semantics of simply-typed lambda terms for infinite runs of automata. Logical Methods in Computer Science 3(3) (2007)
5. Engelfriet, J.: Iterated stack automata and complexity classes. Information and Computation 95(1), 21–75 (1991)
6. Kobayashi, N., Ong, C.-H.L.: A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In: Proceedings of LICS 2009 (2009)
7. Kobayashi, N., Ong, C.-H.L.: Complexity of model checking recursion schemes for fragments of the modal mu-calculus. A long version, available from the authors' web page (2009)
8. Knapik, T., Niwiński, D., Urzyczyn, P.: Higher-order pushdown trees are easy. In: Nielsen, M., Engberg, U. (eds.) FOSSACS 2002. LNCS, vol. 2303, pp. 205–222. Springer, Heidelberg (2002)
9. Janin, D., Walukiewicz, I.: Automata for the modal mu-calculus and related results. In: Hájek, P., Wiedermann, J. (eds.) MFCS 1995. LNCS, vol. 969, pp. 552–562. Springer, Heidelberg (1995)

LTL Path Checking Is Efficiently Parallelizable[*]

Lars Kuhtz and Bernd Finkbeiner

Universität des Saarlandes
66123 Saarbrücken, Germany
{kuhtz,finkbeiner}@cs.uni-sb.de

Abstract. We present an $\mathsf{AC}^1(\mathsf{logDCFL})$ algorithm for checking LTL formulas over finite paths, thus establishing that the problem can be efficiently parallelized. Our construction provides a foundation for the parallelization of various applications in monitoring, testing, and verification.

Linear-time temporal logic (LTL) is the standard specification language to describe properties of reactive computation paths. The problem of checking whether a given finite path satisfies an LTL formula plays a key role in monitoring and runtime verification [12,10,6,1,4], where individual paths are checked either online, during the execution of the system, or offline, for example based on an error report. Similarly, path checking occurs in testing [2] and in several static verification techniques, notably in Monte-Carlo-based probabilistic verification, where large numbers of randomly generated sample paths are analyzed [22].

Somewhat surprisingly, given the widespread use of LTL, the complexity of the path checking problem is still open [18]. The established upper bound is P: The algorithms in the literature traverse the path sequentially (cf. [10,18,12]); by going backwards from the end of the path, one can ensure that, in each step, the value of each subformula is updated in constant time, which results in bilinear running time. The only known lower bound is NC^1 [8], the complexity of evaluating Boolean expressions. The large gap between the bounds is especially unsatisfying in light of the recent trend to implement path checking algorithms in hardware, which is inherently parallel. For example, the IEEE standard temporal logic PSL [13], an extension of LTL, has become part of the hardware description language VHDL, and several tools [6,4] are available to synthesize hardware-based monitors from assertions written in PSL. Can we improve over the sequential approach by evaluating entire blocks of path positions in parallel?

In this paper, we show that LTL path checking can indeed be parallelized efficiently. Our approach is inspired by work in the related area of evaluating monotone Boolean circuits [11,9,15,3,17,5]. Rather than sequentially traversing the path, we consider the circuit that results from unrolling the formula over the

[*] This work was partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 235–246, 2009.

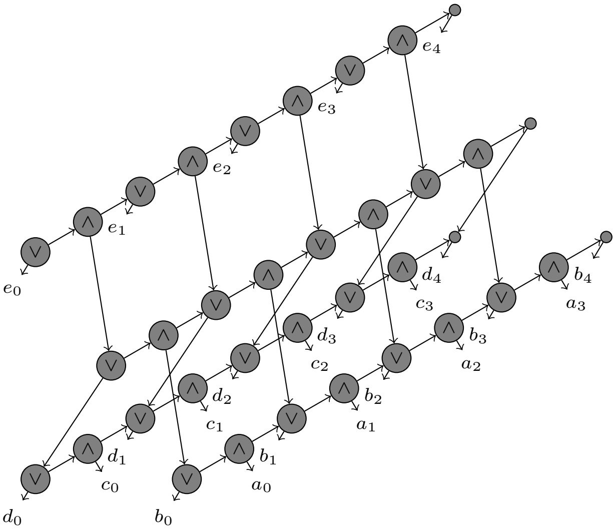


Fig. 1. Circuit resulting from unrolling the LTL formula $((a \,\mathrm{U}\, b) \,\mathrm{U}\, (c \,\mathrm{U}\, d)) \,\mathrm{U}\, e$ over a path ρ of length 5. We denote the value of an atomic proposition p at a path position $i = 0, \ldots, 4$ by p_i. The graph of the circuit has no planar embedding.

path. Figure 1 shows such a circuit for the formula $((a \,\mathrm{U}\, b) \,\mathrm{U}\, (c \,\mathrm{U}\, d)) \,\mathrm{U}\, e$ and a path of length 5.

Yang [21] and, independently, Delcher and Kosaraju [7] have shown that monotone Boolean circuits can be evaluated efficiently in parallel *if the graph of the circuit has a planar embedding.* Unfortunately, this condition is already violated in the simple example of Figure 1. Individually, however, each operator results in a planar circuit: for example, d U e results in $e_0 \vee (d_0 \wedge (e_1 \vee (d_1 \wedge \ldots) \cdots)$. The complete formula thus defines a tree of planar circuits.

Our path checking algorithm works on this tree of circuits. We introduce a contraction technique that combines a parent node and its children into a single planar circuit. Simple paths in the tree immediately collapse into a planar circuit; the remaining binary tree is contracted incrementally, until only a single planar circuit remains. The key insight of our solution is that the contraction can be carried out *as soon as one of the children has been evaluated.* Because no evaluated child has to wait for the evaluation of its sibling before it can be contracted with its parent, we can contract a fixed portion of the nodes in every sequential step, and therefore terminate in at most a logarithmic number of steps.

The path checking problem can, hence, be parallelized efficiently. In addition to planarity, our construction maintains some further technical invariants, in particular that the circuits have all input gates on the outer face. Analyzing this construction, we obtain the result that the path checking problem is in $\mathsf{AC}^1(\mathsf{logDCFL})$.

1 Preliminaries

Linear-Time Temporal Logic. We consider specifications in linear-time temporal logic (LTL). We apply the usual finite-path semantics with a weak and a strong version of the *Next*-Operator [16]. Let P be a set of atomic propositions. The *formulas* of LTL are defined inductively as follows: For each atomic proposition $p \in P$ p and $\neg p$ are LTL formulas. If ϕ and ψ are LTL formulas, then so are

$$\phi \wedge \psi, \quad \phi \vee \psi, \quad \mathrm{X}_{\exists}\, \phi, \quad \mathrm{X}_{\forall}\, \phi, \quad \phi \,\mathrm{U}\, \psi, \quad \text{and} \quad \phi \,\mathrm{R}\, \psi \ .$$

LTL formulas are evaluated over computation paths. A *path* $\rho = \rho_0, \ldots, \rho_{n-1}$ is a sequence of states where each *state* ρ_i for $i = 0, \ldots, n-1$ is a valuation $\rho_i \in 2^P$ of the atomic propositions. The *length* of ρ is n and is denoted by $\|\rho\|$. The *suffix* of ρ at position i, $0 \leq i < n$, is denoted by ρ^i. The *empty path* is denoted by ϵ.

Given an LTL formula ϕ, a nonempty path $\rho \neq \epsilon$ satisfies ϕ, denoted by $\rho \models \phi$, if one of the following holds:

- $\phi \in P$ and $\phi \in \rho_0$,
- $\phi = \neg p$ and $p \notin \rho_0$,
- $\phi = \phi_l \wedge \phi_r$ and $\rho \models \phi_l$ and $\rho \models \phi_r$,
- $\phi = \phi_l \vee \phi_r$ and $\rho \models \phi_l$ or $\rho \models \phi_r$,
- $\phi = \mathrm{X}_{\exists}\psi$ and $\rho^1 \models \psi$ and $\rho^1 \neq \epsilon$,
- $\phi = \mathrm{X}_{\forall}\psi$ and $\rho^1 \models \psi$ or $\rho^1 = \epsilon$,
- $\phi = \phi_l \,\mathrm{U}\, \phi_r$ and $\exists 0 \leq i < \|\rho\|$ s.t. $\rho^i \models \phi_r$ and $\forall 0 \leq j < i$, $\rho^j \models \phi_l$, or
- $\phi = \phi_l \,\mathrm{R}\, \phi_r$ and $\forall 0 \leq i < \|\rho\|, \rho^i \models \phi_r$ or $\exists 0 \leq j < i$ s.t. $\rho^j \models \phi_l$.

The semantics of LTL implies the *expansion laws*, which relate the satisfaction of a temporal formula in some position of the path to the satisfaction of the formula in the next position and the satisfaction of its subformulas in the present position:

$$\phi_l \,\mathrm{U}\, \phi_r \;\equiv\; \phi_r \vee (\phi_l \wedge \mathrm{X}_{\exists}\,(\phi_l \,\mathrm{U}\, \phi_r)); \quad \phi_l \,\mathrm{R}\, \phi_r \;\equiv\; \phi_r \wedge (\phi_l \vee \mathrm{X}_{\forall}\,(\phi_l \,\mathrm{R}\, \phi_r)) \ .$$

We are interested in determining if an LTL formula is satisfied by a given path. This is the path checking problem.

Definition 1 (Path Checking Problem). *The path checking problem for LTL is to decide, for an LTL formula ϕ and a nonempty path ρ, whether $\rho \models \phi$.*

Complexity classes within P. We assume familiarity with the standard complexity classes within P. L is the class of problems that can be decided by a logspace restricted deterministic Turing machine. $\mathsf{logDCFL}$ is the class of problems that can be decided by a logspace and polynomial time restricted deterministic Turing machine that is equipped with a stack. $\mathsf{AC}^i, i \in \mathbb{N}$, denotes the class of problems decidable by polynomial size unbounded fan-in Boolean circuits of depth $\log^i$, where the depth of a circuit is the length of a longest directed path in the circuit. AC is defined as $\bigcup_{i \in \mathbb{N}} \mathsf{AC}^i$. Throughout the paper, all circuits are assumed to be uniform in the sense that the circuit for inputs of length n can be generated by a deterministic Turing machine using space $\log(n)$. It holds that

$$\mathsf{L} \subseteq \mathsf{logDCFL} \subseteq \mathsf{AC}^1 \subseteq \mathsf{AC}^2 \subseteq \cdots \subseteq \mathsf{AC} \subseteq \mathsf{P} \ .$$

Given a problem P and a complexity class C, P is AC^1 Turing reducible to C (denoted as $P \in \mathsf{AC}^1(C)$) if there is a family of AC^1 circuits with additional unbounded fan-in C-oracle gates that decides P. It holds that

$$\mathsf{AC}^1 \subseteq \mathsf{AC}^1(\mathsf{logDCFL}) \subseteq \mathsf{AC}^2 \ .$$

Monotone Boolean circuits. A *monotone Boolean circuit* $\langle \Gamma, \gamma \rangle$ consists of a set Γ of *gates* and a gate labeling γ. The *gate labeling* labels each gate either with a Boolean value or with a tuple $\langle \mathit{and}, \mathit{left}, \mathit{right} \rangle$, $\langle \mathit{or}, \mathit{left}, \mathit{right} \rangle$, $\langle \mathit{id}, \mathit{suc} \rangle$, where *left*, *right*, and *suc* are gates.

A gate that is labeled with a Boolean value is called a *constant gate*. For a non-constant gate a labeled with $\langle \mathit{id}, b \rangle$, we say that *$a$ directly depends* on b, denoted by $a \succ b$. Likewise, for a gate a labeled with $\langle \mathit{and}, b, c \rangle$ or $\langle \mathit{or}, b, c \rangle$, a directly depends on b and c. The *dependence* relation is the transitive closure of $\succ$. A gate on which no other gate depends is called a *sink gate. A circuit must not contain any cyclic dependencies.*

For a set of gates G, $\mathrm{const}(G)$ denotes the set of all constant gates in G. If $G = \mathrm{const}(G)$, we call G *constant.*

In the following, we assume that all circuits are monotone Boolean circuits. We omit the labeling whenever it is clear from the context and identify the circuit with its set of gates. We will often analyze subcircuits which are only well-defined in the context of the full circuit. We call such subcircuits partial circuits: Given a circuit $C = \langle \Gamma, \gamma \rangle$, a *partial circuit* is a circuit $D = \langle \Delta, \delta \rangle$ with $\Delta \subseteq \Gamma$ and $\delta = \gamma|_\Delta$. The gates in $\{g \in \Gamma \setminus \Delta \mid \exists h \in \Delta .\, h \succ g\}$ are called the *variable gates* of D. For a variable gate g of D, we define $\delta(g) = \perp$. If C is clear from the context, we refer to D as Δ.

Circuit evaluation. The *evaluation* of a circuit $\langle \Gamma, \gamma \rangle$ is the (unique) circuit $\langle \Gamma, \gamma' \rangle$ where for each gate $g \in \Gamma$ the following holds:

- $\gamma'(g) = 0$ iff $\gamma(g) = \langle \mathit{and}, l, r \rangle$ and $\gamma'(l) = 0$ or $\gamma'(r) = 0$,
- $\gamma'(g) = 1$ iff $\gamma(g) = \langle \mathit{and}, l, r \rangle$ and $\gamma'(l) = 1$ and $\gamma'(r) = 1$,
- $\gamma'(g) = \langle \mathit{id}, l \rangle$ iff $\gamma(g) = \langle \mathit{and}, l, r \rangle$ and $\gamma'(l) \notin \{0, 1\}$ and $\gamma'(r) = 1$,
- $\gamma'(g) = \langle \mathit{id}, r \rangle$ iff $\gamma(g) = \langle \mathit{and}, l, r \rangle$ and $\gamma'(r) \notin \{0, 1\}$ and $\gamma'(l) = 1$,
- $\gamma'(g) = 0$ iff $\gamma(g) = \langle \mathit{or}, l, r \rangle$ and $\gamma'(l) = 0$ and $\gamma'(r) = 0$,
- $\gamma'(g) = 1$ iff $\gamma(g) = \langle \mathit{or}, l, r \rangle$ and $\gamma'(l) = 1$ or $\gamma'(r) = 1$,
- $\gamma'(g) = \langle \mathit{id}, l \rangle$ iff $\gamma(g) = \langle \mathit{or}, l, r \rangle$ and $\gamma'(l) \notin \{0, 1\}$ and $\gamma'(r) = 0$,
- $\gamma'(g) = \langle \mathit{id}, r \rangle$ iff $\gamma(g) = \langle \mathit{or}, l, r \rangle$ and $\gamma'(r) \notin \{0, 1\}$ and $\gamma'(l) = 0$,
- $\gamma'(g) = \gamma'(s)$ iff $\gamma(g) = \langle \mathit{id}, s \rangle$ and $\gamma'(s) \in \{0, 1\}$, and
- $\gamma'(g) = \gamma(g)$ otherwise.

A circuit is *evaluated* if all constant gates are sink gates. In an evaluated circuit, all gates that do not depend on variable gates are constant. Hence, a full circuit evaluates to a constant circuit; for a partial circuit, a subset of the gates is relabeled: some *and-*/*or-*/*id*-gates are labeled as constant or *id*-gates. In the construction presented in this paper, we evaluate circuits in several stages by evaluating partial circuits. In this process, the evaluation of a partial circuit includes substituting the partial circuit by its evaluation within the full circuit. Since the evaluation of a partial circuit is a local operation, disjoint partial circuits can be evaluated in parallel.

The problem of evaluating monotone planar circuits has been studied extensively in the literature. Our construction is based on the evaluation of one-input-face planar circuits:

Given a circuit $G = \langle \Gamma, \gamma \rangle$ with variable gates X, the *graph* $\mathrm{gr}(G)$ *of* G is the directed graph $\langle V, E \rangle$, where $V = \Gamma \cup X$ and $E = \{\langle a, b \rangle \in V \times V \mid a \succ b\}$. A circuit C is *planar* if there exists an planar embedding of the graph of C. The *input gates* of C are all constant and all variable gates of C. A planar partial circuit is *one-input-face* if there is a planar embedding such that all input gates are located on the outer face.

In the following, we abbreviate *evaluated circuit* as EV and *one-input-face planar* as OIF, using the terms EV and OIF for the circuits as well as for the corresponding property of a circuit. Note that an EV circuit with all variables on the outer face is OIF. The evaluation of full OIF circuits can be parallelized efficiently. We make use of a result by Chakraborty and Datta [5]:

Theorem 1 (Chakraborty and Datta 2006). *The problem of evaluating a full* OIF *circuit is in* logDCFL*.*

Using standard techniques for partial circuits [15], the theorem generalizes from full to partial circuits:

Corollary 1. *The problem of evaluating an* OIF *circuit is in* logDCFL*.*

Proof. We first assign the Boolean constant 1 to all variable gates. Each gate that evaluates to 0 is turned into a 0 constant gate. Next, we assign 0 to all variable gates. Each gate that evaluates to 1 is turned into a constant gate with value 1. Since the values of the remaining gates depend on the variables, they are simply copied. If one of the latter gates depends on a constant gate, the dependency is removed by changing such a gate into an *id*-gate. □

2 From LTL to Circuits

In this section, we provide an L many-one reduction from the path checking problem of LTL to the problem of evaluating monotone Boolean circuits.

Given an LTL formula ϕ and a path ρ, we define a circuit $C(\phi, \rho) = \langle \Gamma, \gamma \rangle$ such that $\rho \models \phi$ if and only if a distinguished result gate $c_{0,0}$ is mapped to 1 in the evaluation of $C(\phi, \rho)$. The circuit is constructed by unrolling ϕ on ρ into a DAG according to the expansion laws.

Definition 2. *Given an LTL formula ϕ and a path ρ, the circuit $C(\phi, \rho) = \langle \Gamma, \gamma \rangle$ is defined as follows. Let $\phi_0, \ldots, \phi_{m-1}$ (with $\phi_0 = \phi$) be the subformulas of ϕ and let $\rho = \rho_0, \ldots, \rho_{n-1}$. The set of gates $\Gamma = \bigcup_{\substack{i=0,\ldots,m-1\\ j=0,\ldots,n-1}} C_{i,j}$ contains for each subformula ϕ_i and each path position $0 \leq j < n$ the set $C_{i,j}$ of gates defined below:*

- *$C_{i,j} = \{c_{i,j}\}$ if $j = n - 1$ or if ϕ_i is either an atomic proposition, a negated atomic proposition, a conjunction, a disjunction, an $\mathrm{X}_\exists$-formula, or an $\mathrm{X}_\forall$-formula, and*
- *$C_{i,j} = \{c_{i,j}, c'_{i,j}\}$ if $0 \leq j < n - 1$ and ϕ_i is either an U-formula or an R-formula,*

where $c_{i,j}, c'_{i,j}, i = 0, \ldots, m-1, j = 0, \ldots, n-1$ are distinct gates. The gates are labeled as follows. For $0 \leq j < n - 1$:

- *$\gamma(c_{i,j}) = \langle or, c_{r,j}, c'_{i,j} \rangle$ and*
  $\gamma(c'_{i,j}) = \langle and, c_{l,j}, c_{i,j+1} \rangle$ for $\phi_i = \phi_l \mathrm{U} \phi_r$,
- *$\gamma(c_{i,j}) = \langle and, c_{r,j}, c'_{i,j} \rangle$ and*
  $\gamma(c'_{i,j}) = \langle or, c_{l,j}, c_{i,j+1} \rangle$ for $\phi_i = \phi_l \mathrm{R} \phi_r$, and
- *$\gamma(c_{i,j}) = \langle id, c_{l,j+1} \rangle$ for $\phi_i = \mathrm{X}_\exists \phi_l$ or $\phi_i = \mathrm{X}_\forall \phi_l$;*

for $j = n - 1$:

- *$\gamma(c_{i,j}) = \langle id, c_{r,j} \rangle$ for $\phi_i = \phi_l \mathrm{U} \phi_r$ or $\phi_i = \phi_l \mathrm{R} \phi_r$,*
- *$\gamma(c_{i,j}) = \mathtt{0}$ for $\phi_i = \mathrm{X}_\exists \phi_l$, and*
- *$\gamma(c_{i,j}) = \mathtt{1}$ for $\phi_i = \mathrm{X}_\forall \phi_l$;*

for $0 \leq j < n$:

- *$\gamma(c_{i,j}) = 1$ for either $\phi_i = p$ and $p \in \rho_j$ or $\phi_i = \neg p$ and $p \notin \rho_j$, $p \in P$,*
- *$\gamma(c_{i,j}) = 0$ for either $\phi_i = p$ and $p \notin \rho_j$ or $\phi_i = \neg p$ and $p \in \rho_j$, $p \in P$,*
- *$\gamma(c_{i,j}) = \langle and, c_{l,j}, c_{r,j} \rangle$ for $\phi_i = \phi_l \wedge \phi_r$, and*
- *$\gamma(c_{i,j}) = \langle or, c_{l,j}, c_{r,j} \rangle$ for $\phi_i = \phi_l \vee \phi_r$.*

Lemma 1. *The size of $C(\phi, \rho)$ is polynomial in $\|\rho\|$ and $\|\phi\|$. Moreover, in the evaluation of $C(\phi, \rho)$ the gate $c_{0,0}$ is labeled with the constant $\mathtt{1}$ if and only if $\rho \models \phi$.* □

In the remainder of the paper, we fix the formula ϕ and the path ρ, and refer to the circuit $C(\phi, \rho)$ as C. We now provide an embedding of C.

The *embedding* $Emb_C \colon \mathrm{gr}(C) \to 2^{\mathbb{R}\times\mathbb{R}}$ is defined by $Emb_C(c_{i,j}) = \{\langle j, \mathrm{depth}(\phi_i)\rangle\}$ and $Emb_C(c'_{i,j}) = \{\langle j + 0.5, \mathrm{depth}(\phi_i)\rangle\}$, where $\mathrm{depth}(\phi_i)$ denotes the nesting depth of ϕ_i in ϕ. An edge of $\mathrm{gr}(C)$ is embedded to the line segment between the points onto which the incident nodes are embedded.

In general, Emb_C is not planar. However, for each subformula ϕ_i, $i = 1, \ldots m-1$, we can identify a planar subcircuit $\mu_i = \bigcup_{j=0,\ldots,n-1} C_{i,j}$, which we call the *module* of ϕ_i. Corresponding to the formula structure, the modules form a *module tree* $\mathcal{M} = \langle M, E \rangle$, where $M = \{\mu_i \mid i = 0, \ldots, m-1\}$ and $E = \{\langle \mu_i, \mu_j \rangle \mid$

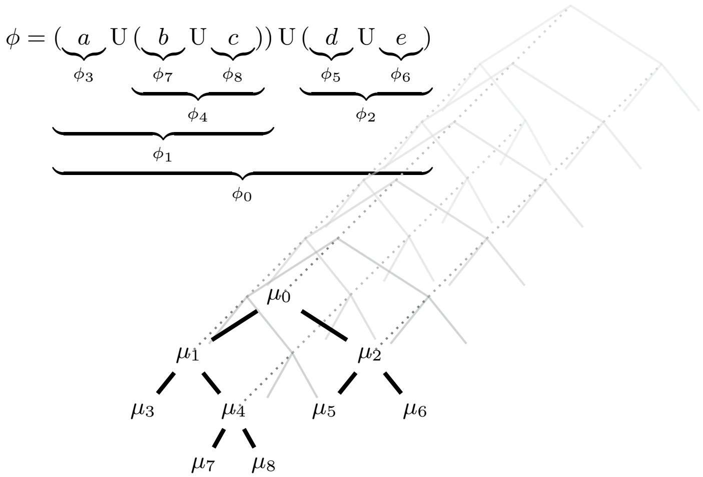

$\phi = (\underbrace{a}_{\phi_3} \,\mathrm{U}\, (\underbrace{b}_{\phi_7} \,\mathrm{U}\, \underbrace{c}_{\phi_8})) \,\mathrm{U}\, (\underbrace{d}_{\phi_5} \,\mathrm{U}\, \underbrace{e}_{\phi_6})$
ϕ_4
ϕ_2
ϕ_1
ϕ_0
μ_0
μ_1
μ_2
μ_3
μ_4
μ_5
μ_6
μ_7
μ_8

Fig. 2. A schematic illustration of the circuit and the module tree for a formula ϕ and a path of length six

$((\phi_i = \phi_k \wedge \phi_l$ or $\phi_i = \phi_k \vee \phi_l$ or $\phi_i = \phi_k \,\mathrm{U}\, \phi_l$ or $\phi_i = \phi_k \,\mathrm{R}\, \phi_l)$ and $(j = l$ or $j = k))$ or $\phi_i = \mathrm{X}_\exists \phi_j$ or $\phi_i = \mathrm{X}_\forall \phi_j\}$. Note that the modules are pairwise disjoint. A schematic illustration of an example circuit and the module tree is shown in Figure 2.

Figure 3 shows the partial circuit that corresponds to a single branch of the module tree from the example of Figure 2.

Our evaluation algorithm, which will be presented in the following section, uses the fast evaluation of OIF circuits from Corollary 1 to evaluate subcircuits of C. The following lemma establishes the connection between the embedding Emb_C and the module tree $\mathcal{M}$ that will allow for the application of Corollary 1 to increasingly larger subtrees of $\mathcal{M}$.

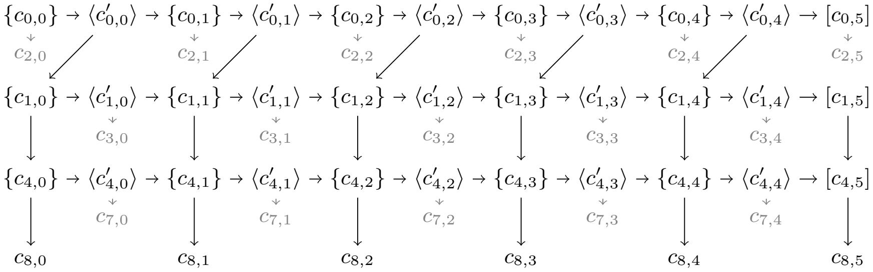

$\{c_{0,0}\}$ $\langle c'_{0,0}\rangle$ $\{c_{0,1}\}$ $\langle c'_{0,1}\rangle$ $\{c_{0,2}\}$ $\langle c'_{0,2}\rangle$ $\{c_{0,3}\}$ $\langle c'_{0,3}\rangle$ $\{c_{0,4}\}$ $\langle c'_{0,4}\rangle$ $[c_{0,5}]$
$c_{2,0}$ $c_{2,1}$ $c_{2,2}$ $c_{2,3}$ $c_{2,4}$ $c_{2,5}$
$\{c_{1,0}\}$ $\langle c'_{1,0}\rangle$ $\{c_{1,1}\}$ $\langle c'_{1,1}\rangle$ $\{c_{1,2}\}$ $\langle c'_{1,2}\rangle$ $\{c_{1,3}\}$ $\langle c'_{1,3}\rangle$ $\{c_{1,4}\}$ $\langle c'_{1,4}\rangle$ $[c_{1,5}]$
$c_{3,0}$ $c_{3,1}$ $c_{3,2}$ $c_{3,3}$ $c_{3,4}$
$\{c_{4,0}\}$ $\langle c'_{4,0}\rangle$ $\{c_{4,1}\}$ $\langle c'_{4,1}\rangle$ $\{c_{4,2}\}$ $\langle c'_{4,2}\rangle$ $\{c_{4,3}\}$ $\langle c'_{4,3}\rangle$ $\{c_{4,4}\}$ $\langle c'_{4,4}\rangle$ $[c_{4,5}]$
$c_{7,0}$ $c_{7,1}$ $c_{7,2}$ $c_{7,3}$ $c_{7,4}$
$c_{8,0}$ $c_{8,1}$ $c_{8,2}$ $c_{8,3}$ $c_{8,4}$ $c_{8,5}$

Fig. 3. The partial circuit for the modules $\mu_0, \mu_1, \mu_4, \mu_8$ from the example in Figure 2. The circuit is planar because the modules form a directed path in $\mathcal{M}$. Braces denote *or*-gates, angle brackets denote *and*-gates, and square brackets denote *id*-gates. Variable gates are shown in gray. For constant gates the labeling is omitted.

Lemma 2. *For a directed path $\pi \subseteq M$ in the tree $\mathcal{M}$, the circuit $P = \bigcup_{m \in \pi} m$ is planar. If P is* EV *and all variable gates in P belong to the terminating module of π then P is* OIF*. This property is stable under evaluation of partial circuits of C.*

Proof. The first sentence follows directly from the definition of Emb_C. The second sentence follows from the definition of Emb_C and the observation that an EV circuit with all variables on the outer face is OIF. For the proof of the third sentence, note that evaluation does not add any edge to $\mathrm{gr}(C)$. If the graph of P was planar (OIF) before the evaluation, it is planar (OIF) after the evaluation of any partial circuit of C. □

3 The Evaluation Algorithm

We now present our circuit evaluation algorithm. The problem of evaluating the circuit C from Section 2 is AC^1 Turing reduced to the evaluation problem for OIF circuits. Our algorithm repeatedly evaluates subcircuits of C. In the following, we always refer to the current circuit as C.

The central data structure of our algorithm is the *evaluation tree* $\mathcal{M}_\simeq$, which is the quotient of $\mathcal{M}$ with respect to an equivalence $\simeq$. As the algorithm progresses, more and more of the modules are collected into single nodes of $\mathcal{M}_\simeq$.

We define $\simeq$ as an equivalence relation on the modules of $\mathcal{M}$ such that the equivalence classes of $\simeq$ are full subtrees, i.e., for each equivalence class τ and each node $t \in \tau$, either each child or no child of t in $\mathcal{M}$ is in τ. For a node ν of $\mathcal{M}_\simeq$ we denote the circuit $\bigcup_{m \in \nu} m$ by $\mathrm{cir}(\nu)$. We call the nodes of $\mathcal{M}_\simeq$ the *e*nodes. An *e*node ν is called constant if $\mathrm{cir}(\nu)$ is constant.

Initially, each simple path in $\mathcal{M}$ forms a class. Starting from the leaves of $\mathcal{M}_\simeq$, our algorithm then evaluates the circuits corresponding to adjacent *e*nodes and updates $\simeq$ by collapsing the equivalence classes.

Throughout this process, we maintain the invariant that, for every *e*node, the corresponding partial circuit is OIF. This allows us to apply the evaluation algorithm from Corollary 1 on the partial circuit and, hence, perform the contraction within logDCFL. The process ends when $\mathcal{M}_\simeq$ has been contracted into a single class. At that point, C is fully evaluated.

To ensure the invariant, we maintain that the equivalence relation is *well-formed*, as specified in the following definition:

Definition 3. *The equivalence relation $\simeq$ is well-formed if for each* e*node α of $\mathcal{M}_\simeq$ it holds that*

- *$cir(\alpha)$ is* EV*,*
- *α is a full subtree of $\mathcal{M}$, and*
- *α is either a leaf or there is a single module $bo(\alpha) \in \alpha$ such that there are modules $b, c \in M$ with $b \neq c$, $b, c \notin \alpha$, and $\langle a, b\rangle \in E$ and $\langle a, c\rangle \in E$.*

Together with Lemma 2, well-formedness ensures that for each *e*node, the corresponding circuit is OIF.

Lemma 3. *Let $\simeq$ be well-formed and let α be an enode of $\mathcal{M}_{\simeq}$. It holds that*

- *$\mathcal{M}_{\simeq}$ is a full binary tree,*
- *the circuit cir(α) is* OIF*, and*
- *if α is a leaf in $\mathcal{M}_{\simeq}$ then α is constant.*

Proof. Since *e*nodes are full subtrees of $\mathcal{M}$, the modules b and c from Definition 3 belong to different *e*nodes. Each *e*node is therefore either a leaf or has exactly two child *e*nodes. Hence, $\mathcal{M}_{\simeq}$ is a full binary tree. The uniqueness of $\mathrm{bo}(\alpha)$ implies that all variable gates of $\mathrm{cir}(\alpha)$ belong to $\mathrm{bo}(\alpha)$. Since $\mathrm{cir}(\alpha)$ is EV, all but the ancestor *e*nodes of $\mathrm{bo}(\alpha)$ are constant. Hence, all non-constant modules in α are on the directed path from the root of α to $\mathrm{bo}(\alpha)$. Since $\mathrm{cir}(\alpha)$ is EV, we conclude, by Lemma 2, that $\mathrm{cir}(\alpha)$ is OIF. If α is a leaf in $\mathcal{M}_{\simeq}$, then $\mathrm{cir}(\alpha)$ has no variable gates. Then α is constant, because $\mathrm{cir}(\alpha)$ is EV. □

Initialization. Initially, $\simeq$ is set to be the reflexive, symmetric, and transitive closure of $\simeq'$, where $a \simeq' b$ iff $(a, b) \in E$ and there is no c different from b s.t. $(a, c) \in E$.

$\mathrm{X}_{\exists}$ and $\mathrm{X}_{\forall}$ operators in the LTL formula give rise to modules with only a single child in $\mathcal{M}$. The initialization of $\simeq$ via $\simeq'$ causes the corresponding simple paths in $\mathcal{M}$ to collapse, such that $\mathcal{M}_{\simeq}$ is a full binary tree. Note that all classes of $\simeq$ are subtrees of $\mathcal{M}$.

To ensure well-formedness, we evaluate (in parallel) all non-singleton *e*nodes that contain constants. These are exactly the *e*nodes that correspond to modules originating from $\mathrm{X}_{\exists}$ and $\mathrm{X}_{\forall}$ operators stacked upon a single constant module. From the definition of Emb_C it is clear that those nodes are OIF and thus the evaluation can be performed in parallel within $\mathsf{logDCFL}$, by using Corollary 1.

Lemma 4. *After the initial evaluation, $\simeq$ is well-formed.* □

Tree contraction. Each contraction step combines a leaf *e*node of $\mathcal{M}_{\simeq}$ with its parent and its sibling into a single *e*node. Well-formedness is preserved by evaluating the circuit of the resulting *e*node.

Lemma 5. *Let $\mathcal{M}_{\simeq}$ be well-formed. Given an* e*node α of $\mathcal{M}_{\simeq}$ with child* e*nodes β and γ. Let β be a leaf* e*node. The evaluation of cir$(\alpha \cup \beta \cup \gamma)$ can be performed in* $\mathsf{logDCFL}$*. Updating $\simeq$ such that $\alpha \simeq \beta \simeq \gamma$ preserves well-formedness of $\mathcal{M}_{\simeq}$.*

Proof. Let $\mathcal{A} = \alpha \cup \beta \cup \gamma$. $\mathrm{cir}(\alpha)$ is OIF and β is constant. Thus the circuit $\mathrm{cir}(\alpha \cup \beta \cup \mathrm{const}(\gamma))$ is OIF and can be evaluated in $\mathsf{logDCFL}$. After the evaluation, since $\mathrm{cir}(\gamma)$ is EV, all constants in $\mathrm{cir}(\mathcal{A})$ are sinks, and, hence, $\mathrm{cir}(\mathcal{A})$ is EV. $\mathcal{M}_{\simeq}$ is a full binary tree. Thus the *e*nodes α, β, γ together form a full subtree in $\mathcal{M}_{\simeq}$. $\mathcal{M}_{\simeq}$ is the quotient of $\mathcal{M}$, and α, β, and γ are each full subtrees of $\mathcal{M}$. It follows that $\mathcal{A}$ is a full subtree in $\mathcal{M}$ as well. In the subtree $\mathcal{A}$ of $\mathcal{M}$, the module $\mathrm{bo}(\alpha)$ is an internal node. Since β is a leaf, $\mathrm{bo}(\beta)$ does not exist. If γ is a leaf, $\mathcal{A}$ also becomes a leaf. Otherwise, $\mathrm{bo}(\mathcal{A}) = \mathrm{bo}(\gamma)$. □

Since, as we show in the following lemma, we can a contract a constant portion of the $e$nodes in parallel, the time consumed for the full contraction is logarithmic in the size of $\mathcal{M}$.

Lemma 6. *The circuit $C(\phi, \rho)$ can be evaluated within $AC^1(\mathsf{logDCFL})$.*

Proof. First, number the $e$nodes of $\mathcal{M}_{\simeq}$ that have a child that is a leaf from left to right (using depth first search on the tree, starting with 1) in L. Then, on every odd-numbered $e$node, apply Lemma 5. Since the involved circuits are disjoint for all odd-numbered $e$nodes, all applications of Lemma 5 can be performed in parallel. This eliminates at least $\left\lceil \frac{(\|\mathcal{M}_{\simeq}\|+1)/2}{2} \right\rceil$ leaves from the tree resulting in a tree $\mathcal{M}'_{\simeq}$ with $\|\mathcal{M}'_{\simeq}\| \leq \lfloor 3/4 \|\mathcal{M}_{\simeq}\| \rfloor$. Iterating this procedure leads in $O(\log \|\mathcal{M}_{\simeq}\|)$ steps to a single leaf $e$node. At this point, $C(\phi, \rho)$ is fully evaluated. The whole procedure can be implemented as an AC^1 circuit with $\mathsf{logDCFL}$ oracle gates.

The reduction circuit operates in stages. Each stage is structured as follows: an L oracle gate that takes the current $\mathcal{M}_{\simeq}$ as input identifies the sets of $e$nodes to be contracted on the current stage and feeds these into $\mathsf{logDCFL}$ oracle gates that implement Lemma 5. The remaining $e$nodes are just copied. The output of the stage is the updated version of $\mathcal{M}_{\simeq}$. Since $\mathsf{L} \subseteq \mathsf{logDCFL}$, each stage is of constant depth. A logarithmic number of sequential stages is stacked upon an initialization step that consists of a single L oracle gate that initializes $\mathcal{M}_{\simeq}$ from ϕ and ρ and parallel $\mathsf{logDCFL}$ oracle gates that evaluate $e$nodes that initially are simple paths in $\mathcal{M}$. □

Applying the evaluation algorithm to the circuit defined in Definition 2, we obtain an $\mathsf{AC}^1(\mathsf{logDCFL})$ solution to the path checking problem.

Theorem 2. *The LTL path checking problem is in* $\mathsf{AC}^1(\mathsf{logDCFL})$.

Proof. Given an LTL formula ϕ and a path ρ. In L build the circuit $C(\phi, \rho)$. Apply Lemma 6. The value of $c_{0,0}$ is the result. □

4 Conclusions

We have presented a positive answer to the question whether LTL can be checked efficiently in parallel on finite paths. Our construction can, for example, be used in hardware-based monitors to reduce the time needed to evaluate a block of path positions from linear to just logarithmic.

The LTL path checking problem is closely related to the membership problems for the various types of regular expressions: the membership problem is in NL for regular expressions [14], in logCFL for semi-extended regular expressions [20], and P-complete for star-free regular expressions and extended regular expressions [19]. Of particular interest is the comparison to the star-free regular expressions, since they have the same expressive power as LTL on finite paths [16]. With $\mathsf{AC}^1(\mathsf{logDCFL})$ vs. P, our result demonstrates a computational advantage for LTL.

Tight bounds for the complexity of LTL path checking remain a challenging open problem. There is some hope to further reduce the upper bound towards NC^1, the currently known lower bound, because our construction relies on the algorithm by Chakraborty and Datta (cf. Theorem 1) for evaluating monotone Boolean planar circuits with all constant gates on the outer face. The circuits that appear in our construction actually exhibit much more structure. However, we are not aware of any algorithm that takes advantage of that and performs better than logDCFL.

References

1. Armoni, R., Korchemny, D., Tiemeyer, A., Vardi, M.Y., Zbar, Y.: Deterministic Dynamic Monitors for Linear-Time Assertions. In: Havelund, K., Núñez, M., Roşu, G., Wolff, B. (eds.) FATES 2006 and RV 2006. LNCS, vol. 4262, pp. 163–177. Springer, Heidelberg (2006)
2. Artho, C., Barringer, H., Goldberg, A., Havelund, K., Khurshid, S., Lowry, M., Pasareanu, C., Rosu, G., Sen, K., Visser, W., Washington, R.: Combining test case generation and runtime verification. Theoretical Computer Science 336(2-3), 209–234 (2005)
3. Barrington, D.A.M., Lu, C.-J., Miltersen, P.B., Skyum, S.: On monotone planar circuits. In: Proceedings of the 14th Annual IEEE Conference on Computational Complexity (COCO 1999), Washington, DC, USA, pp. 24–31. IEEE Computer Society Press, Los Alamitos (1999)
4. Boule, M., Zilic, Z.: Automata-based assertion-checker synthesis of PSL properties. ACM Transactions on Design Automation of Electronic Systems (TODAES) 13(1) (2008)
5. Chakraborty, T., Datta, S.: One-input-face MPCVP is hard for L, but in LogDCFL. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 57–68. Springer, Heidelberg (2006)
6. Dahan, A., Geist, D., Gluhovsky, L., Pidan, D., Shapir, G., Wolfsthal, Y., Benalycherif, L., Kamdem, R., Lahbib, Y.: Combining system level modeling with assertion based verification. In: ISQED 2005: Proceedings of the 6th International Symposium on Quality of Electronic Design, Washington, DC, USA, pp. 310–315. IEEE Computer Society, Los Alamitos (2005)
7. Delcher, A.L., Kosaraju, S.R.: An NC algorithm for evaluating monotone planar circuits. SIAM J. Comput. 24(2), 369–375 (1995)
8. Demri, S., Schnoebelen, P.: The complexity of propositional linear temporal logics in simple cases. Inf. Comput. 174(1), 84–103 (2002)
9. Dymond, P.W., Cook, S.A.: Complexity theory of parallel time and hardware. Information and Computation 80(3), 205–226 (1989)
10. Finkbeiner, B., Sipma, H.B.: Checking finite traces using alternating automata. Formal Methods in System Design 24, 101–127 (2004)
11. Goldschlager, L.M.: A space efficient algorithm for the monotone planar circuit value problem. Inf. Process. Lett. 10(1), 25–27 (1980)
12. Havelund, K., Roşu, G.: Efficient monitoring of safety properties. Software Tools for Technology Transfer (2004)
13. IEEE Std 1850-2007. Standard for Property Specification Language (PSL). IEEE, New York (2007)

14. Jiang, T., Ravikumar, B.: A note on the space complexity of some decision problems for finite automata. Information Processing Letters 40, 25–31 (1991)
15. Rao Kosaraju, S.: On parallel evaluation of classes of circuits. In: Veni Madhavan, C.E., Nori, K.V. (eds.) FSTTCS 1990. LNCS, vol. 472, pp. 232–237. Springer, Heidelberg (1990)
16. Lichtenstein, O., Pnueli, A., Zuck, L.D.: The glory of the past. In: Proceedings of the Conference on Logic of Programs, London, UK, pp. 196–218. Springer, London (1985)
17. Limaye, N., Mahajan, M., Sarma, J.M.N.: Evaluating monotone circuits on cylinders, planes and tori. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 660–671. Springer, Heidelberg (2006)
18. Markey, N., Schnoebelen, P.: Model checking a path (preliminary report). In: Amadio, R.M., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 251–265. Springer, Heidelberg (2003)
19. Petersen, H.: Decision problems for generalized regular expressions. In: Proc. 2nd International Workshop on Descriptional Complexity of Automata, Grammars and Related Structures, London (Ontario), pp. 22–29 (2000)
20. Petersen, H.: The membership problem for regular expressions with intersection is complete in LOGCFL. In: Alt, H., Ferreira, A. (eds.) STACS 2002. LNCS, vol. 2285, pp. 513–522. Springer, Heidelberg (2002)
21. Yang, H.: An NC algorithm for the general planar monotone circuit value problem. In: Proc. 3rd IEEE Symposium on Parallel and Distributed Processing, pp. 196–203 (1991)
22. Younes, H.L.S., Simmons, R.G.: Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, p. 223. Springer, Heidelberg (2002)

An Explicit Formula for the Free Exponential Modality of Linear Logic[⋆]

Paul-André Melliès, Nicolas Tabareau, and Christine Tasson

Laboratoire Preuves Programmes Systèmes
CNRS & Université Paris 7 - Denis Diderot

Abstract. The exponential modality of linear logic associates a commutative comonoid $!A$ to every formula A in order to duplicate it. Here, we explain how to compute the free commutative comonoid $!A$ as a sequential limit of equalizers in any symmetric monoidal category where this sequential limit exists and commutes with the tensor product. We then apply this general recipe to two familiar models of linear logic, based on coherence spaces and on Conway games. This algebraic approach enables to unify for the first time apparently different constructions of the exponential modality in spaces and games. It also sheds light on the subtle duplication policy of linear logic. On the other hand, we explain at the end of the article why the formula does not work in the case of the finiteness space model.

1 Introduction

Linear logic is based on the principle that every hypothesis A_i should appear exactly once in a proof of the sequent

$$A_1, \ldots, A_n \quad \vdash \quad B. \tag{1}$$

This logical restriction enables to represent the logic in monoidal categories, along the idea that every formula denotes an object of the category, and every proof of the sequent (1) denotes a morphism

$$A_1 \otimes \cdots \otimes A_n \quad \longrightarrow \quad B$$

where the tensor product is thus seen as a linear kind of conjunction. Note that, for clarity's sake, we use the same notation for a formula A and for its interpretation (or denotation) in the monoidal category.

This linearity policy on proofs is far too restrictive in order to reflect traditional forms of reasoning, where it is accepted to repeat or to discard an hypothesis in the course of a logical argument. This difficulty is nicely resolved by providing linear logic with an exponential modality, whose task is to strengthen every formula A into a formula $!A$ which may be repeated or discarded. From a semantic point of view, the formula $!A$ is most naturally interpreted as a *comonoid*

[⋆] This work has been supported by the ANR Curry-Howard Correspondence and Concurrency Theory (CHOCO).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 247–260, 2009.

of the monoidal category. Recall that a comonoid (C, d, u) in a monoidal category $\mathscr{C}$ is defined as an object C equipped with two morphisms

$$d \quad : \quad C \quad \longrightarrow \quad C \otimes C \qquad\qquad\qquad u \quad : \quad C \quad \longrightarrow \quad \mathbf{1}$$

where $\mathbf{1}$ denotes the monoidal unit of the category. The morphism d and u are respectively called the *multiplication* and the *unit* of the comonoid. The two morphisms d and u are supposed to satisfy *associativity* and *unitality* properties, neatly formulated by requiring that the two diagrams

$$\begin{tikzcd}
& C \arrow[dl, bend right, "d"'] \arrow[dr, bend left, "d"] & \\
C \otimes C \arrow[dr, bend right, "C\otimes d"'] & & C \otimes C \arrow[dl, bend left, "d\otimes C"] \\
& C \otimes C \otimes C &
\end{tikzcd}
\qquad
\begin{tikzcd}
& C \arrow[dl, bend right, "d"'] \arrow[dr, bend left, "d"] \arrow[dd, "id"] & \\
C \otimes C \arrow[dr, bend right, "u\otimes C"'] & & C \otimes C \arrow[dl, bend left, "C\otimes u"] \\
& C &
\end{tikzcd}$$

commute. Note that we draw our diagrams as if the category were *strictly* monoidal, although the usual models of linear logic are only *weakly* monoidal.

The comonoidal structure of the formula $!A$ enables to interpret the *contraction rule* and the *weakening rule* of linear logic

$$\dfrac{\dfrac{\overset{\pi}{\vdots}}{\Gamma, !A, !A, \Delta \vdash B}}{\Gamma, !A, \Delta \vdash B}\ \text{Contraction} \qquad\qquad \dfrac{\dfrac{\overset{\pi}{\vdots}}{\Gamma, \Delta \vdash B}}{\Gamma, !A, \Delta \vdash B}\ \text{Weakening}$$

by pre-composing the interpretation of the proof π with the multiplication d in the case of contraction

$$\Gamma \otimes !A \otimes \Delta \xrightarrow{d} \Gamma \otimes !A \otimes !A \otimes \Delta \xrightarrow{\pi} B$$

and with the unit u in the case of weakening

$$\Gamma \otimes !A \otimes \Delta \xrightarrow{u} \Gamma \otimes \Delta \xrightarrow{\pi} B.$$

Besides, linear logic is generally interpreted in a *symmetric* monoidal category, and one requires that the comonoid $!A$ is commutative, this meaning that the following equality holds:

$$A \xrightarrow{\quad d \quad} A \otimes A \xrightarrow{symmetry} A \otimes A \quad = \quad A \xrightarrow{\quad d \quad} A \otimes A\ .$$

When linear logic was introduced by Jean-Yves Girard, twenty years ago, it was soon realized by Robert Seely and others that the multiplicative fragment of the logic should be interpreted in a $*$-autonomous category, or at least, a symmetric monoidal closed category $\mathscr{C}$; and that the category should have finite products in order to interpret the additive fragment of the logic, see [10]. A more difficult question was to understand what categorical properties of the exponential modality " ! " were exactly required, in order to define a model of propositional linear logic – that is, including the multiplicative, additive and exponential components of the logic. However, Yves Lafont found in his PhD thesis [6] a simple way to define a model of linear logic. Recall that a comonoid morphism between two comonoids (C_1, d_1, u_1) and (C_2, d_2, u_2) is defined as a morphism $f : C_1 \to C_2$ such that the two diagrams

$$\begin{tikzcd}
C_1 \arrow[r, "f"] \arrow[d, "d_1"'] & C_2 \arrow[d, "d_2"] \\
C_1 \otimes C_1 \arrow[r, "f \otimes f"] & C_2 \otimes C_2
\end{tikzcd}
\qquad
\begin{tikzcd}
C_1 \arrow[rr, "f"] \arrow[dr, bend right, "u_1"'] & & C_2 \arrow[dl, bend left, "u_2"] \\
& \mathbf{1} &
\end{tikzcd}$$

commute. One says that the commutative comonoid $!A$ is freely generated by an object A when there exists a morphism

$$\varepsilon \quad : \quad !A \quad \to \quad A$$

such that for every morphism

$$f \quad : \quad C \quad \to \quad A$$

from a commutative comonoid C to the object A, there exists a unique comonoid morphism

$$f^\dagger \quad : \quad C \quad \to \quad !A$$

such that the diagram

$$\begin{tikzcd}
& !A \arrow[dd, "\varepsilon"] \\
C \arrow[ur, bend left, "f^\dagger"] \arrow[dr, bend right, "f"'] & \\
& A
\end{tikzcd} \qquad (2)$$

commutes. From a logical point of view, $!A$ is the weakest comonoid that implies A. Lafont noticed that the existence of a free commutative comonoid $!A$ for every object A of a symmetric monoidal closed category $\mathscr{C}$ induces automatically a model of propositional linear logic. Recall however that this is not the only way to construct a model of linear logic. A folklore example is the coherence space model, which admits two alternative interpretations of the exponential modality: the original one, formulated by Girard [3] where the coherence space $!A$ is defined as a space of *cliques*, and the free construction, where $!A$ is defined as a space of *multicliques* (cliques with multiplicity) of the original coherence space A.

In this paper, we explain how to construct the free commutative comonoid in the symmetric monoidal categories $\mathscr{C}$ typically encountered in the semantics of linear logic. Our starting point is the well-known formula defining the *symmetric algebra*

$$SA \quad = \quad \bigoplus_{n \in \mathbb{N}} \quad A^{\otimes n} \ / \ \sim_n \qquad (3)$$

generated by a vector space A. Recall that the formula (3) computes the free commutative monoid associated to the object A in the category of vector spaces over a given field $\Bbbk$. The group Σ_n of permutations on $\{1, \ldots, n\}$ acts on the vector space $A^{\otimes n}$, and the vector space $A^{\otimes n}/\sim_n$ of equivalence classes (or orbits) modulo the group action is defined as the coequalizer of the $n!$ symmetries

$$\begin{tikzcd}
A^{\otimes n} \arrow[r, shift left=2, "\mathit{symmetry}"] \arrow[r, shift right=2, "\mathit{symmetry}"'] \arrow[r, phantom, "\cdots"] & A^{\otimes n} \arrow[r, dotted, "\mathit{coequalizer}"] & A^{\otimes n}/\sim_n
\end{tikzcd}$$

in the category of vector spaces. Since a comonoid in the category $\mathscr{C}$ is the same thing as a monoid in the opposite category $\mathscr{C}^{op}$, it is tempting to apply the *dual* formula to (3) in order to define the free commutative comonoid $!A$ generated by an object A in the monoidal category $\mathscr{C}$. Although the idea is extremely naive, it is surprisingly close to the solution... Indeed, one significant aspect of our work is to establish that the equalizer A^n of the $n!$ symmetries

$$\begin{tikzcd}
A^{n} \arrow[r, dotted, "\mathit{equalizer}"] & A^{\otimes n} \arrow[r, shift left=2, "\mathit{symmetry}"] \arrow[r, shift right=2, "\mathit{symmetry}"'] \arrow[r, phantom, "\cdots"] & A^{\otimes n}
\end{tikzcd} \qquad (4)$$

exists in several distinctive models of linear logic, and provides there the n-th layer of the free commutative comonoid $!A$ generated by the object A. This principle will be nicely illustrated in Section 3 by the equalizer A^n in the category of coherence spaces, which contains the multicliques of cardinality n in the coherence space A ; and in Section 4 by the equalizer A^n in the category of Conway games, which defines the game where Opponent may open up to n copies of the game A, one after the other, in a sequential order.

Of course, the construction of the free exponential modality does not stop here: one still needs to combine the layers A^n together in order to define $!A$ properly. One obvious solution is to apply the dual of formula (3) and to define $!A$ as the infinite cartesian product

$$!A \quad = \quad \mathop{\Large\&}\limits_{n\in\mathbb{N}} \quad A^n. \tag{5}$$

This formula works perfectly well for symmetric monoidal categories $\mathscr{C}$ where the infinite product commutes with the tensor product, in the sense that the canonical morphism

$$X \quad \otimes \quad \Big(\mathop{\Large\&}\limits_{n\in\mathbb{N}} \quad A^n \Big) \quad \rightarrow \quad \mathop{\Large\&}\limits_{n\in\mathbb{N}} \; (\, X \; \otimes \; A^n \,) \tag{6}$$

is an isomorphism. This useful algebraic degeneracy is not entirely uncommon: it typically happens in the relational model of linear logic, where the free exponential $!A$ is defined according to formula (5) as the set of finite multisets of A, each equalizer A^n describing the set of multisets of cardinality n.

On the other hand, the formula (5) is far too optimistic in general, and does not work when one considers the familiar models of linear logic based either on coherence spaces, or on sequential games. It is quite instructive to apply the formula to the category of Conway games: it defines a game $!A$ where the first move by Opponent selects a component A^n, and thus decides the number n of copies of the game A played subsequently. This departs from the free commutative comonoid $!A$ which we shall examine in Section 4, where Opponent is allowed to open a new copy of the game A at any point of the interaction.

So, there remains to understand how the various layers A^n should be combined together inside $!A$ in order to perform this particular copy policy. One well-inspired temptation is to ask that every layer A^n is "glued" inside the next layer A^{n+1} in order to allow the computation to transit from one layer to the next in the course of interaction. One simple way to perform this "glueing" is to introduce the notion of (co)pointed (or affine) object. By pointed object in a monoidal category $\mathscr{C}$, one means a pair (A, u) consisting of an object A and of a morphism $u : A \to \mathbf{1}$ to the monoidal unit. So, a pointed object is the same thing as a comonoid, without a comultiplication. It is folklore that the category $\mathscr{C}_\bullet$ of pointed objects and pointed morphisms (defined in the expected way) is symmetric monoidal, and moreover *affine* in the sense that its monoidal unit $\mathbf{1}$ is terminal.

The main purpose of this paper is to compute (in Section 2) the free commutative comonoid $!A$ of the category $\mathscr{C}$ as a sequential limit of equalizers. The construction is excessively simple and works every time the sequential limit exists in the category $\mathscr{C}$, and commutes with the tensor product. We establish that the category of coherence spaces (in Section 3) and the category of Conway games (in Section 4) fulfill these hypotheses. This establishes that despite their difference in style, the free exponential modalities are defined in *exactly* the same way in the two models. We then clarify (in Section 5) the topological reasons why neither formula (5) nor the sequential limit of equalizers formulated below (9) define the free exponential modality in the finiteness space model of linear logic recently introduced by Thomas Ehrhard [2].

2 The Sequential Limit Construction

Before stating the general proposition, we present the construction in three steps.

First step. We make the mild hypothesis that the object A of the monoidal category $\mathscr{C}$ generates a free pointed object $(A_\bullet, u)$ in the affine category $\mathscr{C}_\bullet$. This typically happens when the forgetful functor $\mathscr{C}_\bullet \to \mathscr{C}$ has a right adjoint. Informally speaking, the purpose of the pointed object $A_\bullet$ is to describe one copy of the object A, or none... Note that this free pointed object is usually quite easy to define: in the case of coherence spaces, it is the space $A_\bullet = A \,\&\, \mathbf{1}$ obtained by adding a point to the web of A ; in the case of Conway games, it is the game $A_\bullet = A$ itself, at least when the category is restricted to Opponent-starting games.

Second step. The object $A^{\leq n}$ is then defined as the equalizer $(A_\bullet)^n$ of the diagram

$$\begin{tikzcd} A^{\leq n} \arrow[r, dotted, "equalizer"] & A_\bullet^{\otimes n} \arrow[r, shift left, "symmetry"] \arrow[r, phantom, "\cdots"] \arrow[r, shift right, "symmetry"'] & A_\bullet^{\otimes n} \end{tikzcd} \tag{7}$$

in the category $\mathscr{C}$. The purpose of $A^{\leq n}$ is to describe all the layers A^k at the same time, for $k \leq n$. Typically, the object $A^{\leq n}$ computed in the category of coherence spaces is the space of all multicliques in A of cardinality less than or equal to n.

Third step. We take advantage of the existence of a canonical morphism $A^{\leq n} \longleftarrow A^{\leq n+1}$ induced by the unit $u : A_\bullet \to \mathbf{1}$ of the pointed object $A_\bullet$, and define the object A^∞ as the sequential limit of the sequence

$$\begin{tikzcd} 1 & A^{\leq 1} \arrow[l] & A^{\leq 2} \arrow[l] & \cdots \arrow[l] & A^{\leq n} \arrow[l] & A^{\leq n+1} \arrow[l] & \cdots \arrow[l] \end{tikzcd} \tag{8}$$

with limiting cone defined by projection maps

$$\begin{tikzcd} A^\infty \arrow[r, dotted, "projection"] & A^{\leq n}. \end{tikzcd}$$

The 2-dimensional study of algebraic theories and PROPs recently performed by Melliès and Tabareau [8] ensures that this recipe in three steps defines the free commutative comonoid $!A$ as the sequential limit A^∞... when the object A satisfies the following limit properties in the category $\mathscr{C}$.

Proposition 1. *Consider an object A in a symmetric monoidal category $\mathscr{C}$. Suppose that the object A generates a free pointed object $(A_\bullet, u)$. Suppose moreover that the equalizer (7) and the sequential limit (8) exist and commute with the tensor product, in the sense that*

$$\begin{tikzcd} X \otimes A^{\leq n} \arrow[r, dotted, "X\otimes\ equalizer"] & A_\bullet^{\otimes n} \arrow[r, shift left, "X\otimes\ symmetry"] \arrow[r, phantom, "\cdots"] \arrow[r, shift right, "X\otimes\ symmetry"'] & X \otimes A_\bullet^{\otimes n} \end{tikzcd}$$

defines an equalizer diagram, and the family of maps

$$\begin{tikzcd} X \otimes A^\infty \arrow[r, dotted, "X\otimes\ projection"] & X \otimes A^{\leq n} \end{tikzcd}$$

defines a limiting cone, for every object X of the category $\mathscr{C}$. In that case, the free commutative comonoid $!A$ coincides with the sequential limit A^∞.

The proof of Proposition 1 is based on two observations. The first observation is that the category $\mathscr{C}_\bullet$ coincides with the slice category $\mathscr{C} \downarrow \mathbf{1}$, this implying that the forgetful functor $\mathscr{C}_\bullet \to \mathscr{C}$ creates limits. Consequently, the limiting process defining the object A^∞ in the category $\mathscr{C}$ may be alternatively carried out in the category $\mathscr{C}_\bullet$. The second observation is that the limiting process defining A^∞ provides a pedestrian way to compute the end formula

$$A^\infty \quad = \quad \int_{n \in Inj^{op}} FinSet(n, 1) \otimes (A_\bullet)^{\otimes n} \quad = \quad \int_{n \in Inj^{op}} (A_\bullet)^{\otimes n} \tag{9}$$

in the category $\mathscr{C}_\bullet$. As explained in our work on categorical model theory [8], this end formula provides an explicit computation of the object $Ran_f A_\bullet(1)$, where $Ran_f A_\bullet : FinSet^{op} \to \mathscr{C}_\bullet$ denotes the right Kan extension of the pointed object $A_\bullet : Inj^{op} \to \mathscr{C}_\bullet$ along the change of basis $f : Inj^{op} \to FinSet^{op}$ going from the theory of pointed objects to the theory of commutative comonoids. Recall that the category Inj has finite ordinals $[n] = \{0, \ldots, n-1\}$ as objects, and injections $[p] \to [q]$ as morphisms, whereas the category $FinSet$ has functions $[p] \to [q]$ as morphisms. Proposition 1 says that the end formula defines the free commutative comonoid when the end exists and commutes with the tensor product.

3 Coherence Spaces

In this section, we compute the free exponential modality in the category of coherence spaces defined by Jean-Yves Girard [3]. A *coherence space* $E = (|E|, \frown$) consists of a set $|E|$ called its *web*, and of a binary reflexive and symmetric relation $\frown$ over E. A *clique* of E is a set X of pairwise coherent elements of the web:

$$\forall e_1, e_2 \in X, \qquad e_1 \frown e_2.$$

We do not recall here the definition of the category **Coh** of coherence spaces. Just remember that a morphism $R : E \to E'$ in **Coh** is a clique of the coherence space $E \multimap E'$, so in particular, R is a relation on the web $|E| \times |E'|$.

It is easy to see that the tensor product does not commute with cartesian products: simply observe that the canonical morphism

$$A \otimes (\mathbf{1} \,\&\, \mathbf{1}) \quad \to \quad (A \otimes \mathbf{1}) \,\&\, (A \otimes \mathbf{1})$$

is not an isomorphism. This explains why formula (5) does not work, and why the construction of the free exponential modality requires a sequential limit, along the line described in the introduction.

First step: compute the free affine object. Computing the free pointed (or affine) object on a coherence space E is easy, because the category **Coh** has cartesian products: it is simply given by the formula

$$E_\bullet = E \,\&\, 1.$$

It is useful to think of $E \,\&\, 1$ has the space of multicliques of E with *at most* one element: the very first layer of the construction of the free exponential modality. Indeed, the unique element of 1 may be seen as the empty clique, while every element e of E may be seen as the singleton clique $\{e\}$. Recall that a multiclique of E is just a multiset on $|E|$ whose underlying set is a clique of E.

Second step: compute the symmetric tensor power $E^{\leq n}$. It is not difficult to see that the equalizer $E^{\leq n}$ of the symmetries

$$\begin{tikzcd}[column sep=huge]
(E \,\&\, \mathbf{1})^{\otimes n} \arrow[r, shift left=2, "symmetry"] \arrow[r, phantom, "\cdots"] \arrow[r, shift right=2, "symmetry"'] & (E \,\&\, \mathbf{1})^{\otimes n}
\end{tikzcd}$$

is given by the set of multicliques of E with at most n elements, two multicliques being coherent iff their union is still a multiclique. As explained in the introduction, one also needs to check that the tensor product commutes with those equalizers. Consider a cone

$$\begin{tikzcd}[column sep=huge]
& Y \arrow[dl, bend right, "R"'] \arrow[dr, bend left, "R'"] & \\
X \otimes (E \,\&\, \mathbf{1})^{\otimes n} \arrow[rr, shift left=2, "X\otimes symmetry"] \arrow[rr, phantom, "\cdots"] \arrow[rr, shift right=2, "X\otimes symmetry"'] & & X \otimes (E \,\&\, \mathbf{1})^{\otimes n}
\end{tikzcd} \tag{10}$$

First, observe that $R = R'$ because one may choose the identity among the $n!$ symmetries. Next, we show that the morphism R factors uniquely through the morphism

$$\begin{tikzcd}[column sep=huge]
X \otimes E^{\leq n} \arrow[r, dotted, "X\otimes\ equalizer"] & X \otimes (E \,\&\, \mathbf{1})^{\otimes n}
\end{tikzcd}$$

To that purpose, one defines the relation

$$R^{\leq n} \ : \ Y \ \longrightarrow \ X \otimes E^{\leq n} \qquad \text{by} \qquad y \ R^{\leq n} \ (x, \mu) \quad \text{iff} \quad y \ R \ (x, u)$$

where μ is a multiset of $|E|$ of cardinal less than n, and u is any word of length n whose letters with multiplicity in $|E \,\&\, \mathbf{1}| = |E| \sqcup \{*\}$ define the multiset μ. Remark that the fact that R equalizes the symmetries implies that any u' defining the same multiset μ will also be in the relation: $y \ R \ (x, u')$. We let the reader check that the definition is correct, that it defines a clique $R^{\leq n}$ of $Y \multimap (X \otimes E^{\leq n})$, and that it is the unique way to factor R through (10).

Third step: compute the sequential limit

$$\begin{tikzcd}
E^{\leq 0} = \mathbf{1} & E^{\leq 1} = (E \,\&\, \mathbf{1}) \arrow[l] & E^{\leq 2} \arrow[l] & E^{\leq 3} \cdots \arrow[l]
\end{tikzcd}$$

whose arrows are (dualized) inclusions from $E^{\leq n}$ into $E^{\leq n+1}$. Again, it is a basic fact that the limit $!E$ of the diagram is given by the set of all finite multicliques, two multicliques being coherent iff their union is a multiclique. At this point, one needs to check that the sequential limit commutes with the tensor product. Consider a cone

$$\begin{tikzcd}
& & Y \arrow[dll, bend right, "R_0"'] \arrow[dl, bend right, "R_1"] \arrow[d, "R_2"'] \arrow[dr, bend left, "R_3"] & \\
X \otimes 1 & X \otimes (E \,\&\, 1) \arrow[l] & X \otimes E^{\leq 2} \arrow[l] & X \otimes E^{\leq 3} \cdots \arrow[l]
\end{tikzcd}$$

and define the relation

$$R_\infty \ : \ Y \ \longrightarrow \ X \otimes !E \qquad \text{by} \qquad y \ R_\infty \ (x, \mu) \quad \text{iff} \quad \exists n, \quad y \ R_n \ (x, u)$$

where μ is a multiset of elements of $|E|$ and the element u of the web of $E^{\leq n}$ is any word of length n whose letters with multiplicity in $|E \& \mathbf{1}| = |E| \sqcup \{*\}$ define the multiset μ. We let the reader check that R_∞ is a clique of $Y \multimap (X \otimes !E)$ and defines the unique way to factor the cone. This concludes the proof that the sequential limit $!E$ defines the free commutative comonoid generated by E in the category **Coh** of coherence spaces.

4 Conway Games

In this section, we compute the free exponential modality in the category of Conway games introduced by André Joyal in [4]. One unifying aspect of our approach is that the construction works in exactly the same way as for coherence spaces.

Conway games. A *Conway game* A is an oriented rooted graph (V_A, E_A, λ_A) consisting of (1) a set V_A of vertices called the *positions* of the game; (2) a set $E_A \subset V_A \times V_A$ of edges called the *moves* of the game; (3) a function $\lambda_A : E_A \to \{-1, +1\}$ indicating whether a move is played by Opponent (-1) or by Proponent $(+1)$. We write $\star_A$ for the root of the underlying graph. A Conway game is called *negative* when all the moves starting from its root are played by Opponent.
A *play* $s = m_1 \cdot m_2 \cdot \ldots \cdot m_{k-1} \cdot m_k$ of a Conway game A is a path $s : \star_A \twoheadrightarrow x_k$ starting from the root $\star_A$

$$s : \star_A \xrightarrow{m_1} x_1 \xrightarrow{m_2} \ldots \xrightarrow{m_{k-1}} x_{k-1} \xrightarrow{m_k} x_k$$

Two paths are parallel when they have the same initial and final positions. A play is *alternating* when

$$\forall i \in \{1, \ldots, k-1\}, \qquad \lambda_A(m_{i+1}) = -\lambda_A(m_i).$$

We note Play_A the set of plays of a game A.

Dual. Every Conway game A induces a *dual* game A^* obtained simply by reversing the polarity of moves.

Tensor product. The tensor product $A \otimes B$ of two Conway games A and B is essentially the asynchronous product of the two underlying graphs. More formally, it is defined as:

- $V_{A \otimes B} = V_A \times V_B$,
- its moves are of two kinds :

$$x \otimes y \to \begin{cases} z \otimes y \text{ if } x \to z \text{ in the game } A \\ x \otimes z \text{ if } y \to z \text{ in the game } B, \end{cases}$$

- the polarity of a move in $A \otimes B$ is the same as the polarity of the underlying move in the component A or the component B.

The unique Conway game 1 with a unique position $\star$ and no move is the neutral element of the tensor product. As usual in game semantics, every play s of the game $A \otimes B$ can be seen as the interleaving of a play $s_{|A}$ of the game A and a play $s_{|B}$ of the game B.

Strategies. Remark that the definition of a Conway game does not imply that all the plays are alternating. The notion of alternation between Opponent and Proponent only appears at the level of strategies (i.e. programs) and not at the level of games (i.e. types). A *strategy* σ of a Conway game A is defined as a non empty set of *alternating plays* of even length such that (1) every non empty play starts with an Opponent move; (2) σ is closed by even length prefix; (3) σ is *deterministic*, i.e. for all plays s, and for all moves m, n, n',

$$s \cdot m \cdot n \in \sigma \ \wedge \ s \cdot m \cdot n' \in \sigma \ \Rightarrow \ n = n'.$$

The category of Conway games. The category **Conway** has Conway games as objects, and strategies σ of $A^* \otimes B$ as morphisms $\sigma : A \to B$. The composition is based on the usual "parallel composition plus hiding" technique and the identity is defined by a copycat strategy. The resulting category **Conway** is compact-closed in the sense of [5].

It appears that the category **Conway** does not have finite nor infinite products [9]. For that reason, we compute the free exponential modality in the full subcategory $\mathbf{Conway}^{-}$ of negative Conway games, which is symmetric monoidal closed, and has products. We explain in a later stage how the free construction on the subcategory $\mathbf{Conway}^{-}$ induces a free construction on the whole category.

First step: compute the free affine object. The monoidal unit $\mathbf{1}$ is terminal in the category $\mathbf{Conway}^{-}$. In other words, every negative Conway game may be seen as an affine object in a unique way, by equipping it with the empty strategy $t_A : A \to 1$. In particular, the free affine object $A_\bullet$ is simply A itself.

Second step: compute the symmetric tensor power. A simple argument shows that the equalizer $A^n = A^{\leq n}$ of (7) is the following Conway game:

- the positions of the game A^n are the finite words $w = x_1 \cdots x_n$ of length n, whose letters are positions x_i of the game A, and such that $x_{i+1} = \star_A$ is the root of A whenever $x_i = \star_A$ is the root of A, for every $1 \leq i < n$. The intuition is that the letter x_k in the position $w = x_1 \cdots x_n$ of the game A^n describes the position of the k-th copy of A, and that the $i+1$-th copy of A cannot be opened by Opponent unless all the i-th copy of A has been already opened.
- its root is the word $\star_{A^n} = \star_A \cdots \star_A$ where the n the positions x_k are at the root $\star_A$ of the game A,
- a move $w \to w'$ is a move played in one copy:

$$w_1 \ x \ w_2 \ \to \ \ w_1 \ y \ w_2$$

where $x \to y$ is a move of the game A. Note that the condition on the positions implies that when a new copy of A is opened (that is, when $x = \star_A$) no position in w_1 is at the root, and all the positions in w_2 are at the root.
- the polarities of moves are inherited from the game A in the obvious way.

Note that A^n may be also seen as the subgame of $A^{\otimes n}$ where the $i+1$-th copy of A is always opened after the i-th copy of A.

Third step: compute the sequential limit. We now consider Diagram (8)

$$\begin{tikzcd} A^0 = 1 & A^1 = A \arrow[l] & A^2 \arrow[l] & A^3 \arrow[l] & \cdots \arrow[l] \end{tikzcd}$$

whose morphisms are the partial copycat strategies $A^n \leftarrow A^{n+1}$ identifying A^n as the subgame of A^{n+1} where only the first n copies of A are played. The limit of this diagram in the category $\mathbf{Conway}^{-}$ is the game A^∞ defined in the same way as $A^{\leq n}$ except that its positions $w = x_1 \cdot x_2 \cdots$ are infinite sequences of positions of A, all of them at the root except for a finite prefix $x_1 \cdots x_k$. It is possible to show that A^∞ is indeed the limit of this diagram, and that the tensor product commutes with this limit. From this, we deduce that the sequential limit A^∞ describes the free commutative comonoid in the category $\mathbf{Conway}^{-}$.

It is nice to observe that the free construction extends to the whole category **Conway** of Conway games. Indeed, one shows easily that every commutative comonoid in the category of Conway games is in fact a negative game. Moreover, the inclusion functor from $\mathbf{Conway}^{-}$ to **Conway** has a right adjoint, which associates to every Conway game A, the negative Conway game A^{-} obtained by removing all the Proponent moves from the root $\star_A$. By combining these two observations, we obtain that $(A^{-})^\infty$ is the free commutative comonoid generated by a Conway game A in the category **Conway**.

5 Finiteness Spaces – An Inviting Counter-Example

In Sections 3 and 4 we have seen how to refine Formula (5) into Formula (9) in order to compute the free exponential modality in the coherence space and the Conway game models. We conclude the paper by explaining why the two formulas do not work in the finiteness space model. Recall that there are two levels of finiteness spaces. On the one hand, *relational* finiteness spaces constitute a refinement of the relational model, while on the other hand *linear* finiteness spaces are linearly topologized vector spaces [7] built on the relational layer. We explain the failure of our two formulas at both levels. We refer the reader to [2] for an introduction to finiteness spaces.

Relational finiteness spaces. Two subsets u, u' of a countable set $\mathbb{E}$ are called orthogonal, denoted by $u \perp u'$, whenever their intersection $u \cap u'$ is finite. The orthogonal of $\mathcal{G} \subseteq \mathcal{P}(\mathbb{E})$ is then defined by $\mathcal{G}^\perp = \{u' \subseteq \mathbb{E} \,|\, \forall u \in \mathcal{G},\ u \perp u'\}$.

A *relational finiteness space* $E = (|E|, \mathcal{F}(E))$ is given by its *web* (a countable set $|E|$) and by a set $\mathcal{F}(E) \subseteq \mathcal{P}(|E|)$ orthogonally closed, i.e. such that $\mathcal{F}(E)^{\perp\perp} = \mathcal{F}(E)$. The elements of $\mathcal{F}(E)$ (resp. $\mathcal{F}(E)^{\perp}$) are called *finitary* (resp. *antifinitary*). A finitary relation R between two finiteness spaces E_1 and E_2 is a subset of $|E_1| \times |E_2|$ such that

$$\forall u \in \mathcal{F}(E_1),\ R \cdot u \ := \big\{ b \in |E_2| \ \big|\ \exists a \in u,\, (a,b) \in R \big\} \in \mathcal{F}(E_2),$$
$$\forall v' \in \mathcal{F}(E_2)^{\perp},\ {}^{t}R \cdot v' := \big\{ a \in |E_1| \ \big|\ \exists b \in v',\, (a,b) \in R \big\} \in \mathcal{F}(E_1)^{\perp}.$$

The category **RelFin** of relational finiteness spaces and finitary relations is $*$-autonomous. As such, it provides a model of multiplicative linear logic (MLL).

The exponential modality ! is then defined as follows [2]: given a finiteness space E, the finiteness space $!E$ has its web $|!E| = \mathcal{M}_{\mathrm{fin}}(|E|)$ defined as the set of finite multisets $\mu : |E| \to \mathbb{N}$ and its finiteness structure defined as

$$\mathcal{F}(!E) = \{ M \in \mathcal{M}_{\mathrm{fin}}(|E|) \mid \mathit{\Pi}_E(M) \in \mathcal{F}(E) \},$$

where for every $M \in \mathcal{M}_{\mathrm{fin}}(|E|)$, $\mathit{\Pi}_E(M) \overset{\mathrm{def}}{=} \{ x \in |E| \mid \exists \mu \in M,\, \mu(x) \neq 0 \}$.

Given a finiteness space E, let us compute the finiteness space E^{∞} defined by Formula (9). The free pointed space generated by E exists, and is defined as

$$E_{\bullet} \overset{\mathrm{def}}{=} E \,\&\, 1.$$

The equalizer $E^{\leq n}$ of the $n!$ symmetries exists in **RelFin** and provides the n-th layer of $!E$. Its web $|E^{\leq n}| = \mathcal{M}_{\mathrm{fin}}^{\leq n}(|E|)$ consists of the multisets of cardinality at most n and its finiteness structure is defined as

$$\mathcal{F}(E^{\leq n}) \quad = \quad \{ \quad M_n \subseteq \mathcal{M}_{\mathrm{fin}}^{\leq n}(|E|) \quad | \quad \mathit{\Pi}_E(M_n) \in \mathcal{F}(E) \quad \}.$$

Finally, the limit defined by Formula (9) is given by the finiteness space E^{∞} whose web is $|E^{\infty}| = \mathcal{M}_{\mathrm{fin}}(|E|)$ and whose finiteness structure is

$$\mathcal{F}\left(E^{\infty}\right) \quad = \quad \left\{ \quad M \in \mathcal{M}_{\mathrm{fin}}(|E|) \quad \middle| \quad \forall n \in \mathbb{N}, \quad \begin{array}{l} M_n = M \cap \mathcal{M}_{\mathrm{fin}}^{\leq n}(|E|), \\ \mathit{\Pi}_E(M_n) \in \mathcal{F}(E). \end{array} \quad \right\}.$$

Note that the webs of $!E$ and of E^{∞} are equal, and coincide in fact with the free exponential in the relational model. However, it is obvious that the finiteness structures of $!E$ and E^{∞} do not coincide in general:

$$\mathcal{F}(!E) \quad \subsetneq \quad \mathcal{F}\left(E^{\infty}\right).$$

In fact, Formula (9) does not work here because the sequential limit (8) does not commute with the tensor product. This phenomenon comes from the fact that an infinite directed union of finitary sets is not necessarily finitary in the finiteness space model – whereas an infinite directed union of cliques is a clique in the coherence space model, this explaining the success of Formula (9) in this model. The interested reader will check that Formula (5) computes the same finiteness space E^{∞} as Formula (9) because $E^{\leq n}$ coincides with the cartesian product of E^k for $k \leq n$. We now turn to the topological version of finiteness spaces to understand the topological difference between $!E$ and E^{∞}.

Linear finiteness spaces. Let $\Bbbk$ be an infinite field endowed with the discrete topology. Every relational finiteness space E generates a vector space, the *linear finiteness space*

$$\Bbbk\langle E\rangle = \{x \in \Bbbk^{|E|} \mid |x| \in \mathcal{F}(E)\},$$

where for any sequence $x \in \Bbbk^{|E|}$, $|x| = \{a \in |E| \mid x_a \neq 0\}$. Endowed with a topology defined with respect to the antifinitary parts, $\Bbbk\langle E\rangle$ is a linearly topologized space [7]. The category **LinFin**, with linear finiteness spaces as objects and linear continuous functions as morphisms, is $*$-autonomous and provides a model of MLL.

We now consider $\Bbbk\langle E^\infty\rangle$ and $\Bbbk\langle !E\rangle$, or more precisely their duals since the functional definition is more intuitive. In **LinFin**, the dual space $\Bbbk\langle E\rangle^\perp = (\Bbbk\langle E\rangle \multimap \Bbbk)$ consists of continuous linear forms and is endowed with the topology of uniform convergence on *linearly compact subspaces*, i.e. subspaces $K \subseteq \Bbbk\langle E\rangle$ that are closed and have a finitary support $|K| \overset{\text{def}}{=} \cup_{x\in K}|x|$.

It appears that $\Bbbk\langle E^\infty\rangle^\perp$ is the space of *polynomials*[1]. However, thanks to the Taylor formula shown in [2], the functions in $\Bbbk\langle !E\rangle^\perp$ are *analytic*, i.e. they coincide with the limits of converging sequences of polynomials. Moreover, the topology of $\Bbbk\langle E^\infty\rangle^\perp$ is generated by the subspaces whose restrictions to polynomials of degree at most n are opens. This topology differs from the linearly compact open topology. Therefore, $\Bbbk\langle E^\infty\rangle^\perp$ is topologically different from $\Bbbk\langle !E\rangle^\perp$, which is the *completion* of the space of polynomials, endowed with the linearly compact open topology as shown in [1].

In a word, the dual of $\Bbbk\langle E^\infty\rangle$ gives rise to a simple space of computation, the polynomials. Its topology is related to the local information given at each degree. On the contrary, the dual of the exponential modality $\Bbbk\langle !E\rangle$ gives rise to the richer space of analytic functions, where the Taylor formula makes sense. Its topology is related to a global information which is not reduced to its finite approximations. One main open question in the future is to understand the algebraic nature of this exponential construction, as was achieved here for the coherence space and the Conway game model.

Acknowledgements. The authors would like to thank Thomas Ehrhard and Martin Hyland for enlightening discussions and feedbacks.

References

1. Ehrhard, T.: On finiteness spaces and extensional presheaves over the lawvere theory of polynomials. Journal of Pure and Applied Algebra (to appear)
2. Ehrhard, T.: Finiteness spaces. Mathematical. Structures in Comp. Sci. 15(4) (2005)

[1] An *homogeneous polynomial* of degree n is a function $P : \Bbbk\langle E\rangle \to \Bbbk$ which is associated with a symmetric n-linear form $\phi : \Bbbk\langle E\rangle \times \cdots \times \Bbbk\langle E\rangle \to \Bbbk$ which is hypocontinuous (a notion of continuity in between continuity and separate continuity) such that $P(x) = \phi(x,\ldots,x)$. A *polynomial* is then a finite linear combination of homogeneous polynomials.

3. Girard, J.-Y.: Linear logic. Theoretical Computer Science 50, 1–102 (1987)
4. Joyal, A.: Remarques sur la théorie des jeux à deux personnes. Gazette des Sciences Mathématiques du Québec 1(4), 46–52 (1977); English version by Robin Houston
5. Kelly, M., Laplaza, M.: Coherence for compact closed categories. Journal of Pure and Applied Algebra 19, 193–213 (1980)
6. Lafont, Y.: Logique, catégories et machines. Thèse de doctorat, Université de Paris 7, Denis Diderot (1988)
7. Lefschetz, S., et al.: Algebraic topology. American Mathematical Society (1942)
8. Melliès, P.A., Tabareau, N.: Free models of T-algebraic theories computed as Kan extensions. Submitted to Journal of Pure and Applied Algebra
9. Melliès, P.A.: Asynchronous Games 3: An Innocent Model of Linear Logic. Electronic Notes in Theoretical Computer Science 122, 171–192 (2005)
10. Seely, R.: Linear logic, $*$-autonomous categories and cofree coalgebras. In: Applications of categories in logic and computer science, vol. (92), Contemporary Mathematics (1989)

Decidability of the Guarded Fragment with the Transitive Closure*

Jakub Michaliszyn**

Institute of Computer Science,
University of Wroclaw,
ul. Joliot-Curie 15, 50-383 Wroclaw, Poland

Abstract. We consider an extension of the guarded fragment in which one can guard quantifiers using the transitive closure of some binary relations. The obtained logic captures the guarded fragment with transitive guards, and in fact extends its expressive power non-trivially, preserving the complexity: we prove that its satisfiability problem is 2Exptime-complete.

1 Introduction

The guarded fragment of first-order logic, GF, introduced in [1], is a well-known generalisation of modal logics. The main idea is to allow only a restricted form of quantification, simulating local nature of the modal operators $\Diamond$, $\Box$. GF retains a lot of nice properties of modal logics, including (a generalisation of) the tree model property, the finite modal property and the robust decidability. This makes it a promising starting point for logics for reasoning about programs and hardware.

Many extensions and variants of GF have been extensively investigated last years. One important direction is considering satisfiability of GF over restricted classes of structures, in which some distinguished binary symbols are interpreted as transitive relations (or, alternatively phrased, satisfiability of GF extended by positive statements about transitivity of some binary relations). It appeared ([7], [3]) that allowing transitive symbols to appear in arbitrary positions in formulas leads quickly to undecidability. However, if we restrict the usage of transitive symbols to guards only, the satisfiability problem becomes decidable and 2Exptime-complete ([11], [9]). The lower bound can be proved even in the presence of only two variables. This two-variable version, $[\mathrm{GF}^2 + \mathrm{TG}]$, captures (and non-trivially extends) modal logics K4, S4 and S5.

Transitivity of some binary relations is a desirable property in many reasoning tasks. However, to reason about programs it would be nice to have some way of expressing recursion. One idea is to extend GF by least and greatest fixed point operators. It was done in [5]. This way we obtain a powerful logic embedding

[*] Part of the M.Sc. thesis, under the supervision of Emanuel Kieroński.

[**] Partially supported by Polish Ministry of Science and Higher Education research project N206 022 31/3660, 2006/2009.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 261–272, 2009.

modal μ-calculus with backward modalities. Another idea is to add to GF some form of the transitive closure operator. In this paper we consider [GF^2+$^+$], an extension of the two-variable guarded fragment, in which the transitive closure operator can be applied to some atomic formulas. We note that augmenting fragments of first-order logic even by a week form of the transitive closure leads quickly to undecidability (see, e. g., [6]). Thus we have to be careful: analogously to the variant with transitive relations we restrict the usage of the transitive closure operator $^+$ to symbols appearing only in guards. More formally, the signature has a distiguished subset Σ_+, containing binary symbols, which may be used only in guards, either individually or under the transitive closure operator. We note that a variant, in which symbols from Σ_+ are used additionally outside guards, but not under the transitive closure, is undecidable.

A tempting idea is to go towards an extension of GF which would be strong enough to embed propositional dynamic logic, PDL [2]. However, this is not easy. For example, a variant allowing generalised guards of the form $T \odot S(x, y)$, simulating the composition of actions, is undecidable, even if the symbols which can be used in compositions are allowed only in guards, and there is no transitive closure operator [8].

[GF^2+$^+$] easily simulates [GF^2 + TG]: in a formula of [GF^2 + TG] instead of a transitive relation T we can simply use a transitive closure of a relation T'. However [GF^2+$^+$] is strictly more expressive than [GF^2 + TG]. For example, consider the formula $\exists x(S(x) \wedge \exists y(xT^+y \wedge R(y)))$, stating that from some point in S there exists a T-path to a point in R. This is clearly not a first-order property and thus it cannot be expressed in [GF^2 + TG].

We prove that the satisfiability problem for [GF^2+$^+$] is decidable in 2Exptime, exactly as [GF^2 + TG]. Similarly to the case of [GF^2 + TG] the proof is based on a tree-like model property. However, there are some serious complications, due to the fact that [GF^2+$^+$] can speak both about direct successors and about reachable elements. For example, in contrast to [GF^2 + TG], for a given element its witnesses cannot always be its direct successors.

The paper is organised as follows. In Section 2 we give some basic definitions and introduce a normal form of formulas. In Section 3 we prove a useful result on the two-variable logic FO^2, which will be an important tool in our proof. In Section 4 we show that every satisfiable [GF^2+$^+$] formula in normal form has a model of a special, tree-like shape. In Section 5 we outline an alternating algorithm checking the existence of such a special model for a given normal form sentence.

2 Preliminary

2.1 Logics

We work on First Order Logic (FO) with purely relational signatures, containing no constants and functional symbols. Let GF stand for the Guarded Fragment of First Order Logic defined as follows.

- every atomic formula belongs to the language of GF
- GF is closed under boolean operators
- if ψ belongs to the language of GF, $\boldsymbol{x}, \boldsymbol{y}$ are vectors of variables and $\alpha(\boldsymbol{x}, \boldsymbol{y})$ is an atomic formula that contains all variables from $\boldsymbol{x}$ and $\boldsymbol{y}$, then formulas $\forall \boldsymbol{x}.\alpha(\boldsymbol{x}, \boldsymbol{y}) \Rightarrow \psi(\boldsymbol{x}, \boldsymbol{y})$ and $\exists \boldsymbol{y}.\alpha(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y})$ belong to the language of GF

Formulas $\alpha(\boldsymbol{x}, \boldsymbol{y})$ are called *guards*. Note that $x = x$ is a special case of a guard.

Logic with the Transitive Closure In Guards ($[\mathrm{GF}+^{+}]$) is an extension of GF in which some generalised guards are allowed. We divide the signature Σ into three disjoint parts $\Sigma = \Sigma_U \cup \Sigma_B \cup \Sigma_+$. Σ_+ and Σ_B are sets of binary symbols, where symbols from Σ_+ cannot appear outside guards, and Σ_U is a set of symbols with arity different than 2. For a given symbol $T \in \Sigma_+$ we can form a guard in a usual way, e. g. xTy, or by adding operator $^{+}$ to T, e. g. $yT^{+}x$. The semantics of the operator $^{+}$ is defined as usual: T^{+} denotes the transitive closure of T.

We work with two-variable variants of the logics only. We denote them by FO^2, GF^2, $[\mathrm{GF}^2+^{+}]$. Without loss of generality we assume that signatures contain only unary and binary symbols.

2.2 Terminology

1-types and 2-types of elements in a structure over a signature Σ are defined in a standard way. The (atomic) 1-type of an element v is the set of atomic formulas with free variable x satisfied by v. Similarly, the 2-type of a pair of elements v, w is the set of atomic formulas with free variable x, y satisfied by v, w. We assume that 2-types are proper, i. e. contain $x \neq y$. For a 1-type t, let $t[x/y]$ stand for the set that contains the formulas from t in which each occurrence of x is replaced by y. For a 2-type t let $t|_{\mathcal{B}}$, which is called a restriction of t to family $\mathcal{B}$, stand for the set that contains exactly those atomic formulas from t that either have only one free variable or their relation symbol belongs to B. Similarly, $t_B = t|_{\{B\}}$ is called a restriction of t to the relation B.

To simplify the notation, we introduce for a binary symbol R a auxiliary symbol R^{-1}, whose intended meaning is to denote the inverse relation of R. Let, for a given set of binary relations or binary relation symbols $\mathcal{T}$, $\mathcal{T}^{-1}$ be the set $\{R^{-1} | R \in \mathcal{T}\}$.

For a given logic, the satisfiability problem of this logic is defined as follows: for a given formula ψ without free variables, is there any structure $\mathfrak{M}$ such that $\mathfrak{M} \models \psi$ (model $\mathfrak{M}$ satisfies ψ)?

2.3 Normal Forms

Definition 1. *We say that a formula $\psi \in \mathrm{FO}^2$ is in Scott normal form[10] if it is a conjunction of formulas in the following forms:*

(i) $\exists x.\rho(x)$
(ii) $\forall xy.\delta(x, y)$
(iii) $\forall x.\exists y.\delta(x, y)$

where both $\rho(x)$ and $\delta(x, y)$ are quantifier-free.

Every formula ψ of FO^2 can be transformed, in polynomial time, to a formula φ in Scott normal form over am extended signature, such that ψ is satisfiable if and only if φ is satisfiable. Furthermore, ψ has models of the same size as φ.

Definition 2. *We say that formula $\psi \in [\mathrm{GF}^2{+}^+]$ is in normal form if it is a conjunction of formulas in following forms:*

(i') $\exists x.\alpha(x) \wedge \rho(x)$
(ii') $\forall xy.\beta(x,y) \Rightarrow \delta(x,y)$
(iii') $\forall x.\alpha(x) \Rightarrow \exists y.\beta(x,y) \wedge \delta(x,y)$

where both $\alpha(x)$ and $\beta(x,y)$ are proper guards and neither $\rho(x)$ nor $\delta(x,y)$ contains quantifiers.

Lemma 1. *Every formula φ of $[\mathrm{GF}^2{+}^+]$ can be effectively transformed to a set of formulas Δ of $[\mathrm{GF}^2{+}^+]$ over extended signature in normal form such that*

- *φ is satisfiable if and only if $\bigvee \Delta$ is satisfiable*
- *$|\Delta| = O(2^{|\varphi|})$, $\Sigma' = O(|\varphi|)$ and for each $\psi \in \Delta$ we have $|\psi| = O(|\varphi| \log |\varphi|)$*
- *Δ can be computed in exponential time, and every $\psi \in \Delta$ can be computed in polynomial time.*

The proof of this lemma is identical to the proof of Lemma 3.2 from paper [11] about normal form for $[\mathrm{GF}^{\mathrm{k}} + \mathrm{TG}]$, because that proof does not depend on guards. Additionally we assume that a conjunct in form (i') appears exactly once in the whole formula.

We say that w is a *witness* for an element v, if for some conjunct of the form (iii) the formula $\delta(v,w)$ is satisfied or for some conjunct in the form (iii') formulas $\alpha(v)$ and $\beta(v,w) \wedge \delta(v,w)$ are satisfied.

3 Exponential Model Property for Strongly-Connected FO^2

In this section we show a lemma about FO^2, which in fact is an extension of the exponential model property [4]. This lemma will then become a crucial tool in our construction.

We say that a structure $\mathfrak{M}$ is T*-strongly-connected* if the digraph obtained from $\mathfrak{M}$ by removing all edges except T is strongly-connected. Note that a structure is T-strongly-connected if and only if the transitive closure of T contains every pair of different elements of this structure.

Lemma 2. *Let φ be a sentence from FO^2 in Scott normal form over signature Σ, with a distinguished binary symbol T, and let $\mathfrak{M}$ be a T-strongly-connected model of φ. Then φ has a T-strongly-connected model $\mathfrak{M}'$ of size bounded by $2^{4|\Sigma|+5}$, such that*

- *$\mathfrak{M}'$ contains all 1-types realized in $\mathfrak{M}$*

- *for every point v from $\mathfrak{M}'$ of 1-type t_v and every 1-type t_w such that there is a 2-type t in $\mathfrak{M}$ that contains t_v, $t_w[x/y]$ there is a point w from $\mathfrak{M}'$ such that the pair v, w has the 2-type t.*

Proof. Let $\Sigma_U = \{U_1, U_2, \ldots, U_u\}$ be a set of unary relation symbols, $\Sigma_B = \{T$, B_1, B_2, $\ldots$, $B_b\}$ be a set of binary relation symbols and $\Sigma = \Sigma_B \cup \Sigma_U$. Let us fix a sentence from FO^2 in Scott normal form over a signature Σ

$$\varphi = \exists x \rho(x) \wedge \forall xy \phi(x,y) \wedge \bigwedge_{i=1}^{k} \forall x \exists y \psi_i(x,y)$$

such that $\mathfrak{M}$ is a T-strongly-connected model of φ with universe M.

We will now build a T-strongly-connected model $\mathfrak{M}'$ with universe M', where $|M'| \leq 2^{4|\Sigma|+5}$. This construction can be seen as an extension of the construction for the exponential model for FO^2 [4]. However we have to work much harder to preserve strong-connectivity of the model.

Definitions 1.
- *Kings* are these points which have a unique 1-type in the model.
- An R-path $v_1, v_2, \ldots, v_k$ is a substructure generated by pairwise different elements $v_1, v_2, \ldots, v_k$ such that for each $i < k$ we have $\mathfrak{M} \models v_i R v_{i+1}$.
- A path $s, v_1, \ldots, v_n, s'$ is *non-royal*, if none of the elements $v_1, \ldots, v_n$ is a king.
- A *shortcut* of a path $s, \ldots, v_{pi}, v_i, \ldots, v_j, v_{nj}, \ldots, s'$, where vertices v_i and v_j have the same 1-types, is the path $s, \ldots, v_{pi}, v_i, v_{nj}, \ldots, s'$, where v_i and v_{nj} are connected in the same way as v_j with v_{nj}.
- We say that a path s is a *compression* of a path s' if s has no shortcut and there exists a sequence $r_1, r_2, \ldots, r_n$ where $r_1 = s'$, $r_n = s$ and r_{i+1} is a shortcut of r_i for $1 \leq i \leq n$.

Note that if the signature is finite, then every path can be compressed to a path of length bounded exponentially in the size of the signature.

The universe M' contains the following parts: the royal palace V_k, the court V_d and three cities V_1, V_2 and V_3. Their construction proceeds as follows:

1. We insert into the royal palace V_k copies (i. e. elements of the same 1-type) of all kings from $\mathfrak{M}$. If none of the kings satisfies ρ, then we add to the royal palace one copy of a point that satisfies ρ in $\mathfrak{M}$. We preserve the connections between these elements from $\mathfrak{M}$.
2. The court contains all witnesses for kings. More precisely, for each royal 1-type t_r, non-royal type t_n and 2-type t that contains t_r and $t_n[x/y]$, and appears in $\mathfrak{M}$, we insert into the court V_d a new point of 1-type t_n and connect it with the point of type t_r from royal palace as in t.
3. We build three cities. First, for each city we add $2^{2|\Sigma|}$ copies of every non-royal point from $\mathfrak{M}$.
4. For each point v' from the court or one of the cities, we find in $\mathfrak{M}$ a T-path s_1 from a point that has the same type as v' to the point k_1 from the royal palace, and a path s_2 from the point k_2 from the royal palace to a point that has the same type as v', where both s_1 and s_2, do not contain kings except

k_1 and k_2, respectively. We add the compression of the paths s_1 and s_2 to the V_1, if v was from the court, or to the city of v', if v' was from a city, by replacing extreme points with v' and the copies of k_1 and k_2. Similarly, for each pair (v', w') from the royal palace which has not been yet connected by a T-path, we find in $\mathfrak{M}$ a non-royal T-path s from the copy o v' to the copy of w' and add a copy of a compression of s to the court. Finally, for each newly-added element u' that is connected directly (i.e. by some relation, not only by the transitive closure of some relation) with a king of 1-type t_k, we find in $\mathfrak{M}$ an element u which has the same 1-type as u' and is connected with the king of the 1-type t_k in the same way as u'. We set connections between u' and V_K as between u and all kings from $\mathfrak{M}$.

5. We ensure that all non-royal points have non-royal witnesses. For each non-royal point v' from $\mathfrak{M}'$ of 1-type t_s, each non-royal 1-type t_r and each 2-type t from $\mathfrak{M}$ which contains t_s and $t_r[x/y]$, we copy connections from t to $\mathfrak{M}'$ in the following way. If v' was in V_d or V_3, then we connect it with points from V_1, if it was in V_1 then with V_2, and if it was from V_2 then with V_3. Each city contains $2^{2|\Sigma|}$ elements of 1-type t_r, so for each 2-type we can choose a different element.
6. We ensure that all non-royal points have the requested witnesses among kings: for each non-royal element w' from $\mathfrak{M}'$, if a connection between w' and the royal palace is not set yet, then we find in $\mathfrak{M}$ an element of the same 1-type and copy connections between this point and kings to $\mathfrak{M}'$. Note that we always can find such a pair, because all points in $\mathfrak{M}'$ are copied from $\mathfrak{M}'$ and, moreover, if some 1-type appears in $\mathfrak{M}$ only once, then it appears also only once in V_K and it does not appear in the court or in any city.
7. For each pair v', w' of points from $\mathfrak{M}'$, if the connection between v' and w' is not already set, we copy some connection beetwen points of the same 1-types from $\mathfrak{M}$. Again, a proper connection can be found because of special treatment of kings (as in step (6)).

Note that $|V_k| \leq 2^{|\Sigma|}$, $|V_d| \leq 2^{2|\Sigma|}|V_k| + 2^{|\Sigma|}|V_k|^2$ and $|V_i| \leq (1 + 2^{|\Sigma|}) \cdot 2^{2|\Sigma|} + 2^{|\Sigma|}|V_d|$ for $i \in \{1, 2, 3\}$, so $|M| \leq 2^{4|\Sigma|+5}$. Let us observe that:

- The formula $\exists x \rho(x)$ is satisfied in $\mathfrak{M}'$, because a point that satisfies this formula was added in step 1.
- All 1-types and 2-types from $\mathfrak{M}'$ appear also in $\mathfrak{M}$; thus, ϕ is satisfied.
- Every point has all witnesses it needs, so each ψ_j is satisfied in $\mathfrak{M}'$.
- The royal palace is a T-strongly-connected subgraph (because of step 4) and moreover each courtier and citizen is on a T-path from the royal palace to the royal palace because of paths added in step 4.

Therefore, $\mathfrak{M}'$ satisfies φ and is T-strongly-connected. The construction also implies that in $\mathfrak{M}'$ there appear all 1-types from $\mathfrak{M}$ are realized. □

Theorem 1. *Let φ be a an FO^2 formula over a signature Σ with a distinguished symbol T, and let $\mathfrak{M}$ be a T-strongly-connected model of this formula. Then φ has a T-strongly-connected model of size exponential in $|\Sigma|$.*

This Theorem is a straightforward consequence of Lemma 2 and the observation about Scott normal form.

4 Ramified Model Property for $[\mathrm{GF}^2{+}^+]$

The proof of a special model property of $[\mathrm{GF}^2{+}^+]$ has the common outline with the idea presented in [11] for $[\mathrm{GF}^2 + \mathrm{TG}]$. In both proofs we at first prove that each satisfiable formula has a model of a tree-like shape. This is done in the following way:

1. Take a satisfiable formula in normal form and a model $\mathfrak{M}$ of this formula.
2. Take one point from the model $\mathfrak{M}$ and start building the new model $\mathfrak{M}'$ from this point.
3. If this point is in some clique whose edges are defined by a one of transitive relation (a transitive closure of a relation from Σ_+), take this clique, compress it and add it to $\mathfrak{M}'$.
4. If the current point needs a witness outside its cliques, find a path to this witness in $\mathfrak{M}$, compress it and add it $\mathfrak{M}'$.
5. Process recursively the newly-added points (as in points (3) - (5)).

The most important differences between the proofs are in points (3) and (4). The reason why the part (3) of the proof is more difficult in the case of $[\mathrm{GF}^2{+}^+]$ is that the connections defining cliques are not atomic relation as in $[\mathrm{GF}^2 + \mathrm{TG}]$, but paths. To overcome this difficulty we use the result of Section 3. The part (4) is more complicated because in $[\mathrm{GF}^2{+}^+]$ some witnesses cannot allways be direct successors of an element.

It is important to underline why we take care about cliques. Our point is to construct a model which looks like a tree. However, in $[\mathrm{GF}^2{+}^+]$ logic we can write a sentence ψ which is a conjunction of following formulas:

- $\forall x(S_0(x) \Rightarrow \forall y(xR^+y \Rightarrow S_0(y) \Rightarrow x = y))$ (if a point v satisfies S_0, then there is no point reachable from v by relation R that satisfies S_0, except, possibly, v)
- every point satisfies exactly one of the relations $S_0, \ldots, S_{n-1}$
- there exists a point that satisfies S_0
- $\forall x.S_i(x) \Rightarrow \exists y(xRy \wedge S_{(i+1) \bmod n}(y))$ for each $i < n$

It is easy to see that every model of ψ contains a cycle of length n. In fact, we can obtain a cycle of length 2^n, using $S_0, \ldots, S_{n-1}$ to encode a binary number and request that every successor of a point v encodes the value greater by 1 modulo 2^n. As we see, the structure must sometimes have fragments that do not look like a tree. We will see that the size of each such fragment can be bounded exponentially.

4.1 Construction of a Ramified Model for $[\mathrm{GF}^2{+}^+]$

Let $\varphi = \exists x\rho(x) \wedge \bigwedge_{i=1}^{j} \forall xy\delta_i(x,y) \wedge \bigwedge_{i=1}^{k} \forall x.\alpha_i(x) \Rightarrow \exists y.\psi_i(x,y)$ be a fixed formula in normal form from $[\mathrm{GF}^2{+}^+]$ over the signature $\Sigma = \Sigma_U \cup \Sigma_B \cup \Sigma_+$, where $\Sigma_+ = \{T_1, \ldots, T_n\}$.

First, we define some properties of connections:

Definitions 2. – Let t be a 2-type. We say that t is *k-positive* if there are exactly k different relations $R \in \Sigma_+$ such that t satisfies $xRy \vee yRx$.

- We say that a structure is *1-positive*, if every 2-type that appears in this structure is either 1-positive or 0-positive.
- An *extended 1-type* of a point v in a structure $\mathfrak{A}$ is the set that contains the 1-type of v and the pairs $\langle R, t\rangle$, where $R \in \Sigma_+ \cup \Sigma_+^{-1}$ and for some $w \neq v$ of 1-type t formula vR^+w is satisfied in a structure $\mathfrak{A}$.

Note that a restriction of a 2-type t to a binary relation B, $t|_B$, is at most 1-positive. Now we define cliques and some operations on them.

Definitions 3. – An R^+-clique in a structure $\mathfrak{M}$ is a subset K of elements from $\mathfrak{M}$, such that for each $v, w \in K$ we have $\mathfrak{M} \models vR^+w$.

- The maximal R^+-clique that contains v is denoted by R^+*-clique*(v).
- A *path of R^+-cliques* $\langle C_1, v_1^{in}, v_1^{out}\rangle$, $\langle C_2, v_2^{in}, v_2^{out}\rangle$, $\ldots$, $\langle C_k$, v_k^{in}, $v_k^{out}\rangle$ is a substructure generated by pairwise disjoint cliques $C_1, C_2, \ldots, C_k$, where for each $1 \leq i \leq k$ the clique C_i is the maximal R^+-clique containing distinguished vertices v_i^{in} and v_i^{out}, and for $i < k$, $M \models v_i^{out}Rv_{i+1}^{in}$.
- We say that a vertex v is in the clique-distance m from w if the shortest (i. e. of the minimal length) path of R^+-cliques from R^+-clique(v) to R^+-clique(w) has the length m.
- A path of R^+-cliques $\langle C_1, v_1^{in}, v_1^{out}\rangle, \ldots, \langle C_z, v_z^{in}, v_z^{out}\rangle$, $\langle C_d, v_d^{in}, v_d^{out}\rangle, \ldots$ is a *shortcut* of a path of cliques $\langle C_1$, v_1^{in}, $v_1^{out}\rangle$, $\ldots$, $\langle C_z$, v_z^{in}, $v_z^{out}\rangle$, $\langle C_p$, v_p^{in}, $v_p^{out}\rangle$, $\ldots$, $\langle C_d, v_d^{in}$, $v_d^{out}\rangle$, $\ldots$ if v_p^{in} and v_d^{in} has the same 1-types.
- A *compression of a path of R^+-cliques s* is a minimal path of R^+-cliques obtained by an iterated shortcutting of s.

Note that for any $R \in \Sigma_+$ a single vertex is a R^+-clique.

Now we define an operation on a structure that is usefull to express its tree-likeness. Intuitively, a flattening of a model $\mathfrak{M}$ is a graph $G = \langle V, E\rangle$, such that V contains one vertex for each clique from $\mathfrak{M}$, and E connects cliques C_1 and C_2 if at least one of the following conditions holds:

- C_1 has a common vertex with C_2 in $\mathfrak{M}$
- C_1 is connected in $\mathfrak{M}$ with C_2 by some relation from Σ_+
- C_1 is connected in $\mathfrak{M}$ with C_2 by some relation from Σ_B and is not connected by the transitive closure of any of relation from Σ_+.

The formal definition is more complicated. Each vertex is in n cliques (one for each relation from Σ_+), so connecting cliques with a common vertex lead us to cliques of size n. We want to show that some flattening are trees, so we arbitrary choose relation T_1 and connect cliques with common vertex only if one of this cliques is T_1^+-clique.

Definition 3. *We say that an undirected graph $G = \langle V, E\rangle$ is a flattening of a model $\mathfrak{M}$, if*

- $V = \{R^+\text{-}clique(w) | R \in \Sigma_+ \wedge w \in M\}$

– *$\{T_i^+$-clique(w), T_j^+-clique$(w')\}$ is in E if T_i^+-clique$(w) \neq T_j^+$-clique(w') and at least one of the following conditions holds:*
  - *$w = w'$ and $i = 1$*
  - *$i = j$, and $\mathfrak{M} \models wT_iw'$ holds*
  - *$i = j = 1$, for some $S \in \Sigma_B \cup \Sigma_B^{-1}$ we have $\mathfrak{M} \models wSw'$ and for each $S \in \Sigma_+ \cup \Sigma_+^{-1}$ condition $\mathfrak{M} \models wS^+w'$ is not satisfied*

We are ready to define a property of models that will be used to build an algorithm that checks if a given formula from $[\mathrm{GF}^2{+}^+]$ has any model.

Definition 4. *We say that a model $\mathfrak{M}$ of a formula φ is r-ramified, if $\mathfrak{M}$ is 1-positive, the size of each R^+-clique in $\mathfrak{M}$ for $R \in \Sigma_+$ is bounded by r and the flattening of $\mathfrak{M}$ is a tree.*

Theorem 2. *Every satisfiable sentence from $[\mathrm{GF}^2{+}^+]$ over a signature Σ has a r-ramified model for $r = 2^{4|\Sigma|+5}$, in which every point has all the required witnesses in clique-distance not greater than $2^{|\Sigma|}$.*

Proof. Let $\varphi = \exists x \rho(x) \wedge \bigwedge_{i=1}^{j} \forall xy \delta_i(x,y) \wedge \bigwedge_{i=1}^{k} \forall x.\alpha_i(x) \Rightarrow \exists y.\psi_i(x,y)$ be a fixed formula from $[\mathrm{GF}^2{+}^+]$ in normal form over the signature $\Sigma = \Sigma_U \cup \Sigma_B \cup \Sigma_+$, where $\Sigma_+ = \{T_1, \ldots, T_n\}$. Furthermore, let $\mathfrak{M}$ be a model of φ. Now we define a recursive procedure that, for a given $2^{4|\Sigma|+5}$-ramified structure $\mathfrak{M}'$, point $v \in \mathfrak{M}'$ and function $from : \mathfrak{M}' \to \mathfrak{M}$, extends $\mathfrak{M}'$ to a $2^{4|\Sigma|+5}$-ramified structure where v and every newly-added vertex have all needed witnesses. Simultaneously, it extends the function $from$. Intuitively, $from(v)$ indicates a point "similar" to w.

1. Build the cliques of v, modifying the cliques of $from(v)$, proceed as follows. For each relation $R \in \Sigma_+$ if v is not inside an R^+-clique with size greater then 1, then we take from $\mathfrak{M}$ the R^+-clique($from(v)$) and we restrict all 2-types in this clique to the family of relations $\{R\} \cup \Sigma_B$. Such a structure is an R-strongly-connected component. If this component has more then 1 vertex, then, using Lemma 2 for formula $(\exists x \top) \wedge \bigwedge_{i=1}^{j} \forall xy {\delta'}_i(x,y)$ and this component, where ${\delta'}_i$ is obtained from δ_i by replacing all guards xR^+y by $\top$, we transform this component to a structure H' with an exponential size, such that in H' all witnesses are preserved. Then we choose from H' an element w with the same 1-type as v and add H' to $\mathfrak{M}'$ by identifying v with w. Finally, for each element $u' \neq w$ from H' we find in H an element u with the same 1-type and we set $from(u') = u$.
2. We ensure that v has required witnesses outside its cliques. In order to do that, for each formula $\psi_i = \gamma_i \wedge \delta_i$, where γ_i is a guard, vertex v satisfies α_i and ψ_i is not satisfied in $\mathfrak{M}'$ yet, we choose from $\mathfrak{M}$ a point w' that is a witness of this formula for v'. Observe that w' is not in any clique with v'.
   (a) If $\gamma_i = xRy$ or $\gamma_i = yRx$ for some $R \in \Sigma_B \cup \Sigma_+$, then we add a copy w of a point w' to the model $\mathfrak{M}'$ and we set connections between v and w in the same way as it was in $\mathfrak{M}$ after restriction to R. Moreover, we put $from(w) = w'$.

(b) If $\gamma_i = xR^+y$ or $\gamma_i = yR^+x$ for some $R \in \Sigma_+$, then we choose from $\mathfrak{M}$ a full path that provides the fulfilment of γ_i, with all R^+-cliques that appear on this path. We do it in following way.
Assume that $\gamma_i = xR^+y$. We take a path of R^+-cliques from R^+-clique(v') to R^+-clique(w') with minimal length, we restrict every 2-types on this path to R, and then we compress this path, obtaining a path $\langle R^+\text{-clique}(v'), v', v'^{out}\rangle$, $\langle C_1, v_1^{in}, v_1^{out}\rangle$, $\dots$, $\langle C_k, v_k^{in}, v_k^{out}\rangle$, where $C_k = R^+$-clique(w) and $v_k^{out} = w'$.
We compress every clique C_i to C'_i in the way presented in step 1, obtaining cliques in which points v'^{in}_i and v'^{out}_i have the same 1-types as v_i^{in} and v_i^{out}, respectively. Then, in the R^+-clique(v), we find a vertex v^{out} with the same 1-type as v'^{out}. We add to $\mathfrak{M}'$ path $\langle C'_1, v'^{in}_1, v'^{out}_1\rangle$, $\dots$, $\langle C'_k, v'^{in}_k, v'^{out}_k\rangle$ and we connect v^{out} and v'^{in}_1 in the same way as v'^{out} and v'^{in}_1 were connected.
For each i we put $from(v'^{in}_i) = v_i^{in}$ and $from(v'^{out}_i) = v_i^{out}$. Moreover, for each vertex $u' \in C'_i$ such that $from(u')$ is not set yet, we find in C_i a vertex u with the same 1-type and put $from(u') = u$.
When $\gamma_i = yR^+x$, we do the same for R^{-1}.

3. We connect vertices from $\mathfrak{M}'$ by relations from Σ_B using some patterns from $\mathfrak{M}$. More precisely, for each two vertices v, w:
   (a) If for some $R \in \Sigma_+$ condition $vR^+w \wedge wR^+v$ occurs, then these vertices are in the same clique and the connections are already established.
   (b) If these vertices are connected by R^+ for some $R \in \Sigma_+$, the connection is asymmetric (without loss of generality we may assume that $vR^+w \wedge \neg wR^+v$ holds) and connections between these points were not established yet, then we find in $\mathfrak{M}$ a vertex w' which has the following property: in $\mathfrak{M}$ there is a R-path from $from(v)$ to w' and w' has the same 1-type as w. Such a vertex exists, because the 1-type of w belongs to the set of the 1-types that are reachable by relation R, written in extended 1-type of $from(v)$. We add connections from Σ_B between v and w in the same way as $from(v)$ and w' were connected.
   (c) If these two points are not connected by the transitive closure of some relation from Σ_+, then either all connections between this points were already set, or they are not connected at all – then we set empty connection between this elements.
4. We repeat steps 1-4 for all vertices added in this stage.

We take from $\mathfrak{M}$ a point v', that satisfies ρ, add its copy v to $\mathfrak{M}'$, set $from(v) = v'$ and apply the procedure above to v. The structure built by this procedure is a model of φ. The proof is omitted in this version due page limit. □

5 Algorithm

In this section we describe an alternating algorithm working in exponential space that checks if a given formula has a ramified model. From Subsection 4.1 we know that every satisfiable formula in $[\mathrm{GF}^2{+}^{+}]$ has a ramified model, so this algorithm resolves the satisfiability problem. At first, let us introduce some definitions:

- 1-types of elements are defined as above.
- full 1-type of an element contains the following information: 1-type of element, list of 1-types of all direct successors for each binary relation, and, for each relation R which appears under the transitive closure in the formula, a list of 1-types of vertices reachable by R, and the information about 1-types of vertices reachable by R^{-1}, except for vertices that are in R^{+}-clique with considered element.
- type of a clique, containing the following information: its size, full 1-types of all the vertices in the clique, information about connections between the vertices and function *promise*, which for a given 1-type t and $b \in \{-1, 1\}$ returns the length of the path of $(R^{b})^{+}$-cliques from the current clique to a clique that contains a vertex of type t (or 0, if there is no such path).

Note that every two cliques of the same type are isomorphic. For the sake of simplification, in this section we look upon a type of a clique as a clique of this type with the attached function *promise*. This function is needed because the algorithm guesses in each stage only cliques from a direct neighbourhood, while Theorem 2 guarantees only that all needed witnesses are in clique-distance not greater than exponential in the size of the signature. Due to the page limit, only a sketch of the algorithm is presented.

Step 1. For a given formula $\varphi = \exists x \rho(x) \wedge \bigwedge_{i=1}^{j} \forall xy \delta_i(x, y) \wedge \bigwedge_{i=1}^{k} \forall x.\alpha_i(x) \Rightarrow \exists y.\psi_i(x, y)$, the algorithm starts from guessing (i.e. existentially choosing) type of a starting clique K containing an element satisfying ρ. Then it checks if local properties of K are correct: if connections between vertices in the clique satisfy δ_i for each i, if all successors from the full 1-type of points in the clique can be connected with points from the clique in a way that satisfies each δ_i, and if the *promise* function and full 1-types of vertices are not inconsistent.
Step 2. The algorithm finds direct witnesses for each vertex in the clique by guessing types of the cliques containing witnesses and connections between guessed points and K. The algorithm checks if new cliques are locally proper. Then, for each R^{+}-clique K', which is connected with the previous clique by R, the algorithm checks if 1-types of every vertex from K' are included in full 1-types of vertices from K to make sure that these vertices can be connected by binary relations in a way that satisfy each δ_i. Furthermore, the algorithm checks if the sets of 1-types reachable by R from vertices in K' are subsets of the sets reachable by R from vertices in K and vice versa. Then the algorithm checks if the type of K (including the *promise* function) and guessed cliques guarantee all witnesses for each vertex from K.
Step 3. The algorithm guesses types of cliques that are on paths of cliques to some clique that contains points of 1-types guaranteed in the *promise* function from K. For the easier control of dependencies, the algorithm guess on this stage only the first clique from this path and check if the guessed clique has less value of the *promise* function or realizes this 1-type.

Step 4. The algorithm checks the counter of stages. If the value of the counter is greater then $2^{8|\Sigma|}$, then algorithm stops and return "Yes", because then we know that some type of cliques occurred twice and another computation would be the same as previously. In the other case the algorithm increments the counter, universally chooses a clique K from the set of cliques added in this stage and goes to step 2.

The algorithm needs only exponential memory, because each type of clique has at most exponential size, so, since 2EXPTIME = AEXPSPACE, the satisfiability problem can be solved in 2EXPTIME.

The lower bound follows from the 2EXPTIME-hardness of the satisfiability problem for logic $[\mathrm{GF}^2 + \mathrm{TG}]$, presented in [9], since we can simply replace transitive relations by transitive closure of these relations.

Corollary 1. *The satisfiability problem for* $[\mathrm{GF}^2{+}^{+}]$ *is* 2EXPTIME-*complete.*

References

1. Andreka, H., Nemeti, I., van Benthem, J.: Modal languages and bounded fragments of predicate logic. Journal of Philosophical Logic 27, 217–274 (1998)
2. Fischer, M.J., Ladner, R.E.: Propositional modal logic of programs. In: Proceedings of the Ninth Annual ACM Symposium on theory of Computing, STOC 1977, Boulder, Colorado, United States, May 04 - 04, 1977, pp. 286–294. ACM, New York (1977)
3. Ganzinger, H., Meyer, C., Veanes, M.: The two-variable guarded fragment with transitive relations. In: Proc. LICS 1999. IEEE Computer Soc. Press, Los Alamitos (1999)
4. Grädel, E., Kolaitis, P.G., Vardi, M.Y.: On the decision problem for two-variable first-order logic. Bulletin of Symbolic Logic 3(1), 53–69 (1997)
5. Grädel, E., Walukiewicz, I.: Guarded Fixed Point Logic. In: Proceedings of 14th IEEE Symposium on Logic in Computer Science LICS 1999, Trento (1999)
6. Grädel, E., Otto, M., Rosen, E.: Undecidability results on two-variable logics. In: Reischuk, R., Morvan, M. (eds.) STACS 1997. LNCS, vol. 1200, pp. 249–260. Springer, Heidelberg (1997)
7. Grädel, E.: On the restraining power of guards. Journal of Symbolic Logic 64(4), 1719–1742 (1999)
8. Kazakov, Y.: Saturation-Based Decision Procedures for Extensions of the Guarded Fragment. PhD thesis, Universität des Saarlandes, Saarbrücken (March 2006)
9. Kieroński, E.: The two-variable guarded fragment with transitive guards is 2EXPTIME - Hard. In: Gordon, A.D. (ed.) FOSSACS 2003. LNCS, vol. 2620, pp. 299–312. Springer, Heidelberg (2003)
10. Scott, D.: A decision method for validity of sentences in two variables. J. Symb. Logic 27, 477 (1962)
11. Szwast, W., Tendera, L.: The guarded fragment with transitive guards. Annals of Pure and Applied Logic 128, 227–276 (2004)

Weak Alternating Timed Automata

Pawel Parys[1,⋆] and Igor Walukiewicz[2,⋆⋆]

[1] Warsaw University, Poland
[2] LaBRI, CNRS and Bordeaux University, France

Abstract. Alternating timed automata on infinite words are considered. The main result is a characterization of acceptance conditions for which the emptiness problem for the automata is decidable. This result implies new decidability results for fragments of timed temporal logics. It is also shown that, unlike for MITL, the characterisation remains the same even if no punctual constraints are allowed.

1 Introduction

Timed automata [5] is a widely used model of real-time systems. It is obtained from finite automata by adding clocks that can be reset and whose values can be compared with constants. The crucial property of timed automata is that their emptiness is decidable. Alternating timed automata have been introduced in [15,20] following a sequence of results [1,2,19] indicating that a restriction to one clock can make some problems decidable. The emptiness of one clock alternating automata is decidable over finite words, but not over infinite words [23,16]. Undecidability proofs rely on the ability to express "infinitely often" properties. Our main result shows that once these kind of properties are forbidden the emptiness problem is decidable.

To say formally what are "infinitely often" properties we look at the theory of infinite sequences. We borrow from that theory the notion of an index of a language. It is known that the index hierarchy is infinite with "infinitely often" properties almost at its bottom. From this point of view, the undecidability result mentioned above left open the possibility that safety properties and "almost always" properties can be decidable. This is indeed what we prove here.

Automata theoretic approach to temporal logics [26] is by now a standard way of understanding these formalisms. For example, we know that the modal μ-calculus corresponds to all automata, and LTL to very weak alternating automata, or equivalently, to counter-free nondeterministic automata [29]. By translating a logic to automata we can clearly see combinatorial challenges posed by the formalism. We can abstract from irrelevant details, such as a choice of operators for a logic. This approach was very beneficial for the development of logical formalisms over sequences.

An automata approach has been missing in timed models for an obvious reason: no standard model of timed automata is closed under boolean operations.

⋆ Author supported by Polish government grant no. N206 008 32/0810.

⋆⋆ Author supported by project DOTS (ANR-06-SETI-003).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 273–284, 2009.

Event-clock automata [7] may be considered as an exception, but the price to pay is a restriction on the use of clocks. Alternating timed automata seem to be a good model, although the undecidability result over infinite words shows that the situation is more difficult than for sequences. Nevertheless, Ouaknine and Worrell [22] have shown decidability of the emptiness problem provided all states are accepting, and some locality restriction on the transition function holds. Using this, they have identified a decidable fragment of MTL called Safety MTL.

In this paper we show that our main result allows to get a decidable fragment of TPTL [8] with one variable, that we call Constrained TPTL. This fragment contains Safety MTL and allows all eventually formulas. Its syntax has also some similarities with another recently introduced logic: FlatMTL [11,12]. We give some elements of comparison between the logics later in the paper. In brief, the reason why Constrained TPTL is not strictly more expressive than FlatMTL is that the later includes MITL [6]. The later is a sub-logic of MTL where punctual constraints are not allowed.

The case of MITL makes it natural to ask what happens to alternating timed automata when we disallow punctual constraints. This is an interesting question also because all known undecidability proofs have used punctual constraints in an essential way. Our second main result (Theorem 4), says that the decidability frontier does not change even if we only allow to test if the value of a clock is bigger than 1. Put it differently, it is not only the lack of punctual constraints, but also very weak syntax of the logic that makes MITL decidable.

Related work. The idea of restricting to one clock automata dates back at least to [14]. Alternating timed automata where studied in a number of papers [16,23,4,3]. The most relevant result here is the decidability of the emptiness for the case when when all states are accepting and some locality condition holds [22]. One of technical contributions of the present paper is to remove the locality restriction, and to add a non-accepting layer of states on the top of the accepting one.

For a long time MITL [6] was the most prominent example of a decidable logic for real-time. In [23] Ouaknine and Worrell remark that MTL over finite words can be translated to alternating timed automata, and hence it is decidable. They also show that over infinite words the logic is undecidable (which is a stronger result than undecidability for the automata model in [16]). They have proposed a fragment of MTL, called Safety MTL. Decidability of this fragment was shown in [22] by reducing to the class of ATA mentioned in the previous paragraph. A fragment of MTL called FlatMTL [11,12] represents an interesting but technically different direction of development (cf. Sect. 4).

We should also discuss the distinction between continuous and pointwise semantics. In the later, the additional restriction is that formulas are evaluated only in positions when an action happens. So the meaning of $F_{(x=1)}\alpha$ in the continuous semantics is that in one time unit from now formula α holds, while in the pointwise semantics we additionally require that there is an action one time unit from now. Pointwise semantics is less natural if one thinks of encoding properties of monadic predicates over reals. Yet, it seems sufficient for descriptions of behaviors of devices, like timed automata, over time [24]. Here we consider the pointwise semantics simply because the emptiness of alternating timed automata in continuous semantics is undecidable even over finite words. At present, it seems that an approach through compositional methods [13] is more suitable to deal with continuous semantics.

The depth of nesting of positive and negative conditions of type "infinitely often" is reflected in the concept of the index of an automaton. Wagner [27], as early as in 1977, established the strictness of the hierarchy of indices for deterministic automata on infinite words. Weak conditions were first considered by Staiger and Wagner [28]. There are several results testifying their relevance. For example Mostowski [17] has shown a direct correspondence between the index of weak conditions and the alternation depth of weak second-order quantifiers. For recent results on weak conditions see [18] and references therein.

Organization of the paper. After a section with basic definitions we state our main decidability result (Theorem 2) and an accompanying undecidability result (Theorem 4). We give an outline of the proof of the former theorem. Sect. 4 introduces Constrained TPTL, gives a translation of the logic into a decidable class of alternating timed automata, and discusses relations with FlatMTL.

For the reasons of space, proofs are largely omitted. They can be found in the full version of the paper [25].

2 Preliminaries

A *timed word* over a finite alphabet Σ is a sequence: $w = (a_1, t_1)(a_2, t_2)\dots$ of pairs from $\Sigma \times \mathbb{R}_+$. We require that the sequence $\{t_i\}_{i=1,2,\dots}$ is strictly increasing and unbounded (non Zeno).

We will consider alternating timed automata (ATA) with one clock [16]. Let x be this clock and let Φ denote the set of all comparisons of x with constants, eg. $(x < 1 \wedge x \geq 0)$. A one-clock ATA over an alphabet Σ is a tuple

$$\mathcal{A} = \langle Q, \Sigma, q_o, \delta, \Omega : Q \to \mathbb{N} \rangle$$

where Q is a finite set of states, and Ω determines the parity acceptance condition. The transition function of the automaton δ is a finite partial function:

$$\delta : Q \times \Sigma \times \Phi \dot{\to} \mathcal{B}^+(Q \times \{\texttt{nop}, \texttt{reset}\}).$$

where $\mathcal{B}^+(Q \times \{\texttt{nop}, \texttt{reset}\})$ is the set of positive boolean formulas over atomic propositions of the form $\top$, $\bot$, and (q, f) with $q \in Q$ and $f \in \{\texttt{nop}, \texttt{reset}\}$.

Intuitively, automaton being in a state q, reading a letter a and having a clock valuation satisfying θ can proceed according to the positive boolean formula $\delta(q, a, \theta)$. It means that if a formula is a disjunction then it chooses one of the disjuncts to follow, if it is a conjunction then it makes two copies of itself each following one conjunct. If a formula is "atomic" of the form $(q, \texttt{nop})$ or $(q, \texttt{reset})$ then the automaton changes the state to q, and either does nothing or sets the value of the clock to 0, respectively. Formula $\top$ is unconditionally accepting, and $\bot$ unconditinally rejecting.

To simplify the definition of acceptance there is also one more restriction on the transition function:

> *(Partition)* For every $q \in Q$, $a \in \Sigma$ and $v \in \mathbb{R}_+$, there is exaclty one θ s.t. $\delta(q, a, \theta)$ is defined, and v satisfies θ.

The *acceptance condition* of the automaton determines which infinite sequences of states (runs of the automaton) are accepting. A sequence $q_1 q_2 \dots$ satisfies:

- *weak parity condition* if $\min\{\Omega(q_i) : i = 1, 2, \dots\}$ is even,
- *strong parity condition* if $\liminf_{i=1,2,\dots} \Omega(q_i)$ is even.

Observe that the difference between weak and strong condition is that in the weak case we consider all occurrences of states and in the strong case only those that occur infinitely often. We will mostly use automata with weak conditions. Whenever we will be considering strong conditions we will say it explicitly.

An alternating timed automaton $\mathcal{A}$ and a timed word $w = (a_1, t_1)(a_2, t_2) \dots$ determine the *acceptance game* $G_{\mathcal{A},w}$ between two players: Adam and Eve. Intuitively, the objective of Eve is to accept w, while the aim of Adam is the opposite. A play starts at the initial configuration $(q_0, 0)$. It consists of potentially infinitely many phases. The $(k+1)$-th phase starts in (q_k, v_k), ends in some configuration (q_{k+1}, v_{k+1}) and proceeds as follows. Let $v' := v_k + t_{k+1} - t_k$. Let θ be the unique (by the partition condition) constraint such that v' satisfies θ and $b = \delta(q_k, a_{k+1}, \theta)$ is defined. Now the outcome of the phase is determined by the formula b. There are three cases:

- $b = b_1 \wedge b_2$: Adam chooses one of subformulas b_1, b_2 and the play continues with b replaced by the chosen subformula;
- $b = b_1 \vee b_2$: dually, Eve chooses one of subformulas;
- $b = (q, f) \in Q \times \{\mathtt{nop}, \mathtt{reset}\}$: the phase ends with the result $(q_{k+1}, v_{k+1}) := (q, f(v'))$. A new phase is starting from this configuration.
- $b = \top, \bot$ the play ends.

The winner is Eve if the sequence ends in $\top$, or it is infinite and the states appearing in the sequence satisfy the acceptance condition of the automaton.

Formally, a *partial play* is a finite sequence of consecutive game positions of the form $\langle k, q, v \rangle$ or $\langle k, q, v, b \rangle$ where k is the phase number, b a boolean formula, q a location and v a valuation. A *strategy* of Eve is a mapping that assigns to each such sequence ending in Eve's position a next move of Eve. A strategy is winning if Eve wins whenever she applies this strategy.

Definition 1 (Acceptance). *An automaton $\mathcal{A}$* accepts *w iff Eve has a winning strategy in the game $G_{\mathcal{A},w}$. By $L(\mathcal{A})$ we denote the language of all timed words w accepted by $\mathcal{A}$.*

The *Mostowski index* of an automaton with the, strong or weak, acceptance condition given by Ω is the pair consisting of the minimal and the maximal value of Ω: $(\min(\Omega(Q)), \max(\Omega(Q)))$. We may assume without a loss of generality that $\min(\Omega(Q)) \in \{0,1\}$. (Otherwise we can scale down the rank by $\Omega(q) := \Omega(q) - 2$.). Automata with strong conditions of index $(0,1)$ are traditionally called Büchi automata and their acceptance condition is given by a set of accepting states $Q_+ \subseteq Q$; in our presentation theses are states with rank 0.

3 Decidability for One-Clock Timed Automata

We are interested in the emptiness problem for one clock ATA. As mentioned in the introduction, the problem is undecidable for automata with strong Büchi conditions. Here we will show a decidability result for automata with weak acceptance conditions of index $(0,1)$. A different presentation of these automata is that they are strong Büchi automata where there are no transitions from an accepting state to a non-accepting state. Indeed, once the automaton sees a state of priority 0 then any infinite run is accepting (but there may be runs that get blocked). In the following we will write Q_+ for accepting states, and Q_- for the other states. For automata presented in this way the (strong or weak) Büchi acceptance condition says simply: there are only finitely many states from Q_-. So the automaton accepts if Eve has a strategy to reach $\top$, or to satisfy this condition.

Theorem 2. *For one-clock Büchi alternating timed automata with no transitions from states in Q_+ to states in Q_-: it is decidable whether a given automaton accepts a non Zeno timed word.*

Ouaknine and Worrell [21] have proved undecidability of MTL over infinite timed words. Their construction immediately implies undecidability for weak automata with $(1,2)$ condition. So the above decidability result is optimal with respect to index of the accepting condition.

Theorem 3 (Ouaknine, Worell). *It is undecidable whether a given one-clock Büchi nondeterministic timed automaton $\mathcal{A}$ accepts every infinite word, even when there are no transitions in $\mathcal{A}$ from states in Q_- to states in Q_+.*

The construction in op. cit. relies on equality constraints. Indeed, if we do not allow equality constraints in MTL then we get a fragment called MITL, and satisfiability problem for MITL over infinite words is decidable [6]. We show that this phenomenon does not appear in the context of automata.

Theorem 4. *It is undecidable whether a given one-clock Büchi alternating timed automaton $\mathcal{A}$ accepts an infinite word, even when there are no transitions in $\mathcal{A}$ from states in Q_- to states in Q_+, and when $\mathcal{A}$ does not test for equality.*

In the rest of the section we give an outline of the proof of Theorem 2. Due to space restrictions it is not possible to present the proof of Theorem 4. The proof given in the full version of the paper [25] shows undecidability even when one uses only tests: $(x \geq 1)$, and its negation.

To fix the notation we take a one clock ATA:

$$\mathcal{A} = \langle Q, \Sigma, q_o, \delta, Q_+ \subseteq Q \rangle.$$

We will assume that the transition function satisfies the partition condition. For simplicity, we also assume that every value of δ is a boolean formula in a disjunctive normal form. Moreover, we will require that in every disjunct of every transition of $\mathcal{A}$ there is some pair with `reset` and some pair with `nop`. Every automaton can be easily put into this form.

Our first step will be to construct some infinite transition system $\mathcal{H}(\mathcal{A})$, so that the existence of an accepting run of $\mathcal{A}$ is equivalent to the existence of some good path in $\mathcal{H}(\mathcal{A})$. In the second step we will use some structural properties of this transition system to show decidability of the problem.

3.1 Construction of $\mathcal{H}(\mathcal{A})$

This is surely the less interesting part of the proof, unfortunately we need to spend some time here as $\mathcal{H}(\mathcal{A})$ is the object we work with in the second step.

We need to reformulate the definition of acceptance of $\mathcal{A}$ in terms of a sequence instead of a tree. Additionally, we would like to abstract from time values in states and transitions, but still be able to tell if a run is non Zeno.

We start with the definition of regions. As we have only one clock we take:

$$\mathtt{reg} := \{\{0\}, (0,1), \{1\}, (1,2), \dots, (d_{max}-1, d_{max}), \{d_{max}\}, (d_{max}, +\infty)\},$$

where d_{max} denotes the biggest constant appearing in δ, i.e., the transition function of the automaton. There are three kinds of regions: bounded intervals (denoted $\mathtt{reg}_I$), one-point regions (denoted $\mathtt{reg}_P$), and one unbounded interval $(d_{max}, +\infty)$. We will use the notation $\mathcal{I}_i$ for the region $(i-1, i)$. In a similar way, $\mathcal{I}_\infty$ will stand for $(d_{max}, +\infty)$. For $v \in \mathbb{R}_+$, let $\mathtt{reg}(v)$ denote its region; and let $\mathtt{fract}(v)$ denote the fractional part of v.

We will use the transition alphabet:

$$\overline{\Sigma} = \Sigma \cup \{(\mathtt{delay}, \varepsilon)\} \cup (\{\mathtt{delay}\} \times \Sigma).$$

A transition on $a \in \Sigma$ will represent an execution of an action. A transition on $(\mathtt{delay}, \varepsilon)$ will represent a passage of time with some valuation changing a region. Finally, we will have the most complicated case of $(\mathtt{delay}, a)$ transitions. Such a transition simulates a passage of time until a region becomes a one-point region, execution of the action at this moment, and letting some time pass to get into the next interval region. We will only present the transitions of the last type, the other two being simpler.

In $\mathcal{H}(\mathcal{A})$ the states will be finite words of the form $\Lambda_I^* \cdot \Lambda_\infty$, where $\Lambda_I = \mathcal{P}(Q \times \mathtt{reg}_I)$ and $\Lambda_\infty = \mathcal{P}(Q \times \{\infty\})$. The transitions on an action $(\mathtt{delay}, a)$ will have the form:

$$(\lambda_1 \dots \lambda_k, \lambda_\infty) \overset{(\mathtt{delay},a)}{\longrightarrow} (\delta' \lambda'_1 \dots \lambda'_{k-1}, \lambda''_\infty)$$

where the elements on the right are obtained by performing the following steps:

- (Letting the time pass to reach singleton region.) We change regions in λ_k. Every pair $(q, \mathcal{I}_d) \in \lambda_k$ becomes $(q, \{d\})$. Let us denote the result by λ_k^1.

- (Performing the action.) For $i = 1, \dots, k, \infty$ we take λ'_i, δ'_i such that: $\lambda^1_k \xrightarrow{a}_{\mathcal{A}} (\lambda'_k, \delta'_k)$ and $\lambda_i \xrightarrow{a}_{\mathcal{A}} (\lambda'_i, \delta'_i)$ for $i \neq k$.
- (Letting the time pass again) We increase again regions in λ'_k: from $\{d\}$ they become $\mathcal{I}_{d+1}$, or $\mathcal{I}_\infty$ if $d = d_{max}$.
- (Grouping the results) We put $\delta' = \bigcup \delta'_i \cup \{(q, \mathcal{I}_d) : (q, \{d\}) \in \lambda'_k, d < d_{max}\}$ and $\lambda''_\infty = \lambda'_\infty \cup \{(q, \mathcal{I}_\infty) : (q, \{d_{max}\}) \in \lambda'_k\}$.

We write $c \to c'$, $c \xrightarrow{(\mathtt{delay},\cdot)} c'$, $c \twoheadrightarrow c'$, $c \overset{\Sigma^*}{\twoheadrightarrow} c'$ to denote that we may go from a configuration c to c' using one transition, one transition reading a letter of the form $(\mathtt{delay}, \cdot)$, any number of transitions or any number of transitions reading only letters from Σ, respectively.

We say that a path is *good* if it passes through infinitely many transitions labelled by letters $(\mathtt{delay}, \cdot)$. The whole point of this type of transitions is that they allow to capture non Zeno behaviours:

Lemma 5. *$\mathcal{A}$ accepts an infinite non Zeno timed word iff there is a good path in $\mathcal{H}(\mathcal{A})$ starting in the state $(\{(q_0, \mathcal{I}_1)\}, \{\emptyset, I_\infty\})$ with only finitely many appearances of states from Q_-.*

3.2 Finding a Good Path in $\mathcal{H}(\mathcal{A})$

By Lemma 5, our problem reduces to deciding if there is a good path in $\mathcal{H}(\mathcal{A})$. The decision procedure works in two steps. In the first step we compute the set $\widehat{G}$ of all configurations from which there exists a good computation. Observe that if a configuration from $\widehat{G}$ has only states from Q_+ then this configuration is accepting. So, in the second step it remains to consider configurations that have states from both Q_- and Q_+. This is relatively easy as an accepting run from such a configuration consists of a finite prefix ending in a configuration without states from Q_- and a good run from that configuration. Hence, there is a good accepting computation from a configuration iff it is possible to reach from it a configuration in $\widehat{G}$ that has only Q_+ states. Once we know $\widehat{G}$, the later problem can be solved using the standard reachability tree technique.

Computing accepting configurations. We start with the second step of our procedure as it is much easier than the first one. We need to decide if from an initial state one can reach a configuration in $\widehat{G}$ having only Q_+ states. We can assume that we are given $\widehat{G}$ but we need to discuss a little how it is represented. It turns out that there are useful well-quasi-orders on configurations that allow to represent $\widehat{G}$ in a finitary way.

A *well-quasi-order* is a relation with a property that for every infinite sequence $c_1, c_2, \dots$ there exist indexes $i < j$ such that the pair (c_i, c_j) is in the relation.

The order we need is the relation, denoted $\preceq$, over configurations of $\mathcal{H}(\mathcal{A})$: we put $(\lambda_1 \dots \lambda_k, \lambda_\infty) \preceq (\lambda'_1 \dots \lambda'_{k'}, \lambda'_\infty)$ if $\lambda_\infty \subseteq \lambda'_\infty$ and there exists a strictly increasing function $f\colon \{1, \dots, k\} \to \{1, \dots, k'\}$ such that $\lambda_i \subseteq \lambda'_{f(i)}$ for each i. Observe that here we use the fact that each λ_i is a set so we can compare them by inclusion. This relation is somehow similar to the relation of being a subsequence, but we do not require that the corresponding letters are equal, only that the one from the smaller word is included in the one from the greater word. A standard application of Higman's lemma proves that $\preceq$ is a well-quasi-order.

The next lemma shows an important interplay between $\preceq$ relation and transitions of $\mathcal{H}(\mathcal{A})$.

Lemma 6. *Let c_1, c_1', c_2 be configurations of $\mathcal{H}(\mathcal{A})$ such that $c_1' \preceq c_1$. Whenever $c_1 \twoheadrightarrow c_2$, then there exist $c_2' \preceq c_2$ such that $c_1' \twoheadrightarrow c_2'$ and the second computation has the length not greater than the first one. Similarly, when from c_1 there exists a good computation, then from c_1' such a computation exists.*

The proof of the lemma follows from examining the definition of transitions. Some care is needed to ensure that the matching computation is good.

Corollary 7. *The set $\widehat{G}$ is downwards closed, so it can be described by the finite set of minimal elements that do not belong to it.*

As we have mentioned, there is a good accepting computation from a configuration iff it is possible to reach from it a configuration from $\widehat{G}$ that has only Q_+ states. A standard argument based on well-quasi-orders and examination of a finite part of the reachability tree shows that this property is decidable.

Lemma 8. *Let X be a downwards closed set in $\mathcal{H}(\mathcal{A})$. It is decidable if from a given configuration one can reach a configuration in X with all states in Q_+.*

Computing $\widehat{G}$. In the rest of the section we deal with the main technical problem of the proof that is computing the set $\widehat{G}$ of all configurations from which there exist a good computation. We will actually compute the complement of $\widehat{G}$. While we will use well-orderings in the proof, standard termination arguments do not work in this case. We need to use in an essential way a very special form of transitions our systems have.

We write $X{\uparrow} = \{c : \exists_{c' \in X} c' \preceq c\}$ for an upward closure of set X. Observe that by Lemma 6 the complement of $\widehat{G}$ is upwards closed.

Let set $pre^{\forall}_{\mathtt{delay}}$ (respectively $pre^{\forall}_{\Sigma^*}$) contain all configurations, from which after reading any letter $(\mathtt{delay}, \cdot)$ (any number of letters from Σ), we have to reach a configuration from X:

$$pre^{\forall}_{\mathtt{delay}}(X) = \{c : \forall_{c'}(c \xrightarrow{(\mathtt{delay},\cdot)} c' \Rightarrow c' \in X)\}$$
$$pre^{\forall}_{\Sigma^*}(X) = \{c : \forall_{c'}(c \overset{\Sigma^*}{\twoheadrightarrow} c' \Rightarrow c' \in X)\}.$$

We use these *pre* operations to compute a sequence of sets of configurations:

$$Z_{-1} = \emptyset \qquad Z_i = pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\mathtt{delay}}(Z_{i-1}{\uparrow})).$$

It is important that we may effectively represent and compare all the sets $Z_i{\uparrow}$. Because the relation $\preceq$ is a well-quasi-order, any upward closed set $X{\uparrow}$ may be represented by finitely many elements $c_1, \ldots, c_k$ (called *generators*) such that

$X\uparrow = \{c_1, \ldots, c_k\}\uparrow$. Moreover, an easy induction shows that $Z_{i-1}\uparrow \subseteq Z_i\uparrow$ for every i (because both $pre^{\forall}$ operations preserve inclusion). Once again, because relation $\preceq$ is a well-quasi-order, there has to be i such that $Z_{i-1}\uparrow = Z_i\uparrow$. Let us write Z_∞ for this Z_i.

First, we show that Z_∞ is indeed the complement of $\widehat{G}$.

Lemma 9. *There is a good computation from a configuration c iff $c \notin Z_\infty\uparrow$.*

To compute Z_∞ it is enough to show how to compute $Z_i\uparrow$ from $Z_{i-1}\uparrow$. This is the most difficult part of the proof. Once this is done, we can calculate all the sets $Z_i\uparrow$, starting with $Z_{-1} = \emptyset$ and ending when $Z_{i-1}\uparrow = Z_i\uparrow$.

The main idea in calculating $pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\mathtt{delay}}(X))$ is that the length of its generators may be bounded by some function in the length of generators of X.

Lemma 10. *Given an upwards closed set X we can compute a constant $D(X)$ (which depends also on our fixed automaton $\mathcal{A}$) such that the size of every minimal element of $pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\mathtt{delay}}(X))$ is bounded by $D(X)$*

Once we know the bound on the size of generators, we can try all potential candidates. The following lemma shows that it is possible.

Lemma 11. *For all upper-closed sets X, the membership in $pre^{\forall}_{\Sigma^*}(pre^{\forall}_{\mathtt{delay}}(X))$ is decidable.*

These two lemmas allow us to calculate Z_i from Z_{i-1}, and this is the last piece we need to complete the proof the theorem. The proofs of the lemmas are quite long, and require some additional notions. They can be found in the full version of the paper [25].

4 Constrained TPTL

We present a fragment of TPTL (timed propositional temporal logic) that can be translated to automata from our decidable class. We compare this fragment with other known logics for real time. We will be rather brief in presentations of different formalisms and refer the reader to recent surveys [9,24].

TPTL[8] is a timed extension of linear time temporal logic that allows to explicitly set and compare clock variables. We will consider the logic with only one clock variable, and we denote it by TPTL^1. The syntax of the logic is:

$$p \mid \alpha \wedge \beta \mid \alpha \vee \beta \mid \alpha \mathbb{U} \beta \mid \alpha \widetilde{\mathbb{U}}\, \beta \mid x \sim c \mid x.\alpha$$

where: p ranges over action letters, x is the unique clock variable, and $x \sim c$ is a comparison of x with a constant. We do not have negation in the syntax, but from the semantics it will be clear that the negation is definable.

The logic is evaluated over timed sequences $w = (a_1, t_1)(a_2, t_2)\ldots$ We define the satisfiability relation, $w, i, v \vDash \alpha$ saying that a formula α is true at a position i of a timed word w with a valuation v of the unique clock variable:

$w, i, v \vDash p$ if $a_i = p$
$w, i, v \vDash x.\alpha$ if $w, i, t_i \vDash \alpha$ $w, i, v \vDash x \sim c$ if $t_i - v \sim c$
$w, i, v \vDash \alpha \mathbb{U} \beta$ if $\exists_{j>i}$ $(w, j, v \vDash \beta$ and $\forall_{k \in (i,j)}$ $w, k, v \vDash \alpha)$
$w, i, v \vDash \alpha \widetilde{\mathbb{U}} \beta$ if $\forall_{j>i}$ $(w, j, v \vDash \beta$ or $\exists_{k \in (i,j)}$ $w, k, v \vDash \alpha)$

Until operators permit us to introduce sometimes and always operators:

$$F\alpha \equiv tt\mathbb{U}\alpha \qquad A\alpha \equiv f\!\!f\widetilde{\mathbb{U}}\alpha.$$

For the following, it will be interesting to note that the two until operators are inter-definable once we have always and sometimes operators:

$$\alpha\widetilde{\mathbb{U}}\beta \equiv A\beta \vee \beta\mathbb{U}\alpha \qquad \alpha\mathbb{U}\beta \equiv F\beta \wedge \beta\widetilde{\mathbb{U}}\alpha.$$

Observe that TPTL^1 subsumes metric temporal logic (MTL). For example: $\alpha\mathbb{U}_{(i,j)}\beta$ of MTL is equivalent to $x.(\alpha\mathbb{U}((x > i) \wedge (x < j) \wedge \beta))$. We will not present MTL here, but rather refer the reader to [10] where it is also shown that the following TPTL^1 formula is not expressible in MTL:

$$x.(F(b \wedge F(c \wedge x \leq 2))). \tag{1}$$

Informally, the formula says that there is an event b followed by an event c in less than 2 units of time.

The satisfiability problem over infinite timed sequences is undecidable for MTL [21], hence also for TPTL^1. Using our decidability result for alternating timed automata, we can nevertheless find a decidable fragment, that we call Constrained TPTL. The definition of this fragment will use an auxiliary notion of *positive TPTL*^{1} *formulas*:

$$p \mid x.\varphi \mid x \sim c \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \varphi\widetilde{\mathbb{U}}\psi \mid F((x \leq c) \wedge \psi).$$

These formulas can be translated into automata where all states are accepting. Observe that the formula (1) belongs to the positive fragment if we add redundant $(x \leq 2)$ after b. The set of formulas of *Constrained TPTL* is:

$$p \mid x.\varphi \mid x \sim c \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \alpha\mathbb{U}\beta \mid \varphi \quad \varphi \text{ positive.}$$

A translation of Constrained TPTL to automata is similar in a spirit to that for Safety MTL [22]. Once again we refer the reader to the full version [25].

Theorem 12. *It is decidable if there is a non Zeno timed word that is a model of a given Constrained TPTL formula. The complexity of the problem cannot be bounded by a primitive recursive function.*

Safety MTL [22] can be seen as a MTL fragment of positive TPTL. Indeed, both formalisms can be translated to automata with only accepting states, but the automata obtained from MTL formulas have also the locality property (cf. [22]). This property ensures that the clock is always reset when changing state. The example (1) shows that this is not the case for TPTL.

Using equivalences mentioned above, FlatMTL [11] with pointwise non Zeno semantics can be presented as a set of formulas given by the grammar:

$$p \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \alpha \mathbb{U}_J \beta \mid \chi \mathbb{U}_I \beta \mid \chi \qquad J \text{ bounded and } \chi \in MITL$$

The original definition admits more constructs, but they are redundant in the semantics we consider. Both FlatMTL and Constrained TPTL use two sets of formulas. The MTL part of the later logic would look like

$$p \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \alpha \mathbb{U}_I \beta \mid \varphi \qquad \varphi \text{ positive.}$$

From this presentation it can be seen that there are at least two important differences: Constrained TPTL does not have restrictions on the left hand side of until, and it uses positive fragment instead of MITL. We comment on these two aspects below.

Unrestricted until makes the logic more expressive but also more difficult algorithmically. For example, already the logic generated by the later grammar without the clause for positive formulas has non primitive recursive complexity. This should be contrasted with Expspace-completeness result for FlatMTL.

The use of positive fragment instead of MITL is also important. The two formalisms have very different expressive powers. The crucial technical property of MITL is that a formula of the form $\alpha \mathbb{U}_I \beta$ can change its value at most three times in every unit interval. This is used in the proof of decidability of FlatMTL, as the MITL part can be described in a "finitary" way. The crucial property of the positive fragment is that it can express only safety properties (and all such properties). We can remark that by reusing the construction of [21] we get undecidability of the positive fragment extended with a formula expressing that some action appears infinitely often. Theorem 4 implies that this is true even if we do not use punctual constraints in the positive fragment. In conclusion, we cannot add MITL to the positive fragment without losing decidability.

Acknowledgements. The second author would like to thank Abraham Riche for his cooperation in early stages of this work.

References

1. Abdulla, P., Jonsson, B.: Veryfying networks of timed processes. In: Steffen, B. (ed.) TACAS 1998. LNCS, vol. 1384, pp. 298–312. Springer, Heidelberg (1998)
2. Abdulla, P., Jonsson, B.: Timed Petri nets and BQOs. In: Colom, J.-M., Koutny, M. (eds.) ICATPN 2001. LNCS, vol. 2075, pp. 53–70. Springer, Heidelberg (2001)
3. Abdulla, P.A., Deneux, J., Ouaknine, J., Quaas, K., Worrell, J.: Universality analysis for one-clock timed automata. Fundam. Inform. 89(4), 419–450 (2008)
4. Abdulla, P.A., Ouaknine, J., Quaas, K., Worrell, J.: Zone-based universality analysis for single-clock timed automata. In: Arbab, F., Sirjani, M. (eds.) FSEN 2007. LNCS, vol. 4767, pp. 98–112. Springer, Heidelberg (2007)
5. Alur, R., Dill, D.: A theory of timed automata. Theoretical Computer Science 126, 183–235 (1994)
6. Alur, R., Feder, T., Henzinger, T.A.: The benefits of relaxing punctuality. J. ACM 43(1), 116–146 (1996)
7. Alur, R., Fix, L., Henzinger, T.: Event-clock automata: A determinizable class of timed automata. Theoretical Computer Science 204 (1997)

8. Alur, R., Henzinger, T.A.: A really temporal logic. J. ACM 41(1), 181–204 (1994)
9. Bouyer, P.: Model-checking timed temporal logics. In: Workshop on Methods for Modalities (M4M-5), Cachan, France. Electronic Notes in Theoretical Computer Science. Elsevier Science Publishers, Amsterdam (2009)
10. Bouyer, P., Chevalier, F., Markey, N.: On the expressiveness of TPTL and MTL. In: Sarukkai, S., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 432–443. Springer, Heidelberg (2005)
11. Bouyer, P., Markey, N., Ouaknine, J., Worrell, J.: The cost of punctuality. In: LICS, pp. 109–120 (2007)
12. Bouyer, P., Markey, N., Ouaknine, J., Worrell, J.: On expressiveness and complexity in real-time model checking. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 124–135. Springer, Heidelberg (2008)
13. Hirshfeld, Y., Rabinovich, A.M.: Logics for real time: Decidability and complexity. Fundam. Inform. 62(1), 1–28 (2004)
14. Hung, D.V., Ji, W.: On the design of hybrid control systems using automata models. In: Chandru, V., Vinay, V. (eds.) FSTTCS 1996. LNCS, vol. 1180, pp. 156–167. Springer, Heidelberg (1996)
15. Lasota, S., Walukiewicz, I.: Alternating timed automata. In: Sassone, V. (ed.) FOSSACS 2005. LNCS, vol. 3441, pp. 250–265. Springer, Heidelberg (2005)
16. Lasota, S., Walukiewicz, I.: Alternating timed automata. ACM Trans. Comput. Log. 9(2) (2008)
17. Mostowski, A.W.: Hierarchies of weak automata and week monadic formulas. Theoretical Computer Science 83, 323–335 (1991)
18. Murlak, F.: Weak index versus borel rank. In: STACS, Dagstuhl Seminar Proceedings, pp. 573–584. Dagsr (2008)
19. Ouaknine, J., Worrell, J.: On the language inclusion problem for timed automata: Closing a decidability gap. In: Proc. LICS 2004, pp. 54–63 (2004)
20. Ouaknine, J., Worrell, J.: On the decidability of metric temporal logic. In: LICS, pp. 188–197 (2005)
21. Ouaknine, J., Worrell, J.: On metric temporal logic and faulty Turing machines. In: Aceto, L., Ingólfsdóttir, A. (eds.) FOSSACS 2006. LNCS, vol. 3921, pp. 217–230. Springer, Heidelberg (2006)
22. Ouaknine, J., Worrell, J.: Safety metric temporal logic is fully decidable. In: Hermanns, H., Palsberg, J. (eds.) TACAS 2006. LNCS, vol. 3920, pp. 411–425. Springer, Heidelberg (2006)
23. Ouaknine, J., Worrell, J.: On the decidability and complexity of metric temporal logic over finite words. Logical Methods in Computer Science 3(1) (2007)
24. Ouaknine, J., Worrell, J.: Some recent results in metric temporal logic. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 1–13. Springer, Heidelberg (2008)
25. Parys, P., Walukiewicz, I.: Weak alternating timed automata. HAL (2009), http://hal.archives-ouvertes.fr/hal-00360122/fr/
26. Vardi, M.Y., Wolper, P.: Automata theoretic techniques for modal logics of programs. In: Sixteenth ACM Symposium on the Theoretical Computer Science (1984)
27. Wagner, K.: Eine topologische Charakterisierung einiger Klassen regulärer Folgenmengen. J. Inf. Process. Cybern. EIK 13, 473–487 (1977)
28. Wagner, K., Staiger, L.: Automatentheoretische und automatenfreie charakterisierungen topologischer klassen regularer folgenmengen. EIK 10, 379–392 (1974)
29. Wilke, T.: Classifying discrete temporal properties. Habilitation thesis, Kiel, Germany (1998)

A Decidable Characterization of Locally Testable Tree Languages

Thomas Place and Luc Segoufin

INRIA and LSV at ENS Cachan

Abstract. A regular tree language L is locally testable if the membership of a tree into L depends only on the presence or absence of some neighborhoods in the tree. In this paper we show that it is decidable whether a regular tree language is locally testable.

1 Introduction

This paper is part of a general program trying to understand the expressive power of first-order logic over trees. We say that a class of regular tree languages has a decidable characterization if the following problem is decidable: given as input a finite tree automaton, decide if the recognized language belongs to the class in question. Usually a decision algorithm requires a solid understanding of the expressive power of the corresponding class and is therefore useful in any context where a precise boundary of this expressive power is crucial. For instance, we do not possess yet a decidable characterization of the tree languages definable in FO($\leq$), the first-order logic using a binary predicate $\leq$ for the ancestor relation.

We consider here the class of tree languages definable in a fragment of FO($\leq$) known as *Locally Testable* (LT). A language is in LT if its membership depends only on the presence or absence of neighborhoods of a certain size in the tree. A closely related family of languages is the class LTT of *Locally Threshold Testable* languages. Membership in such languages is obtained by counting the number of neighborhoods of a certain size up to some threshold. The class LT is the special case where no counting is done, the threshold is 1. In this paper we provide a decidable characterization of the class LT over trees.

Decidable characterizations are usually obtained by exhibiting a set of closure properties that holds exactly for the languages in the class under investigation. It is therefore necessary to have a formalism for expressing these properties. This formalism must also come with some tools for proving that the properties do characterize the class, typically with induction mechanisms, but also for proving the decidability of those properties.

Over words one formalism turned out to be successful for characterizing many class of regular languages. The closure properties are expressed as identities on the syntactic monoid of the regular language. The syntactic monoid of a regular language being the transition monoid of its minimal deterministic automata. For instance the class of languages definable in FO($\leq$) is characterized by the fact that their syntactic monoid is aperiodic. The later property corresponds to the identity $x^{\omega} = x^{\omega+1}$ where ω is the size of the monoid. This equation is easily verifiable automatically on the syntactic monoid. Similarly, the classes

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 285–296, 2009.

LTT and LT have been characterized using decidable identities on the syntactic monoid [BS73, McN74, BP89, TW85].

Over trees the situation is more complex and right now there is no formalism that can easily express all the closure properties of the classes for which we have a decidable characterization. The most successful formalism is certainly forest algebra [BW07]. For instance, forest algebra was used for obtaining decidable characterizations for the classes of tree languages definable in EF+EX [BW06], EF+F^{-1} [Boj07b, Pla08], BC-$\Sigma_1(<)$ [BSS08, Pla08], $\Delta_2(\leq)$ [BS08, Pla08]. However it is not clear yet how to use forest algebra for characterizing the class LTT over trees and a different formalism was used for obtaining a decidable characterization for this class [BS09].

We were not able to obtain a reasonable set of identities for LT either by using forest algebra or the formalism used for characterizing LTT. Our approach is slightly different.

There is another technique that worked on words for deciding the class LT. It is based on the "delay theorem" [Str85, Til87] for computing the expected size of the neighborhoods: Given a regular language L, a number k can be computed such that if L is in LT then it is in LT by investigating the neighborhoods of size k. Once this k is available, deciding whether L is indeed in LT or not is a simple exercise. On words, a decision algorithm for LT (and also for LTT) has been obtained successfully using this approach [Boj07a]. Unfortunately all efforts to prove a similar delay theorem on trees have failed so far.

We obtain a decidable characterization of LT by combining the two approaches mentioned above. We first exhibit a set of necessary and decidable conditions for a regular tree language to be in LT. Those conditions are expressed using the formalism introduced for characterizing LTT. We then show that for languages satisfying such conditions one can compute the expected size of the neighborhoods. Using this technique we obtain a characterization of LT for ranked trees and for unranked unordered trees.

Other related work. There exists several formalisms that were used for expressing identities corresponding to several classes of languages but not in a decidable way. Among them let us mention the notion of preclones introduced in [EW05] as it is close to the one we use in this paper for expressing our necessary conditions.

Organization of the paper. We start with ranked trees and give the necessary notations and preliminary results in Section 2. Section 3 exhibits several conditions and proves they are decidable and necessary for being in LT. In Section 4 we show that for the languages satisfying the necessary conditions the expected size of the neighborhoods can be computed, hence concluding the decidability of the characterization. Finally in Section 5 we show how our result extends to unranked trees. Due to space limitations several proofs are missing.

2 Notations and Preliminaries

We first prove our result for the case of binary trees. The case of unranked unordered trees will be considered in Section 5.

Trees. We fix a finite alphabet Σ, and consider finite binary trees with labels in Σ. All the results presented here extend to arbitrary ranks in a straightforward way. A *language* is a set of trees. We use standard notations for trees. For instance by the *descendant* (resp. ancestor) relation we mean the reflexive transitive closure of the child (resp. inverse of child) relation.

Given a tree t and a node x of t the subtree of t rooted at x, consisting of all the nodes of t which are descendants of x, is denoted by $t|_x$. A *context* is a tree with a designated (unlabeled) leaf called its *port* which acts as a hole. Given contexts C and C', their concatenation $C \cdot C'$ is the context formed by identifying the root of C' with the port of C. A tree $C \cdot t$ can be obtained similarly by concatenating a context C and a tree t. Given a tree t and two nodes x, y of t such that y is a descendant (not necessarily strict) of x, the context $C = t[x, y]$ is defined from $t_1 = t|_x$ by replacing $t_1|_y$ by a port. In this case we say that C is a context *occurring* in t.

Types. Let t be a tree and x be a node of t and k be a positive integer, the *k-type* of x in t is the (isomorphism type of the) restriction of $t|_x$ to the set of nodes of t at distance at most k from x. A k-type τ *occurs* in a tree t if there exists a node in t of k-type τ. If C is a context occurring in a tree t then the k-type of a node of C is the k-type of that node in t. Notice that the k-type of a node of C depends on the surrouding tree t, in particular the port of C has a k-type.

Given two trees t and t' we denote by $t \preccurlyeq_k t'$ the fact that all k-types that occur in t also occur in t'. Similarly we can speak of $t \preccurlyeq_k C$ when t is a tree and C is a context occurring in some tree t'. We denote by $t \simeq_k t'$ the property that the root of t and the root of t' have the same k-type and t and t' agree on their k-types: $t \preccurlyeq_k t'$ and $t' \preccurlyeq_k t$. Note that when k is fixed the number of k-types is finite and hence the equivalence relation $\simeq_k$ has finite index.

A language L is said to be κ-locally testable (is in LT_κ) if L is a union of equivalence classes of $\simeq_\kappa$. A language is said to be *locally testable* (is in LT) if there is a κ such that it is κ-locally testable. In words this says that in order to test whether a tree t belongs to L it is enough to check for the presence or absence of κ-types in t, for some big enough κ.

The problem. We want an algorithm deciding if a given regular language is in LT. If complexity does not matter, we can assume that the given language L is given as a MSO formula. Another option would be to start with a bottom-up tree automata for L or, even better, the minimal deterministic bottom-up tree automata that recognizes L. The main difficulty is to compute a bound on κ, the size of the neighborhood, whenever such a κ exists.

The string case is a special case of the tree case as it corresponds to trees of rank 1. A decision procedure for LT was obtained in the string case independently by [BS73, McN74]. It is based on a characterization of the syntactic semigroup of the language by means of the equations $exe = exexe$ and $exeye = eyexe$, where e is an arbitrary idempotent ($ee = e$) while x and y are arbitrary elements of the semigroup. The equations are then easily verified on the syntactic semigroup.

In the case of trees, we were not able to obtain a reasonably simple set of identities for characterizing LT. Nevertheless we can show:

Theorem 1 *It is decidable whether a regular tree language is in LT.*

Our strategy for proving Theorem 1 is as follows. In a first step we provide necessary, and decidable, conditions for a language to be in LT. In a second step we show that if a language verifies those conditions then we can compute a κ such that it is in LT iff it is in LT_κ. Finally we show that once κ is fixed, it is decidable whether a regular language is a finite union of classes of $\simeq_\kappa$.

Before starting providing the proof details we note that there exists examples showing that the necessary conditions are not sufficient, see the end of Section 3. This is not immediate to see and goes beyond the scope of this paper. We also note that the problem of finding κ whenever such a κ exists is a special case of the *delay-theorem*. In the case of LT, the delay theorem says that if a finite state automaton A recognizes a language in LT then this language must be in LT_κ for a κ computable from A. This theorem holds over strings [Str85, Til87] and can be used in order to decide whether a regular language is in LT as explained in [Boj07a]. We were not able to prove such a general theorem for trees. Our second step can be seen as a particular case of the delay theorem for languages satisfying the conditions provided by the first step.

3 Necessary Conditions

In this section we exhibit necessary conditions for a regular language to be in LT. These conditions will play a crucial role in our decision algorithm. These conditions are expressed using the same formalism as the one used in [BS09] for characterizing LTT.

Guarded operations. Let t be a tree, and x, x' be two nodes of t such that x and x' are not related by the descendant relationship. The *horizontal swap* of t at nodes x and x' is the tree t' constructed from t by replacing $t|_x$ with $t|_{x'}$ and vice-versa, see Figure 1 (left). A horizontal swap is said to be *k-guarded* if x and x' have the same k-type.

Let t be a tree and x, y, z be three nodes of t such that x, y, z are not related by the descendant relationship and such that $t|_x = t|_y$. The *horizontal transfer* of t at x, y, z is the tree t' constructed from t by replacing $t|_y$ with a copy of $t|_z$, see Figure 1 (right). A horizontal transfer is k-guarded if x, y, z have the same k-type.

Let t be a tree of root a, and x, y, z be three nodes of t such that y is a descendant of x and z is a descendant of y. The *vertical swap* of t at x, y, z is

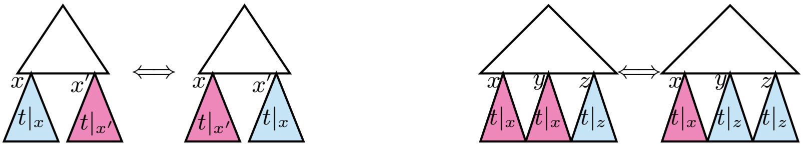


Fig. 1. Horizontal Swap (left) and Horizontal Transfer (right)

the tree t' constructed from t by swapping the context between x and y with the context between y and z, see Figure 2 (left). A vertical swap is *k-guarded* if x, y, z have the same k-type.

Let t be a tree of root a, and x, y, z be three nodes of t such that y is a descendant of x and z is a descendant of y such that $\Delta = t[x, y] = t[y, z]$. The *vertical stutter* of t at x, y, z is the tree t' constructed from t by removing the context between x and y, see Figure 2 (right). A vertical stutter is k-guarded if x, y, z have the same k-type.

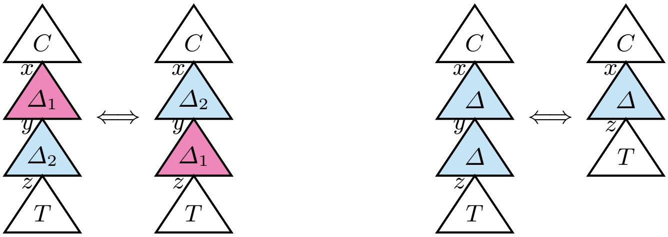


Fig. 2. Vertical Swap (left) and Vertical Stutter (right)

Let L be a tree language and k be a number. If X is any of the four constructions above, horizontal or vertical swap, or vertical stutter or horizontal transfer, we say that L is *closed under k-guarded X* if for every tree t and every tree t' constructed from t using k-guarded X then t is in L iff t' is in L. Notice that being closed under k-guarded X implies being closed under k'-guarded X for $k' > k$. An important observation is that each of the k-guarded operations preserves $(k + 1)$-types.

If L is closed under all the k-guarded operations described above, we say that L is *k-tame.* A language is said to be *tame* if it is k-tame for some k. The following simple result shows that tameness is a necessary condition for LT.

Proposition 1 *If L is in LT then L is tame.*

We now turn to the decision procedure for testing tameness. If k is fixed, it is simple to check whether L is k-tame, see for instance [BS09]. The following proposition shows that k can be assumed to be bounded by a simple function of the size of any bottom-up tree automata recognizing L. It can be shown using a pumping argument.

Proposition 2 *Given a regular language L, it is decidable whether L is tame. Moreover, if this is the case, a k such that L is k-tame can be effectively computed.*

Example 1 *Over strings tameness characterizes exactly LT as vertical swap and vertical stutter are exactly the extensions to trees of the equations given in Section 2 for LT. Over trees this is no longer the case as the following example shows. For simplifying the presentation we assume that nodes may have between*

0 to three children. All trees in our language L have the same structure consisting of a root of label **a** *from which exactly three sequences of nodes with only one child (strings) are attached. The trees in L have therefore exactly three leaves, and those must have three distinct labels among* $\{\mathbf{h}_1, \mathbf{h}_2, \mathbf{h}_3\}$. *One branch, excepted for its leaf, must be of the form b^*cd^*, another one of the form b^*cd^* and the last one of the form $b^*c'd^*$, where* **b**, **c**, **c'** *and* **d** *are distinct labels. The reader can verify that L is tame. It is not in LT because replacing exactly one node of label* **c** *by a node of label* **c'** *in a tree with long branches yields a tree with the same neighborhood but not in L.*

4 Deciding LT for Tame Languages

In this section we show that it is decidable whether a regular tree language is in LT. In view of Proposition 1 and Proposition 2 it is sufficient to show that it is decidable whether a tame regular tree language is in LT. Hence Theorem 1 follows from the following proposition.

Proposition 3 *Assume L is a tame regular tree language. Then it is decidable whether there exists a κ such that L is in LT_κ.*

Proof. Assume L is tame. Then from Proposition 2 one can effectively compute a k such that L is k-tame. The proof of the proposition is then divided in two steps. The first one shows that for k-tame languages, if such a κ exists then κ is at most exponential in k. The second step, Lemma 1 below, shows that when κ is fixed then being a union of $\simeq_\kappa$ is decidable. The proof of the second step is straightforward and is left to the reader.

Lemma 1 Let L be a regular tree language and κ a number. It is decidable whether L is in LT_κ or not.

The rest of this section is devoted to the proof of the first step showing that for k-tame regular tree language a bound on κ can be obtained. This is a consequence of the lemma below. Recall that for each k, the number of k-types is finite. Let β_k be this number.

Lemma 2 Let L be a k-tame regular tree language. Set $l = \beta_k + 1$. Then for all $l' > l$ and any two trees t, t' if $t \simeq_l t'$ then there exists two trees T, T' with

1. $t \in L$ iff $T \in L$
2. $t' \in L$ iff $T' \in L$
3. $T \simeq_{l'} T'$

To see that the first step follows from Lemma 2, assume that L is a k-tame regular tree language in LT. Then, by definition of LT, L is in $\mathrm{LT}_{l'}$ for some l'. If $l' > \beta_k + 1$ then, from Lemma 2, L is also in LT_l. Hence for testing membership in LT it is sufficient to test membership in LT_l for $l = \beta_k + 1$ which is decidable by Lemma 1.

Before proving Lemma 2 we need some extra terminology. A non-empty context C occurring in a tree t is a *loop of k-type* τ if the k-type of its root and the k-type of its port is τ. A non-empty context C occurring in a tree t is a k-loop if there is some k-type τ such that C is a loop of k-type τ. Note that if C is a loop of k-type τ and x is a node of type τ in a tree t then inserting C at node x does not modify the k-type of the nodes of t (it may add new k-types, coming from the nodes of C). In particular the k-type of x is unchanged. Given a context C we call the path from the root of C to its port the *principal path of* C. Finally, the result of the *insertion* of a k-loop C at a node x of a tree t is a tree T such that if $t = D \cdot t|_x$ then $T = D \cdot C \cdot t|_x$. Typically an insertion will occur only when the k-type of x is τ and C is a loop of k-type τ. In this case the k-types of the nodes of t are unchanged by this operation.

Proof (of Lemma 2). Suppose that L is k-tame. Recall that the number of k-types is β_k. Therefore, by choice of l, in every branch of a l-type one can find at least one k-type that is repeated. This provides many k-loops that can be moved around whenever necessary in order to obtain similar bigger types.

Take $l' > l$, we build T and T' from t and t' by inserting k-loops in t and t' without affecting their membership in L.

Let $B = \{\tau_0, ..., \tau_n\}$ be the set of k-types τ such that there is a loop of k-type τ in t or in t'. For each $\tau \in B$ we fix a context C_τ as follows. Because $\tau \in B$ there is a context C in t or t' that is a loop of k-type τ. For each $\tau \in B$, we fix arbitrarily such a C and set C_τ as $\underbrace{C \cdot \ldots \cdot C}_{l'}$, l' concatenation of the context C.

Notice that the path from the root of C_τ to its port is then bigger than l'.

We now describe the construction of T from t. The construction of T' from t' is done similarly. The tree T is constructed by inserting simultaneously a copy of the context C_τ at all nodes of type $\tau \in B$ of t.

We now show that T and T' have the desired properties. The third property is easy to verify (proof omitted in this abstract).

Claim 1 $T \simeq_{l'} T'$

The other two properties, $t \in L$ iff $T \in L$ and $t' \in L$ iff $T' \in L$, are not obvious at all. This is the difficult part of the proof and it requires tameness of the language. It is a consequence of Lemma 3 below. □

In order to conclude the proof of Proposition 3 it remains to show the following lemma. This lemma shows a key consequence of the fact that L is k-tame: the insertion of k-loops that don't introduce new $(k+1)$-types preserves membership into L.

Lemma 3 Let L be a k-tame regular tree language. Let t be a tree and x a node of t of k-type τ. Let t' be another tree such that $t \simeq_{k+1} t'$ and C be a loop of k-type τ in t'. Consider the tree T constructed from t by inserting a copy of C at x. Then $t \in L$ iff $T \in L$.

Proof. The proof is done in two steps. First we use the k-tame property of L to show that we can insert a k-loop C' at x in t such that the principal path of C is the same as the principal path of C'. By this we mean that there is a bijection from the principal path of C' to the principal path of C that preserves $(k+1)$-types. In a second step we replace one by one the subtrees hanging from the principal path of C' with the corresponding subtrees in C.

First some terminology. Given two nodes x, y of some tree T, we say that x is a **l**-ancestor of y if y is a descendant of the left child of x. Similarly we define **r**-ancestorship.

Consider the context C occurring in t'. Let $y_0, \cdots, y_n$ be the nodes of t' on the principal path of C and $\tau_0, \cdots, \tau_n$ be their respective $(k+1)$-type. For $0 \leq i < n$, set c_i to **l** if y_{i+1} is a left child of y_i and **r** otherwise.

From t we construct using k-guarded swaps and k-vertical stutter a tree t_1 such that there is a sequence of nodes $x_0, \cdots, x_n$ in t_1 with for all $0 \leq i < n$, x_i is of type τ_i and x_i is an c_i-ancestor of x_{i+1}. The tree t_1 is constructed by induction on n. If $n = 0$ then this is a consequence of $t \simeq_{k+1} t'$ that one can find in t a node of type τ_0. Consider now the case $n > 0$. By induction we have constructed from t a tree t'_1 such that $x_0, \cdots, x_{n-1}$ is an appropriate sequence in t'_1. By symmetry we consider the case where y_n is the left child of y_{n-1}. Because all k-guarded operations preserve $(k+1)$-types, we have $t \simeq_{k+1} t'_1$ and hence there is a node x of t'_1 of type τ_n. If x_{n-1} is a **l**-ancestor of x then we are done. Otherwise consider the left child x' of x and notice that because y_n is a child of y_{n-1} and x_{n-1} has the same $(k+1)$-type than y_{n-1} then x', y_n and x have the same k-type.

We know that x is not a descendant of x'. There are two cases. If x and x' are not related by the descendant relationship then by k-guarded swaps we can replace the subtree rooted in x' by the subtree rooted in x and we are done. If x is an ancestor of x' then the context between x and x' is a k-loop and we can use k-guarded vertical stutter to duplicate it. This places x as the left child of x_{n-1} and we are done.

From t_1 we construct using k-guarded swaps and k-guarded vertical stutter a tree t_2 such that there is a path $x_0, \cdots, x_n$ in t_2 with for all $0 \leq i < n$, x_i is of type τ_i.

Consider the sequence $x_0, \cdots, x_n$ obtained in t_1 from the previous step. Recall that the k-type of x_0 is that same as the k-type of x_n. Hence using k-guarded vertical stutter we can duplicate in t_1 the context rooted in x_0 and whose port is x_n. Let t'_1 the resulting tree. We thus have two copies of the sequence $x_0, \cdots, x_n$ that we denote by the *top copy* and the *bottom copy.* Assume x_i is not a child of x_{i-1}. Notice that the context between the appropriate child of x_{i-1} and x_i is a k-loop. Using k-guarded vertical swap we can move the top copy of this context next to its bottom copy. Using k-guarded vertical stutter this extra copy can be removed. We are left with an instance of the initial sequence in the bottom copy, while in the top one x_i is a child of x_{i-1}. Repeating this argument yields the desired tree t_2.

Consider now the context $C' = t_2[x_0, x_n]$. It is a loop of k-type τ. Let T' be the tree constructed from t by inserting C' in x. The proof of the following claim is omitted in this abstract.

Claim 2 *$T' \in L$ iff $t \in L$.*

It remains to show that $T' \in L$ iff $T \in L$. By construction of T' we have $C' \preccurlyeq_{k+1} t$. Consider now a node x_i in the principal path of C'. Let T_i be the subtree branching out the principal path of C at x_i and T'_i be the subtree branching out the principal path of C' at x_i. The claim below shows that replacing T'_i with T_i does not affect membership into L. Hence a repeated use of that claim eventually shows that $T' \in L$ iff $T \in L$.

Claim 3 *Let u and u' be two trees. Assume s and s' are subtrees respectively of u and u' such that the roots of s and s' have the same k-type. Consider the context D such that $u = Ds$.*

If $s \preccurlyeq_{k+1} D$ and $s' \preccurlyeq_{k+1} D$ then $Ds \in L$ iff $Ds' \in L$.

Proof (sketch). The proof is done by induction on the depth of s'. The idea is to replace s with s' node by node.

Assume first that s' is of depth less than k. Then because the k-type of the roots of s and s' are equal, we have $s = s'$ and the result follows.

Assume now that s' is of depth greater than k. Let x be the root of s. Let τ be the $(k+1)$-type of the root of s'. Because $s' \preccurlyeq_{k+1} D$ we know that there exists a node y in D of type τ. We consider two cases depending on the relation between x and y.

- If y is an ancestor of x, let E be $u[y, x]$ and notice that x and y have the same k-type. Hence we can duplicate E using a k-guarded vertical stutter. The resulting tree is DEs and because L is k-tame, $DEs \in L$ iff $Ds \in L$. Let z be the root of E in DEs. Notice that by construction z is of type τ. Let s_1 be the subtree of DEs rooted at the left child of z and let s'_1 be the subtree of s' rooted at the left child of the root of s'. By construction $s_1 \preccurlyeq_{k+1} D$, $s'_1 \preccurlyeq_{k+1} D$. Because their parent have the same $(k+1)$-type, the roots of s_1 and s'_1 have the same k-type. As the depth of s'_1 is strictly smaller than the depth of s', by induction we can replace s_1 by s'_1 without affecting membership into L. Similarly we do the same for the right child and we are done.
- Assume now that x and y are not related by the descendant relationship. We know that x, y have the same k-type and that $s \preccurlyeq_{k+1} D$. Let s'' be the subtree of Ds rooted at y. It can be shown that, as a consequence of tameness (this is where k-guarded horizontal transfer is used), replacing Ds by Ds'' does not affect membership in L . As y'' is of type τ, we can proceed by induction as above and replace the left and right subtrees of s'' by their corresponding subtrees of s' to get the desired result. □

This concludes the proof of the decision algorithm in the case of trees. We note that in the case of strings Lemma 3 is extremely powerful and is sufficient for showing that tameness implies membership in LT. This is due to the fact that, on strings, any two nodes with the same type induce a loop and therefore Lemma 3 applies to this loop. This lemma can then be used for transforming by induction a string to any other one with the same occurrences of types. However over trees this no longer work as the two nodes may be incomparable.

5 Unranked Trees

In this section we consider unranked unordered trees, where each node has an arbitrary number of children but no order is assumed on these children. Our goal is to extend the result of the previous section and provide a decidable characterization of Locally Testable languages of unranked trees. Due to space limitations, we mostly only mention here our results, their proofs will appear in the journal version of this paper.

The first issue is to find an appropriate notion of LT for unranked trees. Recall that a language of binary trees is LT if its membership depends only on the presence or absence of neighborhoods of a certain size. With unranked trees, there may be infinitely many different possible neighborhoods of a certain size and hence this definition does not imply that the language is regular[1]. The idea is to replace isomorphism types with FO definable types such that for each isomorphism type there are only finitely many FO definable types.

We will use the following notion of type. The definition of k-type remains unchanged: it is the isomorphism type of tree induced by nodes at depth at most k. As mentioned above there are now infinitely many k-types. We therefore introduce a more flexible notion that depends on one extra parameter l that restricts the horizontal information. It is defined by induction on k. Consider an unordered tree t. For $k = 0$, the (k, l)-type of t is just the label of the root of t. For $k > 0$ the (k, l)-type of t is the label of its root together with, for each $(k - 1, l)$-type, the number, up to threshold l, of children of the root of t of this type.

From this we define two classes of Locally Testable languages. The most general one, denoted ALT (A for *Aperiodic*), is defined as follows. Two trees are (k, l)-equivalent if they have the same occurrences of (k, l)-types and their roots have the same (k, l)-type. A language L is in ALT if there is a k and a l such that L is a union of (k, l)-equivalence classes. In the framework of forest algebra [BW07], languages in ALT have a syntactic forest algebra whose horizontal monoid satisfies $h^{\omega} = h^{\omega+1}$.

The second one, denoted ILT in the sequel (I for *Idempotent*), assumes $l = 1$: A language L is in ILT if there is a k such that L is a union of $(k, 1)$-equivalence classes. In the framework of forest algebra languages in ILT have a syntactic forest algebra whose horizontal monoid satisfies $h + h = h$.

The main result of this section is that we can extend the decidable characterization obtained for ranked trees to both ILT and ALT.

Theorem 2 *It is decidable whether a regular unranked tree language is ILT.*
It is decidable whether a regular unranked tree language is ALT.

The notions of k-tame and (k, l)-tame are defined as in Section 3 using k-types and (k, l)-types. For unranked unordered trees both notions are identical:

[1] In this section, by regular we mean definable in MSO using the child predicate. There is also an equivalent automata model.

Lemma 4 For every regular unordered tree language L and every k there is a number l, computable from k and L, such that if L is k-tame, then L is (k, l)-tame.

In the idempotent case we can completely characterize ILT. It corresponds to tameness together with an extra closure property denoted *horizontal stutter*. A tree language L is closed under horizontal stutter iff for any tree t and any node x of t, replacing $t|_x$ with two copies of $t|_x$ does not affect membership into L.

Theorem 3 *A regular unordered tree language is in ILT iff it is tame and closed under horizontal stutter.*

This immediately implies the ILT part of Theorem 2 as both tameness and closure under horizontal stutter are decidable properties.

Proof (sketch). That the right-hand side conditions are necessary is obvious. For the other direction we first use Lemma 4 to compute l from k. We then prove the adaptation of Claim 3 to forests. Given two forests h and h', $h + h'$ denotes the forest consisting of the trees of h followed by the trees of h'. Given a forest h, $a(h)$ is the tree whose root has label a and whose children are the trees of h.

Consider now two trees t and t' that are (k, l)-equivalent. Then $t = a(h)$ and $t' = a(h')$ for some a and forests h and h' that are $(k-1, l)$-equivalent. Assume now that $t \in L$. By horizontal stutter, $a(h+h)$ is also in L. Because h and h' are $(k-1, l)$-equivalent we can use Claim 3 and replace one copy of h by h' and have $a(h + h') \in L$. Claim 3 applies again and yields $a(h' + h') \in L$. By horizontal stutter this implies that $t' \in L$.

Hence L is a union of (k, l)-equivalence classes. From closure under horizontal stutter this implies that L is a union of $(k, 1)$-equivalence classes and is in ILT. □

For ALT we follow the lines of the binary tree case and Theorem 2 follows from the unranked variant of Proposition 3:

Proposition 4 *Assume L is a tame regular unordered tree language. Then it is decidable whether there exists a κ such that L is in ALT$_\kappa$.*

6 Discussion

We have provided a recursive procedure for testing whether a regular tree language is locally testable.

Our characterization extends to unranked unordered trees. For ordered trees, we believe that tameness together with a property that essentially say that the horizontal monoid is in LT should provide a decision procedure for an intuitive notion of LT over ordered unranked trees. Note that in this setting it is no longer clear whether tameness is decidable or not. We leave this case for future work.

From the minimal deterministic automata defining a regular tree language our procedure yields a multi exponential algorithm. On words this test for LT can be done in polynomial time. Note that testing whether a tree language is tame requires only polynomial time on the minimal deterministic bottom-up tree automata. A better complexity for testing LT could be obtained by exhibiting a nice set of identities for the class of LT. This is left for future work.

References

[Boj07a] Bojańczyk, M.: A new algorithm for testing if a regular language is locally threshold testable. Inf. Process. Lett. 104(3), 91–94 (2007)

[Boj07b] Bojańczyk, M.: Two-way unary temporal logic over trees. In: IEEE Symposium on Logic in Computer Science (LICS), pp. 121–130 (2007)

[BP89] Beauquier, D., Pin, J.-E.: Factors of words. In: Ronchi Della Rocca, S., Ausiello, G., Dezani-Ciancaglini, M. (eds.) ICALP 1989. LNCS, vol. 372, pp. 63–79. Springer, Heidelberg (1989)

[BS73] Brzozowski, J.A., Simon, I.: Characterizations of locally testable languages. Discrete Math. 4, 243–271 (1973)

[BS08] Bojańczyk, M., Segoufin, L.: Tree languages defined in first-order logic with one quantifier alternation. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 233–245. Springer, Heidelberg (2008)

[BS09] Benedikt, M., Segoufin, L.: Regular languages definable in FO and FOmod. In: ACM Trans. of Computational Logic (to appear, 2009)

[BSS08] Bojańczyk, M., Segoufin, L., Straubing, H.: Piecewise testable tree languages. In: IEEE Symposium on Logic in Computer Science (LICS) (2008)

[BW06] Bojańczyk, M., Walukiewicz, I.: Characterizing ef and ex tree logics. Theoretical Computer Science 358, 255–272 (2006)

[BW07] Bojańczyk, M., Walukiewicz, I.: Forest algebras. In: Automata and Logic: History and Perspectives, pp. 107–132. Amsterdam University Press (2007)

[EW05] Esik, Z., Weil, P.: Algebraic characterization of regular tree languages. Theoretical Computer Science 340, 291–321 (2005)

[McN74] McNaughton, R.: Algebraic decision procedures for local testability. Math. Syst. Theor. 8, 60–76 (1974)

[Pla08] Place, T.: Characterization of logics over ranked tree languages. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 401–415. Springer, Heidelberg (2008)

[Str85] Straubing, H.: Finite semigroup varieties of the form V*D. Journal of Pure and Applied Algebra 36, 53–94 (1985)

[Til87] Tilson, B.: Categories as algebra: an essential ingredient in the theory of monoids. J. Pure Appl. Algebra 48, 83–198 (1987)

[TW85] Thérien, D., Weiss, A.: Graph congruences and wreath products. J. Pure and Applied Algebra 36, 205–215 (1985)

[Wil96] Wilke, T.: An algebraic characterization of frontier testable tree languages. Theoretical Computer Science 154(1), 85–106 (1996)

The Complexity of Nash Equilibria in Simple Stochastic Multiplayer Games*

Michael Ummels[1] and Dominik Wojtczak[2,3]

[1] RWTH Aachen University, Germany
ummels@logic.rwth-aachen.de
[2] CWI, Amsterdam, The Netherlands
d.k.wojtczak@cwi.nl
[3] University of Edinburgh, UK

Abstract. We analyse the computational complexity of finding Nash equilibria in simple stochastic multiplayer games. We show that restricting the search space to equilibria whose payoffs fall into a certain interval may lead to undecidability. In particular, we prove that the following problem is undecidable: Given a game $\mathcal{G}$, does there exist a pure-strategy Nash equilibrium of $\mathcal{G}$ where player 0 wins with probability 1. Moreover, this problem remains undecidable if it is restricted to strategies with (unbounded) finite memory. However, if mixed strategies are allowed, decidability remains an open problem. One way to obtain a provably decidable variant of the problem is to restrict the strategies to be positional or stationary. For the complexity of these two problems, we obtain a common lower bound of NP and upper bounds of NP and PSPACE respectively.

1 Introduction

We study *stochastic games* [18] played by multiple players on a finite, directed graph. Intuitively, a play of such a game evolves by moving a token along edges of the graph: Each vertex of the graph is either controlled by one of the players, or it is *stochastic*. Whenever the token arrives at a non-stochastic vertex, the player who controls this vertex must move the token to a successor vertex; when the token arrives at a stochastic vertex, a fixed probability distribution determines the next vertex. The play ends when it reaches a terminal vertex, in which case each player receives a payoff. In the simplest case, which we discuss here, the possible payoffs of a single play are just 0 and 1 (i.e. each player either wins or loses a given play). However, due to the presence of stochastic vertices, a player's *expected payoff* (i.e. her probability of winning) can be an arbitrary probability.

Stochastic games have been successfully applied in the verification and synthesis of reactive systems under the influence of random events. Such a system is usually modelled as a game between the system and its environment, where the environment's objective is the complement of the system's objective: the environment is considered hostile. Therefore, traditionally, the research in this area

* This research was supported by the DFG Research Training Group 1298 (AlgoSyn).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 297–308, 2009.

has concentrated on two-player games where each play is won by precisely one of the two players, so-called *two-player, zero-sum games*. However, the system may comprise of several components with independent objectives, a situation which is naturally modelled by a multiplayer game.

The most common interpretation of rational behaviour in multiplayer games is captured by the notion of a *Nash equilibrium* [17]. In a Nash equilibrium, no player can improve her payoff by unilaterally switching to a different strategy. Chatterjee & al. [6] showed that any simple stochastic multiplayer game has a Nash equilibrium, and they also gave an algorithm for computing one. We argue that this is not satisfactory. Indeed, it can be shown that their algorithm may compute an equilibrium where all players lose almost surely (i.e. receive expected payoff 0), while there exist other equilibria where all players win almost surely (i.e. receive expected payoff 1).

In applications, one might look for an equilibrium where as many players as possible win almost surely or where it is guaranteed that the expected payoff of the equilibrium falls into a certain interval. Formulated as a decision problem, we want to know, given a k-player game $\mathcal{G}$ with initial vertex v_0 and two thresholds $\overline{x}, \overline{y} \in [0,1]^k$, whether $(\mathcal{G}, v_0)$ has a Nash equilibrium with expected payoff at least $\overline{x}$ and at most $\overline{y}$. This problem, which we call NE for short, is a generalisation of Condon's *SSG Problem* [8] asking whether in a two-player, zero-sum game one of the two players, say player 0, has a strategy to win the game with probability at least $\frac{1}{2}$.

Our main result is that NE is undecidable if only pure strategies are considered. In fact, even the following, presumably simpler, problem is undecidable: Given a game $\mathcal{G}$, decide whether there exists a pure Nash equilibrium where player 0 wins almost surely. Moreover, the problem remains undecidable if one restricts to pure strategies that use (unbounded) finite memory. However, for the general case of arbitrary mixed strategies, decidability remains an open problem. If one restricts to simpler types of strategies like stationary ones, the problem becomes provably decidable. In particular, for positional (i.e. pure, stationary) strategies the problem becomes NP-complete, and for arbitrary stationary strategies the problem is NP-hard but contained in PSPACE. We also relate the complexity of the latter problem to the complexity of the infamous *Square Root Sum Problem* (SqrtSum) by providing a polynomial-time reduction from SqrtSum to NE with the restriction to stationary strategies.

Let us remark that our game model is rather restrictive: First, players receive a payoff only at terminal vertices. In the literature, a plethora of game models with more complicated modes of winning have been discussed. In particular, the model of a *stochastic parity game* [5,24] has been investigated thoroughly. Second, our model is *turn-based* (i.e. for every non-stochastic vertex there is only one player who controls this vertex) as opposed to *concurrent* [12,11]. The reason that we have chosen to analyse such a restrictive model is that we are focussing on negative results. Indeed, all our lower bounds hold for (multiplayer versions of) the aforementioned models. Moreover, besides Nash equilibria, our negative results apply to several other solution concepts like subgame perfect equilibria [20,21] and secure equilibria [4].

For games with *rewards* on transitions [15], the situation might be different: While our lower bounds can be applied to games with rewards under the *average reward* or the *total expected reward* criterion, we leave it as an open question whether this remains true in the case of discounted rewards.

Due to space constraints, most proofs are either only sketched or omitted entirely. For the complete proofs, see [23].

Related Work. Determining the complexity of Nash Equilibria has attracted much interest in recent years. In particular, a series of papers culminated in the result that computing a Nash equilibrium of a two-player game in strategic form is complete for the complexity class PPAD [10,7]. More in the spirit of our work, Conitzer and Sandholm [9] showed that deciding whether there exists a Nash equilibrium in a two-player game in strategic form where player 0 receives payoff at least x and related decision problems are all NP-hard. For infinite games without stochastic vertices, (a qualitative version of) the problem NE was studied in [22]. In particular, it was shown that the problem is NP-complete for games with parity winning conditions and even in P for games with Büchi winning conditions.

For stochastic games, most results concern the classical SSG problem: Condon showed that the problem is in NP $\cap$ co-NP [8], but it is not known to be in P. We are only aware of two results that are closely related to our problem: First, Etessami & al. [13] investigated Markov decision processes with, e.g., multiple reachability objectives. Such a system can be viewed as a stochastic multiplayer game where all non-stochastic vertices are controlled by one single player. Under this interpretation, one of their results states that NE is decidable in polynomial time for such games. Second, Chatterjee & al. [6] showed that the problem of deciding whether a (concurrent) stochastic game with reachability objectives has a positional-strategy Nash equilibrium with payoff at least $\overline{x}$ is NP-complete. We sharpen their hardness result by showing that the problem remains NP-hard when it is restricted to games with only three players (as opposed to an unbounded number of players) where, additionally, payoffs are assigned at terminal vertices only (cf. Theorem 5).

2 Simple Stochastic Multiplayer Games

The model of a *(two-player, zero-sum) simple stochastic game* [8] easily generalises to the multiplayer case: Formally, a *simple stochastic multiplayer game (SSMG)* is a tuple $\mathcal{G} = (\Pi, V, (V_i)_{i\in\Pi}, \Delta, (F_i)_{i\in\Pi})$ such that:

- Π is a finite set of *players* (usually $\Pi = \{0, 1, \ldots, k-1\}$);
- V is a finite, non-empty set of *vertices*;
- $V_i \subseteq V$ and $V_i \cap V_j = \emptyset$ for each $i \neq j \in \Pi$;
- $\Delta \subseteq V \times ([0,1] \cup \{\bot\}) \times V$ is the *transition relation*;
- $F_i \subseteq V$ for each $i \in \Pi$.

We call a vertex $v \in V_i$ *controlled by player* i and a vertex that is not contained in any of the sets V_i a *stochastic* vertex. We require that a transition is labelled by a probability iff it originates in a stochastic vertex: If $(v, p, w) \in \Delta$ then $p \in [0, 1]$ if v is a stochastic vertex and $p = \bot$ if $v \in V_i$ for some $i \in \Pi$. Moreover, for each pair of a stochastic vertex v and an arbitrary vertex w, we require that there exists precisely one $p \in [0, 1]$ such that $(v, p, w) \in \Delta$. For computational purposes, we require additionally that all these probabilities are rational.

For a given vertex $v \in V$, we denote the set of all $w \in V$ such that there exists $p \in (0, 1] \cup \{\bot\}$ with $(v, p, w) \in \Delta$ by $v\Delta$. For technical reasons, we require that $v\Delta \neq \emptyset$ for all $v \in V$. Moreover, for each stochastic vertex v, the outgoing probabilities must sum up to 1: $\sum_{(p,w):(v,p,w)\in\Delta} p = 1$. Finally, we require that each vertex v that lies in one of the sets F_i is a *terminal (sink) vertex*: $v\Delta = \{v\}$. So if F is the set of all terminal vertices, then $F_i \subseteq F$ for each $i \in \Pi$.

A *(mixed) strategy of player* i *in* $\mathcal{G}$ is a mapping $\sigma : V^*V_i \to \mathcal{D}(V)$ assigning to each possible *history* $xv \in V^*V_i$ of vertices ending in a vertex controlled by player i a (discrete) probability distribution over V such that $\sigma(xv)(w) > 0$ only if $(v, \bot, w) \in \Delta$. Instead of $\sigma(xv)(w)$, we usually write $\sigma(w \mid xv)$. A *(mixed) strategy profile of* $\mathcal{G}$ is a tuple $\overline{\sigma} = (\sigma_i)_{i\in\Pi}$ where σ_i is a strategy of player i in $\mathcal{G}$. Given a strategy profile $\overline{\sigma} = (\sigma_j)_{j\in\Pi}$ and a strategy τ of player i, we denote by $(\overline{\sigma}_{-i}, \tau)$ the strategy profile resulting from $\overline{\sigma}$ by replacing σ_i with τ.

A strategy σ of player i is called *pure* if for each $xv \in V^*V_i$ there exists $w \in v\Delta$ with $\sigma(w \mid xv) = 1$. Note that a pure strategy of player i can be identified with a function $\sigma : V^*V_i \to V$. A strategy profile $\overline{\sigma} = (\sigma_i)_{i\in\Pi}$ is called *pure* if each σ_i is pure.

A strategy σ of player i in $\mathcal{G}$ is called *stationary* if σ depends only on the current vertex: $\sigma(xv) = \sigma(v)$ for all $xv \in V^*V_i$. Hence, a stationary strategy of player i can be identified with a function $\sigma : V_i \to \mathcal{D}(V)$. A strategy profile $\overline{\sigma} = (\sigma_i)_{i\in\Pi}$ of $\mathcal{G}$ is called *stationary* if each σ_i is stationary.

We call a pure, stationary strategy a *positional strategy* and a strategy profile consisting of positional strategies only a *positional strategy profile*. Clearly, a positional strategy of player i can be identified with a function $\sigma : V_i \to V$. More generally, a pure strategy σ is called *finite-state* if it can be implemented by a finite automaton with output or, equivalently, if the equivalence relation $\sim \subseteq V^* \times V^*$ defined by $x \sim y$ if $\sigma(xz) = \sigma(yz)$ for all $z \in V^*V_i$ has only finitely many equivalence classes.[1] Finally, a *finite-state strategy profile* is a profile consisting of finite-state strategies only.

It is sometimes convenient to designate an initial vertex $v_0 \in V$ of the game. We call the tuple $(\mathcal{G}, v_0)$ an *initialised SSMG*. A strategy (strategy profile) of $(\mathcal{G}, v_0)$ is just a strategy (strategy profile) of $\mathcal{G}$. In the following, we will use the abbreviation SSMG also for initialised SSMGs. It should always be clear from the context if the game is initialised or not.

Given an SSMG $(\mathcal{G}, v_0)$ and a strategy profile $\overline{\sigma} = (\sigma_i)_{i\in\Pi}$, the *conditional probability of* $w \in V$ *given the history* $xv \in V^*V$ is the number $\sigma_i(w \mid xv)$ if

[1] In general, this definition is applicable to mixed strategies as well, but for this paper we will identify finite-state strategies with pure finite-state strategies.

$v \in V_i$ and the unique $p \in [0,1]$ such that $(v,p,w) \in \Delta$ if v is a stochastic vertex. We abuse notation and denote this probability by $\overline{\sigma}(w \mid xv)$. The probabilities $\overline{\sigma}(w \mid xv)$ induce a probability measure on the space V^ω in the following way: The probability of a basic open set $v_1 \dots v_k \cdot V^\omega$ is 0 if $v_1 \neq v_0$ and the product of the probabilities $\overline{\sigma}(v_j \mid v_1 \dots v_{j-1})$ for $j = 2, \dots, k$ otherwise. It is a classical result of measure theory that this extends to a unique probability measure assigning a probability to every Borel subset of V^ω, which we denote by $\mathrm{Pr}^{\overline{\sigma}}_{v_0}$.

For a set $U \subseteq V$, let $\mathrm{Reach}(U) := V^* \cdot U \cdot V^\omega$. We are mainly interested in the probabilities $p_i := \mathrm{Pr}^{\overline{\sigma}}_{v_0}(\mathrm{Reach}(F_i))$ of reaching the sets F_i. We call the number p_i the *(expected) payoff of $\overline{\sigma}$ for player i* and the vector $(p_i)_{i \in \Pi}$ the *(expected) payoff of $\overline{\sigma}$.*

Drawing an SSMG. When drawing an SSMG as a graph, we will use the following conventions: The initial vertex is marked by an incoming edge that has no source vertex. Vertices that are controlled by a player are depicted as circles, where the player who controls a vertex is given by the label next to it. Stochastic vertices are depicted as diamonds, where the transition probabilities are given by the labels on its outgoing edges (the default being $\frac{1}{2}$). Finally, terminal vertices are represented by their associated payoff vector. In fact, we allow arbitrary vectors of rational probabilities as payoffs. This does not increase the power of the model since such a payoff vector can easily be realised by an SSMG consisting of stochastic and terminal vertices only.

3 Nash Equilibria

To capture rational behaviour of (selfish) players, John Nash [17] introduced the notion of, what is now called, a *Nash equilibrium.* Formally, given a strategy profile $\overline{\sigma}$, a strategy τ of player i is called a *best response to $\overline{\sigma}$* if τ maximises the expected payoff of player i: $\mathrm{Pr}^{(\overline{\sigma}_{-i},\tau')}_{v_0}(\mathrm{Reach}(F_i)) \leq \mathrm{Pr}^{(\overline{\sigma}_{-i},\tau)}_{v_0}(\mathrm{Reach}(F_i))$ for all strategies τ' of player i. A Nash equilibrium is a strategy profile $\overline{\sigma} = (\sigma_i)_{i \in \Pi}$ such that each σ_i is a best response to $\overline{\sigma}$. Hence, in a Nash equilibrium no player can improve her payoff by (unilaterally) switching to a different strategy.

Previous research on algorithms for finding Nash equilibria in infinite games has focused on computing *some* Nash equilibrium [6]. However, a game may have several Nash equilibria with different payoffs, and one might not be interested in *any* Nash equilibrium but in one whose payoff fulfils certain requirements. For example, one might look for a Nash equilibrium where certain players win almost surely while certain others lose almost surely. This idea leads us to the following decision problem, which we call NE:[2]

> Given an SSMG $(\mathcal{G}, v_0)$ and thresholds $\overline{x}, \overline{y} \in [0,1]^\Pi$, decide whether there exists a Nash equilibrium of $(\mathcal{G}, v_0)$ with payoff $\geq \overline{x}$ and $\leq \overline{y}$.

For computational purposes, we assume that the thresholds $\overline{x}$ and $\overline{y}$ are vectors of rational numbers. A variant of the problem which omits the thresholds just

[2] In the definition of NE, the ordering $\leq$ is applied componentwise.

asks about a Nash equilibrium where some distinguished player, say player 0, wins with probability 1:

> Given an SSMG $(\mathcal{G}, v_0)$, decide whether there exists a Nash equilibrium of $(\mathcal{G}, v_0)$ where player 0 wins almost surely.

Clearly, every instance of the threshold-free variant can easily be turned into an instance of NE (by adding the thresholds $\overline{x} = (1, 0, \ldots, 0)$ and $\overline{y} = (1, \ldots, 1)$). Hence, NE is, a priori, more general than its threshold-free variant.

Our main concern in this paper are variants of NE where we restrict the type of strategies that are allowed in the definition of the problem: Let PureNE, FinNE, StatNE and PosNE be the problems that arise from NE by restricting the desired Nash equilibrium to consist of pure strategies, finite-state strategies, stationary strategies and positional strategies, respectively. In the rest of this paper, we are going to prove upper and lower bounds on the complexity of these problems, where all lower bounds hold for the threshold-free variants, too.

Our first observation is that neither stationary nor pure strategies are sufficient to implement any Nash equilibrium, even if we are only interested in whether a player wins or loses almost surely in the Nash equilibrium. Together with a result from Sect. 5 (namely Proposition 8), this demonstrates that the problems NE, PureNE, FinNE, StatNE, and PosNE are pairwise distinct problems, which have to be analysed separately.

Proposition 1. *There exists an SSMG that has a finite-state Nash equilibrium where player 0 wins almost surely but that has no stationary Nash equilibrium where player 0 wins with positive probability.*

Proof. Consider the game $\mathcal{G}$ depicted in Fig. 1 (a) played by three players 0, 1 and 2 (with payoffs in this order). Obviously, the following finite-state strategy profile is a Nash equilibrium where player 0 wins almost surely: Player 1 plays from vertex v_2 to vertex v_3 at the first visit of v_2 but leaves the game immediately (by playing to the neighbouring terminal vertex) at all subsequent visits to v_2; from vertex v_0 player 1 plays to v_1; player 2 plays from vertex v_3 to vertex v_4 at the first visit of v_3 but leaves the game immediately at all subsequent visits to v_3; from vertex v_1 player 2 plays to v_2. It remains to show that there is no stationary Nash equilibrium of $(\mathcal{G}, v_0)$ where player 0 wins with positive probability: Any such equilibrium induces a stationary Nash equilibrium of $(\mathcal{G}, v_2)$ where both players 1 and 2 receive payoff at least $\frac{1}{2}$ since otherwise one of these players could improve her payoff by changing her strategy at v_0 or v_1. However, it is easy to see that in any stationary Nash equilibrium of $(\mathcal{G}, v_2)$ either player 1 or player 2 receives payoff 0. □

Proposition 2. *There exists an SSMG that has a stationary Nash equilibrium where player 0 wins almost surely but that has no pure Nash equilibrium where player 0 wins with positive probability.*

Proof. Consider the game depicted in Fig. 1 (b) played by three players 0, 1 and 2 (with payoffs given in this order). Clearly, the stationary strategy profile

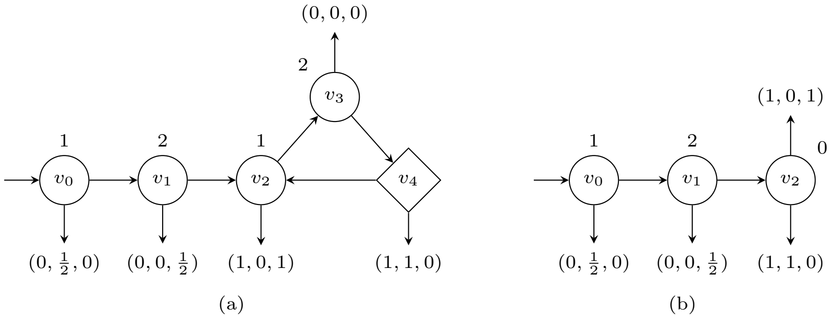


Fig. 1. Two SSMGs with three players

where from vertex v_2 player 0 selects both outgoing edges with probability $\frac{1}{2}$ each, player 1 plays from v_0 to v_1 and player 2 plays from v_1 to v_2 is a Nash equilibrium where player 0 wins almost surely. However, for any pure strategy profile where player 0 wins almost surely, either player 1 or player 2 receives payoff 0 and could improve her payoff by switching her strategy at v_0 or v_1 respectively. □

4 Decidable Variants of NE

In this section, we show that the problems PosNE and StatNE are decidable and analyse their complexity.

Theorem 3. *PosNE is in* NP*.*

Proof (Sketch). Let $(\mathcal{G}, v_0)$ be an SSMG. Any positional strategy profile of $\mathcal{G}$ can be identified with a mapping $\overline{\sigma} : \bigcup_{i \in \Pi} V_i \to V$ such that $(v, \bot, \overline{\sigma}(v)) \in \Delta$ for each non-stochastic vertex v, an object whose size is linear in the size of $\mathcal{G}$. To prove that PosNE is in NP, it suffices to show that we can check in polynomial time whether such a mapping $\overline{\sigma}$ constitutes a Nash equilibrium whose payoff lies in between the given thresholds $\overline{x}$ and $\overline{y}$. This can be done by computing, for each player i, 1. the payoff z_i of $\overline{\sigma}$ for player i and 2. the maximal payoff $r_i = \sup_\tau \Pr_{v_0}^{(\overline{\sigma}_{-i}, \tau)}(\text{Reach}(F_i))$ that player i can achieve when playing against $\overline{\sigma}_{-i}$, and then to check whether $x_i \leq z_i \leq y_i$ and $r_i \leq z_i$. It follows from results on *Markov chains* and *Markov decision processes* that both these numbers can be computed in polynomial time (via linear programming). □

To prove the decidability of StatNE, we appeal to results established for the *Existential Theory of the Reals*, ExTh($\mathfrak{R}$), the set of all existential first-order sentences (over the appropriate signature) that hold in $\mathfrak{R} := (\mathbb{R}, +, \cdot, 0, 1, \leq)$. The best known upper bound for the complexity of the associated decision problem is PSpace [3,19], which leads to the following theorem.

Theorem 4. *StatNE is in* PSpace*.*

Proof (Sketch). Since PSpace = NPSpace, it suffices to give a nondeterministic polynomial-space algorithm for StatNE. On input $\mathcal{G}, v_0, \overline{x}, \overline{y}$, the algorithm starts by guessing a set $S \subseteq V \times V$ and proceeds by computing (in polynomial time), for each player i, the set R_i of vertices from where the set F_i is reachable in the graph $G = (V, S)$. Finally, the algorithm evaluates a certain existential first-order sentence ψ, which can be computed in polynomial time from $(\mathcal{G}, v_0)$, $\overline{x}$, $\overline{y}$, S and $(R_i)_{i \in \Pi}$, over $\mathfrak{R}$ and returns the answer to this query. The sentence ψ states that there exists a stationary Nash equilibrium $\overline{\sigma}$ of $(\mathcal{G}, v_0)$ with payoff $\geq \overline{x}$ and $\leq \overline{y}$ whose *support* is S, i.e. $S = \{(v, w) \in V \times V : \overline{\sigma}(w \mid v) > 0\}$. □

Having shown that PosNE and StatNE are in NP and PSpace respectively, the natural question arises whether there is a polynomial-time algorithm for PosNE or StatNE. The following theorem implies that this is not the case (unless, of course, P = NP) since both problems are NP-hard. Moreover, both problems are already NP-hard for games with only two players.

Theorem 5. *PosNE and StatNE are* NP*-hard, even for games with only two players (three players for the threshold-free variants).*

It follows from Theorems 3 and 5 that PosNE is NP-complete. For StatNE, we have provided an NP lower bound and a PSpace upper bound, but the exact complexity of the problem remains unclear. Towards gaining more insight into the problem StatNE, we relate its complexity to the complexity of the *Square Root Sum Problem* (SqrtSum), the problem of deciding, given numbers $d_1, \ldots, d_n, k \in \mathbb{N}$, whether $\sum_{i=1}^{n} \sqrt{d_i} \geq k$. Recently, it was shown that SqrtSum belongs to the 4th level of the *counting hierarchy* [1], which is a slight improvement over the previously known PSpace upper bound. However, it is an open question since the 1970s whether SqrtSum falls into the polynomial hierarchy [16,14]. We identify a polynomial-time reduction from SqrtSum to StatNE.[3] Hence, StatNE is at least as hard as SqrtSum, and showing that StatNE resides inside the polynomial hierarchy would imply a major breakthrough in understanding the complexity of numerical computation.

Theorem 6. *SqrtSum is polynomial-time reducible to StatNE.*

5 Undecidable Variants of NE

In this section, we argue that the problems PureNE and FinNE are undecidable by exhibiting reductions from two undecidable problems about *two-counter machines*. Our construction is inspired by a construction used by Brázdil & al. [2] to prove the undecidability of stochastic games with branching-time winning conditions.

A two-counter machine $\mathcal{M}$ is given by a list of instructions $\iota_1, \ldots, \iota_m$ where each instruction is one of the following:

[3] Some authors define SqrtSum with $\leq$ instead of $\geq$. With this definition, we would reduce from the complement of SqrtSum instead.

- "inc(j); goto k" (increment counter j by 1 and go to instruction number k);
- "zero(j) ? goto k : dec(j); goto l" (if the value of counter j is zero, go to instruction number k; otherwise, decrement counter j by one and go to instruction number l);
- "halt" (stop the computation).

Here j ranges over $1, 2$ (the two counters), and $k \neq l$ range over $1, \ldots, m$. A configuration of $\mathcal{M}$ is a triple $C = (i, c_1, c_2) \in \{1, \ldots, m\} \times \mathbb{N} \times \mathbb{N}$, where i denotes the number of the current instruction and c_j denotes the current value of counter j. A configuration C' is the *successor* of configuration C, denoted by $C \vdash C'$, if it results from C by executing instruction ι_i; a configuration $C = (i, c_1, c_2)$ with $\iota_i =$ "halt" has no successor configuration. Finally, the *computation of* $\mathcal{M}$ is the unique maximal sequence $\rho = \rho(0)\rho(1)\ldots$ such that $\rho(0) \vdash \rho(1) \vdash \ldots$ and $\rho(0) = (1, 0, 0)$ (the *initial configuration*). Note that ρ is either infinite, or it ends in a configuration $C = (i, c_1, c_2)$ such that $\iota_i =$ "halt".

The *halting problem* is to decide, given a machine $\mathcal{M}$, whether the computation of $\mathcal{M}$ is finite. It is well-known that two-counter machines are Turing powerful, which makes the halting problem and its dual, the *non-halting problem*, undecidable.

Theorem 7. *PureNE is undecidable.*

Proof (Sketch). The proof is by a reduction from the non-halting problem to PureNE: we show that one can compute from a two-counter machine $\mathcal{M}$ an SSMG $(\mathcal{G}, v_0)$ with nine players such that the computation of $\mathcal{M}$ is infinite iff $(\mathcal{G}, v_0)$ has a pure Nash equilibrium where player 0 wins almost surely.

Any pure strategy profile $\overline{\sigma}$ of $(\mathcal{G}, v_0)$ where player 0 wins almost surely determines an infinite sequence ρ of *pseudo configurations* of $\mathcal{M}$ (where the counters may take the value ω). Of course, in general, ρ is not the computation of $\mathcal{M}$. However, the game $\mathcal{G}$ is constructed in such a way that $\overline{\sigma}$ is a Nash equilibrium iff ρ is the computation of $\mathcal{M}$. Since ρ is infinite, this equivalence implies that $\mathcal{G}$ has a pure Nash equilibrium where player 0 wins almost surely iff $\mathcal{M}$ does not halt.

To get a flavour of the full proof, let us consider a machine $\mathcal{M}$ that contains the instruction "inc(1); goto k". An abstraction of the corresponding part of the game $\mathcal{G}$ is depicted in Fig. 2: the game is restricted to three players 0, A and B (with payoffs given in this order), and some irrelevant vertices have been removed.

In the following, let $\overline{\sigma}$ be a pure strategy profile of $(\mathcal{G}, v)$ where player 0 wins almost surely. At both vertices u and u', player 0 can play to either a grey or a white vertex; if she plays to a grey vertex, then with probability $\frac{1}{2}$ the play returns to u or u' respectively; if she plays to the white vertex, the play never returns to u or u'. Let c and c' denote the maximal number of visits to the grey vertex connected to u and u' respectively (the number being ω if player 0 always plays to the grey vertex): these two ordinal numbers represent the counter value before and after executing the instruction.

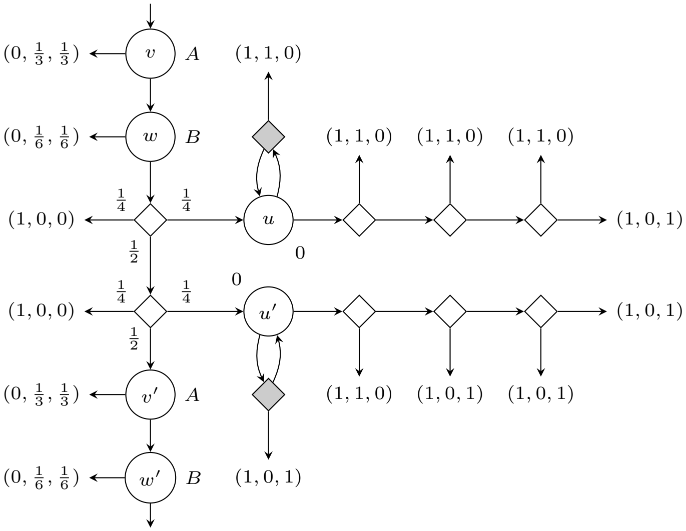


Fig. 2. Incrementing a counter

To see why $c' = 1 + c$ if $\overline{\sigma}$ is a Nash equilibrium, consider the probabilities $a := \Pr_v^{\overline{\sigma}}(\text{Reach}(F_A))$ and $b := \Pr_v^{\overline{\sigma}}(\text{Reach}(F_B))$. We have $a \geq \frac{1}{3}$ and $b \geq \frac{1}{6}$ since otherwise player A or B could improve by changing her strategy at vertex v or w respectively. In fact, the construction of $\mathcal{G}$ ensures that $a + b = \frac{1}{2}$; hence, $a = \frac{1}{3}$. Moreover, the same argumentation proves that $a' := \Pr_{v'}^{\overline{\sigma}}(\text{Reach}(F_A)) = \frac{1}{3}$.

Let $p := \Pr_v^{\overline{\sigma}}(\text{Reach}(F_A) \mid V^\omega \setminus v \ldots v' \cdot V^\omega)$ be the conditional probability that player A wins given that v' is not reached; then $a = p + \frac{1}{4} \cdot a'$ and consequently $p = \frac{1}{4}$. But p can also be written as the following sum of two binary numbers:

$$0.00\underbrace{1\ldots1}_{c\text{ times}}111 + 0.000\underbrace{0\ldots0}_{c'\text{ times}}100\,.$$

Obviously, this sum is equal to $\frac{1}{4}$ iff $c' = 1 + c$. □

It follows from the proof of Theorem 7 that Nash equilibria may require infinite memory (even if we are only interested in whether a player wins with probability 0 or 1). More precisely, we have the following proposition.

Proposition 8. *There exists an SSMG that has a pure Nash equilibrium where player 0 wins almost surely but that has no finite-state Nash equilibrium where player 0 wins with positive probability.*

Proof. Consider the game $(\mathcal{G}, v_0)$ constructed in the proof of Theorem 7 for the machine $\mathcal{M}$ consisting of the single instruction "inc(1); goto 1". We modify this game by adding a new initial vertex v_1 which is controlled by a new player, player 1, and from where she can either move to v_0 or to a new terminal vertex where she receives payoff 1 and every other player receives payoff 0. Additionally, player 1 wins at every terminal vertex of the game $\mathcal{G}$ that is winning for player 0. Let us denote the modified game by $\mathcal{G}'$.

Since the computation of $\mathcal{M}$ is infinite, the game $(\mathcal{G}, v_0)$ has a pure Nash equilibrium where player 0 wins almost surely. This equilibrium induces a pure Nash equilibrium of $(\mathcal{G}', v_1)$ where player 0 wins almost surely. However, it is easy to see that there is no finite-state Nash equilibrium of $(\mathcal{G}, v_0)$ where player 0 (or player 1) wins almost surely. Consequently, in any finite-state Nash equilibrium of $(\mathcal{G}', v_1)$ player 1 will play from v_1 to the new terminal vertex, giving player 0 a payoff of 0. □

It follows from Proposition 8 that the problems FinNE and PureNE are distinct. Nevertheless, we can show that FinNE is undecidable, too. Note however that FinNE is recursively enumerable: To decide whether an SSMG $(\mathcal{G}, v_0)$ has a finite-state Nash equilibrium with payoff $\geq \overline{x}$ and $\leq \overline{y}$, one can just enumerate all possible finite-state profiles and check for each of them whether the profile is a Nash equilibrium with the desired properties (by analysing the finite Markov chain that is generated by this profile).

Theorem 9. *FinNE is undecidable.*

Proof (Sketch). The proof is by a reduction from the halting problem for two-counter machines and similar to the proof of Theorem 7. □

6 Conclusion

We have analysed the complexity of deciding whether a simple stochastic multiplayer game has a Nash equilibrium whose payoff falls into a certain interval. Our results demonstrate that the presence of both stochastic vertices and more than two players makes the problem much more complicated than when one of these factors is absent. In particular, the problem of deciding the existence of a pure-strategy Nash equilibrium where player 0 wins almost surely is undecidable for simple stochastic multiplayer games, whereas it is contained in NP$\cap$co-NP for two-player, zero-sum simple stochastic games [8] and even in P for non-stochastic infinite multiplayer games with, e.g., Büchi winning conditions [22].

Apart from settling the complexity of NE when arbitrary mixed strategies are considered, future research may, for example, investigate restrictions of NE to games with a small number of players. In particular, we conjecture that the problem is decidable for two-player games, even if these are not zero-sum.

References

1. Allender, E., Bürgisser, P., Kjeldgaard-Pedersen, J., Miltersen, P.B.: On the complexity of numerical analysis. In: Proc. CCC 2006, pp. 331–339. IEEE Computer Society Press, Los Alamitos (2006)
2. Brázdil, T., Brožek, V., Forejt, V., Kučera, A.: Stochastic games with branching-time winning objectives. In: Proc. LICS 2006, pp. 349–358. IEEE Computer Society Press, Los Alamitos (2006)

3. Canny, J.: Some algebraic and geometric computations in PSPACE. In: Proc. STOC 1988, pp. 460–469. ACM Press, New York (1988)
4. Chatterjee, K., Henzinger, T.A., Jurdziński, M.: Games with secure equilibria. Theoretical Computer Science 365(1-2), 67–82 (2006)
5. Chatterjee, K., Jurdziński, M., Henzinger, T.A.: Quantitative stochastic parity games. In: Proc. SODA 2004, pp. 121–130. ACM Press, New York (2004)
6. Chatterjee, K., Majumdar, R., Jurdziński, M.: On Nash equilibria in stochastic games. In: Marcinkowski, J., Tarlecki, A. (eds.) CSL 2004. LNCS, vol. 3210, pp. 26–40. Springer, Heidelberg (2004)
7. Chen, X., Deng, X.: Settling the complexity of two-player Nash equilibrium. In: Proc. FOCS 2006, pp. 261–272. IEEE Computer Society Press, Los Alamitos (2006)
8. Condon, A.: The complexity of stochastic games. Information and Computation 96(2), 203–224 (1992)
9. Conitzer, V., Sandholm, T.: Complexity results about Nash equilibria. In: Proc. IJCAI 2003, pp. 765–771. Morgan Kaufmann, San Francisco (2003)
10. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. In: Proc. STOC 2006, pp. 71–78. ACM Press, New York (2006)
11. de Alfaro, L., Henzinger, T.A.: Concurrent omega-regular games. In: Proc. LICS 2000, pp. 141–154. IEEE Computer Society Press, Los Alamitos (2000)
12. de Alfaro, L., Henzinger, T.A., Kupferman, O.: Concurrent reachability games. In: Proc. FOCS 1998, pp. 564–575. IEEE Computer Society Press, Los Alamitos (1998)
13. Etessami, K., Kwiatkowska, M.Z., Vardi, M.Y., Yannakakis, M.: Multi-objective model checking of Markov decision processes. Logical Methods in Computer Science 4(4) (2008)
14. Etessami, K., Yannakakis, M.: On the complexity of Nash equilibria and other fixed points. In: Proc. FOCS 2007, pp. 113–123. IEEE Computer Society Press, Los Alamitos (2007)
15. Filar, J., Vrieze, K.: Competitive Markov decision processes. Springer, Heidelberg (1997)
16. Garey, M.R., Graham, R.L., Johnson, D.S.: Some NP-complete geometric problems. In: Proc. STOC 1976, pp. 10–22. ACM Press, New York (1976)
17. Nash Jr., J.F.: Equilibrium points in N-person games. Proc. National Academy of Sciences of the USA 36, 48–49 (1950)
18. Neyman, A., Sorin, S. (eds.): Stochastic Games and Applications. NATO Science Series C, vol. 570. Springer, Heidelberg (1999)
19. Renegar, J.: On the computational complexity and geometry of the first-order theory of the reals. Journal of Symbolic Computation 13(3), 255–352 (1992)
20. Selten, R.: Spieltheoretische Behandlung eines Oligopolmodells mit Nachfrageträgheit. Zeitschrift für die gesamte Staatswissenschaft 121, 301–324, 667–689 (1965)
21. Ummels, M.: Rational behaviour and strategy construction in infinite multiplayer games. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 212–223. Springer, Heidelberg (2006)
22. Ummels, M.: The complexity of Nash equilibria in infinite multiplayer games. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 20–34. Springer, Heidelberg (2008)
23. Ummels, M., Wojtczak, D.: The complexity of Nash equilibria in simple stochastic multiplayer games. Technical report, University of Edinburgh (2009)
24. Zielonka, W.: Perfect-information stochastic parity games. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 499–513. Springer, Heidelberg (2004)

Google's Auction for TV Ads

Noam Nisan[1,4], Jason Bayer[2], Deepak Chandra[2], Tal Franji[1], Robert Gardner[3], Yossi Matias[1], Neil Rhodes[3], Misha Seltzer[1], Danny Tom[2], Hal Varian[2], and Dan Zigmond[2]

[1] Google, Tel-Aviv
[2] Google, Mountain View
[3] Google, Irvine
[4] Hebrew University of Jerusalem

Abstract. This document describes the auction system used by Google for allocation and pricing of TV ads. It is based on a simultaneous ascending auction, and has been in use since September 2008.

1 Introduction

While Google is known for its advertising on the web, not many people know that it also allows advertisers to buy TV ads, and do so in a convenient way online. At the time of writing, Google offers TV advertising inventory on over 100 channels in the USA, some national and some local. While this paper will not elaborate on the advantages that this offering can provide to advertisers, broadcasters, cable operators, or Google itself, we will shortly mention the following advantages for advertisers:

- **Automation:** all aspects of of creating, buying, and running the campaign are done via a simple online web interface.
- **Flexibility:** in contrast to the old-fashioned habits of the TV advertising industry where complex deals are manually negotiated long in advance, this system provides a simple, transparent, just-in-time, granular auction model. In particular this allows convenient aggregation of inventory over many small networks.
- **Measurement:** Excellent online measurements and analysis of the campaign are provided. Notably, the TV ads are delivered via set-top boxes that track exact and actual numbers of viewers for each ad.

This document concentrates on the auction mechanism that is used for allocation and pricing of TV ads. We only sketch a high-level description of the whole system. The reader who wishes to learn more about the rest of the system may consult Google's web-sites for TV ads [1], or the adwords "traditional media" blog [2]. A variant of the auction mechanism was also used for allocation of Radio ads from October 2008 to May 2009, at which point Google cancelled its Radio operation.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 309–327, 2009.

1.1 How It Works — Overview

Google has deals in place with various “publishers” of TV content: broadcasters and Cable companies. From the point of view of advertising, these publishers own an inventory of ad-slots: time-slots on various stations, where each of these is (part of) a commercial break within some scheduled program. A typical slot may be between 30 second to 120 seconds long, and there may be many dozens of such slots available daily on each station. Publishers make (part of) their inventory of slots on their various stations available for sale through Google. All day-to-day interaction with publishers is automated: slightly over-simplifying, each day the publisher’s IT systems send the next day’s inventory (set of slots to be sold by Google) to Google; Google then auctions it among interested advertisers; finally, Google sends back the resulting schedule and ads to the publisher’s systems that then insert the scheduled ads at the scheduled commercial breaks. The auction also sets the prices for the advertisers and Google handles their billing.

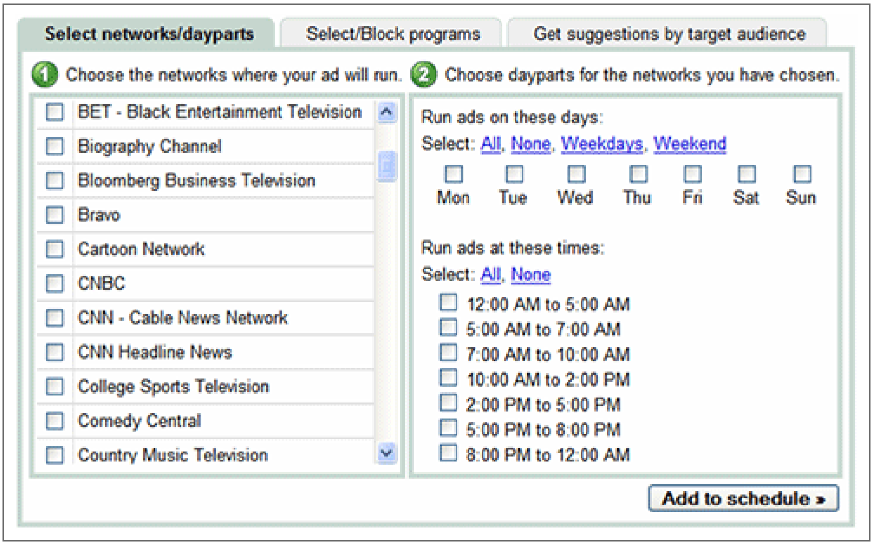
This paper does not discuss the financial arrangements between Google and the different publishers, which are manually negotiated, but rather focuses on the advertiser-facing side which is goverened by an auction.

1.2 The Advertiser-Facing User Interface

An advertiser that wishes to set a TV advertising campaign can do so using the respective section of Google’s Adwords site [1]. There are basically three steps involved. First, the ad itself — the “creative” — a video clip, must be produced in standard format. This is the responsibility of the advertiser, but Google offers an online “ad creation marketplace” which helps connecting an advertiser with specialists that can produce a TV ad for him. Once the ad exists, the advertiser simply uploads it to the site.

The second step is targeting where your ad may appear. This is done using a web-page interface that allows targeting by various criteria: stations, days in the week, day parts, demographics, geographic regions, scheduled programs, etc. Much sophistication went into making this interface convenient and powerful, but logically, the output of this stage is simply the set of slots that the advertiser is interested in. Figure 1 gives a screen shot of (part of) the user interface for this.

The third step is setting the bid: how much the advertiser is willing to pay. Logically, there are two main conceptual parts to this bid: a total budget and a per-ad bid. First, a daily budget is specified, and then a maximum price that the advertiser is willing to pay for his ad to appear in every targeted slot is specified. The latter is commonly given in terms of ”cpm” — cost per Millie — the price for each 1000 viewers. Thus for example a ”$5 cpm” ad that is watched by 3,000 viewers will cost $15. Again, a web-interface lets the advertiser specify these ”max-cpm” bids. While additional constraints and preferences may be specified, logically, the heart of the specified information here is a real value for each targeted slot. Figure 2 gives a screen shot of (part of) the TV-ads user interface for this.



Select networks/dayparts | Select/Block programs | Get suggestions by target audience

1 Choose the networks where your ad will run.

- □ BET - Black Entertainment Television
- □ Biography Channel
- □ Bloomberg Business Television
- □ Bravo
- □ Cartoon Network
- □ CNBC
- □ CNN - Cable News Network
- □ CNN Headline News
- □ College Sports Television
- □ Comedy Central
- □ Country Music Television

2 Choose dayparts for the networks you have chosen.

Run ads on these days:

Select: All, None, Weekdays, Weekend

□ Mon □ Tue □ Wed □ Thu □ Fri □ Sat □ Sun

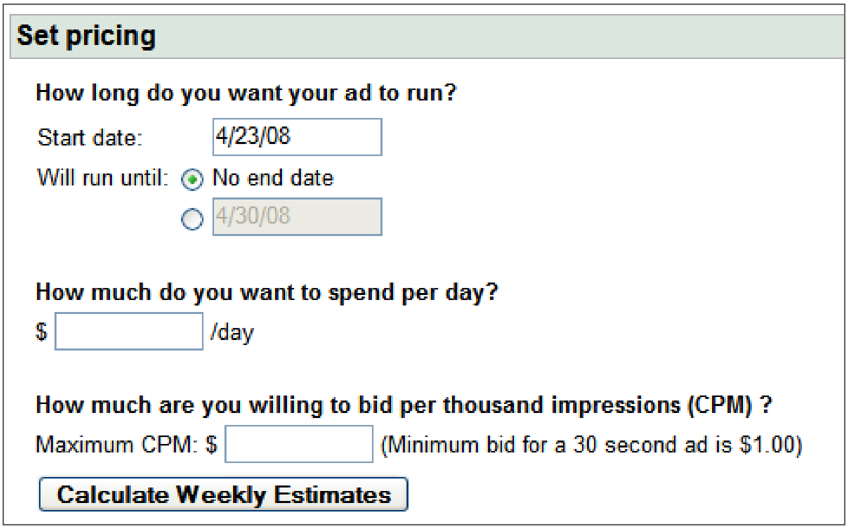
Run ads at these times:

Select: All, None

- □ 12:00 AM to 5:00 AM
- □ 5:00 AM to 7:00 AM
- □ 7:00 AM to 10:00 AM
- □ 10:00 AM to 2:00 PM
- □ 2:00 PM to 5:00 PM
- □ 5:00 PM to 8:00 PM
- □ 8:00 PM to 12:00 AM

Add to schedule »

Fig. 1. Screen shot from TV ads targeting interface



Set pricing

How long do you want your ad to run?

Start date: 4/23/08

Will run until: ● No end date

○ 4/30/08

How much do you want to spend per day?

$ ______ /day

How much are you willing to bid per thousand impressions (CPM) ?

Maximum CPM: $ ______ (Minimum bid for a 30 second ad is $1.00)

Calculate Weekly Estimates

Fig. 2. Screen shot from TV ads bidding interface

1.3 The Auction Goals

Once all the inventory and bids are given, the goal of the auction is to decide on the allocation, i.e. the schedule of ads, as well as the correct pricing. The desired input and output of this auction is clear:

Input:

- The inventory that is available for the next day. This is essentially a set of slots, where each slot is specified by a time-range on a station, as well as its basic parameters.
- The set of all campaigns that are interested in this inventory. The basic information given for each campaign is the daily budget and the cpm for each slot, but there may also be additional constraints and preferences.

Output:

- The schedule: which advertiser gets to air his creative in which slot.
- Pricing: How much is every advertiser charged (in cpm) for each slot that he received.

As opposed to traditional methods in the industry where prices are set by manual negotiations long in advance, the system here allows buying inventory on a daily basis and thus the prices themselves must be determined automatically as manual negotiation is impractical and inconvenient. These prices must be flexible, changing on a day-to-day basis as to reflect the changing market conditions, and thus must be determined by some auction-like (or market-like) mechanism. This flexibility of prices is needed as to ensure basic economic efficiency in the face of changing market conditions. This automatic setting of prices is a major difference from existing systems [3] that automatically handle scheduling of ads, but are given manually defined prices.

The exact criteria according to which the allocation and pricing should be done is slightly subtle. It is clear that we would want to allocate slots to advertisers in a way that maximizes their value from the ads as implied by their bids, and that we can never charge the advertisers more than what is implied by their budget or bid. It is also clear that we would like to maximize the revenue, which is then split between the publishers and Google (according to the negotiated business terms whose details do not concern us here). What is less clear at first sight is the exact desired trade off between the conflicting goals of the different advertisers, between these and the revenue goal, as well as how all this is implemented in a way that encourages advertisers to bid truthfully and not "shade" their bids. This last consideration strongly suggests that we should not charge advertisers directly according to their bid, but rather in the spirit of the "second price auction".

In section 2.2, we study some of the difficulties in even attempting to pose this as an optimization goal (before even worrying about the practicality of the optimization). Our conclusion is to attempt reaching a "market allocation" with minimum equilibrium prices. In such an equilibrium, each ad slot is priced at the minimum price needed for the winner to "take it away" from the competition, and each advertiser is allocated the most cost-effective set of ads under these prices according to his bids. Such an outcome would, in partuclar, be "Pareto-optimal" as well as fair. Taking minimum market equilibrium prices implies that bidders have little strategic incentive to reduce their bids, as in any case they pay the minimum price needed to win their allocation. This approach focuses on setting up an economic environment—a market—that we anticipate will grow in the future. Making sure that the environment provides an efficient and fair outcome is critical for future growth.

1.4 The Auction Logic

The mechanism that was chosen to implement the auction is based on the simultaneous ascending auction with item prices. The theoretical foundations of this ascending approach go back to [4] with a more general point of view taken in [5], and has been famously used in the FCC spectrum auctions [6]. For a background on combinatorial auctions in general and the simultaneous ascending auction in particular we refer the reader to [7,6]. The basic logic of this auction is as follows: each ad-slot has an associated price that keeps increasing throughout the auction. Prices start at low reserve prices, and rise whenever there is "over-demand" for an ad-slot — i.e. a slot that is currently held by one bidder is desired by another. Such small price increases keep going on until there is no "over-demand", at which point the auction closes. The basic step in the auction is the calculation of the "demand" of a bidder at current prices — i.e. which set of slots would this bidder desire to acquire assuming that slots are priced as given. In its basic form the calculation of this demand is done by a greedy algorithm that chooses slots according to decreasing bid-to-price ratio.

Under certain theoretical assumptions ("gross substitutes") it is known that this procedure ends with a "Walrasian market equilibrium" [5] and under even stricter assumptions these prices are incentive compatible — i.e. give no bidder any strategic reason to under-bid [4]. Of course, these theoretical assumptions do not hold in reality, and thus we can not expect these desired properties to perfectly hold in reality. In fact, we show that it is impossible to reach any of these two conditions even under very restricted cases of our basic setting. However, we do find that this general approach does work "well" in practice, with various heuristic solutions to various complications[1].

1.5 The Auction Implementation

The auction system described here has been in operation continuously since September 2008. The complete TV-ads system (which has been in operation longer) is quite sophisticated in terms of architecture involving multiple components that interact with publisher systems, billing systems, databases, monitoring, as well as a web-based front-end. The auction component itself is shielded from this complexity and is quite simple in architecture. The auction is implemented as a single-threaded single-processor program that accepts its input, in

[1] We wish we could quantify this last claim by bringing experimental results, but are not able to do so here. Some of this quantification has been done but is confidential data, some of it has not been completely measured as it requires non-trivial effort, and some of it is even not clear how to measure.

one “chunk”, in the Google-standard open format of a protocol buffer [8], and produces an output in a similar format. The most notable feature of the internal architecture is the central place of a “Bidder” interface that represents a single advertiser to the auction. The auction itself proceeds by repeatedly asking Bidder objects for their “demand”. This internal architecture directly corresponds to the theoretical point of view [9], separates the concerns of each advertiser from those of the auction as a whole, and is very adaptable to new advertiser bidding product features. (The reader may observe the rapid rate of change (product improvements) on the adwords traditional media blog [2].)

The system is considered a success: short-term simulations do show a significant improvement in the quality of allocation (revenue, advertiser value, and other measures) and feedback from the advertiser and publisher directions has been quite positive.

1.6 Rest of the Paper

The rest of this paper describes this auction in detail. We believe that it will be more illuminating to start, in section 2, with a simplified description that captures the basic issues, and only then, in section 3, discuss various “complications” that real life brings. This paper does not present any new theoretical results, it does however attempt to provide a theoretical context to the many issues that were faced and dealt with by the auction. As usual, any real implementation faces multiple complications, many of which are handled in an ad-hoc way but really require new theoretical analysis. We attempt pointing out some of the issues that we believe deserve such theoretical treatment. The first author has in fact collaborated in theoretical analysis on one such topic [10].

2 The Basic Problem

This section discusses the basic version of the auction.

2.1 The Formulation

Let us start by introducing basic notation that captures the essence of the issue:

- We will have m abstract slots, numbered $1...m$. For each slot j we are given a reserve price $r_j \geq 0$, as well as basic slot data (station, time, impressions).
- We will have n bidders. Each bidder is specified by his budget $b_i \geq 0$, and his maximum bid for each slot, specified by a “valuation function” $v_i()$, where $v_i(j) \geq 0$ denotes the bid for slot j. We denote $T_i = \{j|v_i(j) > 0\}$ the set of slots that i targets. From the point of view of the auction, we take v_i as given, with entries that have been already calculated according to the advertiser’s bid as a function of the slot data.
- The allocation produced is a partition of the slots into disjoint subsets $S_0, S_1...S_n \subseteq \{1...m\}$, where each bidder i wins the set of slots S_i, and S_0 are the unallocated slots.

- The pricing produced is a real price p_j for each slot j that satisfy the following properties: (a) at least the reserve price: $p_j \geq r_j$ (b) individual rationality: if $j \in S_i$ then $p_j \leq v_i(j)$ (c) budget constraints: for each bidder i, $\sum_{j \in S_i} p_j \leq b_i$[2].

We first need to model the utilities of the bidders, i.e. what do they desire. While in section 3 we elaborate on additional constraints and preferences that they may express, in the basic formulation described here, we only get from each bidder a bid for each slot. Our basic assumption on the value of a set of items is "additive valuations" *up to the budget limit.* I.e., that the (declared) value that bidder i gets from acquiring a bundle of slots $S \subseteq \{1...m\}$ is simply $v_i(S) = \sum_{j \in S} v_i(j)$. Our assumption is that this is a monetary measure and thus if the bidder pays a total of q for the bundle S then his utility — what we should aim to optimize for him — is $v_i(S) - q$, but this holds only as long as we are within budget $q \leq b_i$. This budget limit takes us out of the usual quasi-linear setting, and is analyzed theoretically in [10].

2.2 What Are the Goals?

Let us start by informally stating the goals that we would like to get, at first not worrying about exact definitions, whether they are feasible, or how to handle the conflicts between them.

- Efficiency: We should try to maximize the values obtained by the bidders, i.e. the vector $v_1(S_1)...v_n(S_n)$. There will clearly be some trade off, which we should specify, between the values obtained by different bidders.
- Revenue: certainly the auctioneer should aim to maximize the revenue, $\sum_{j \notin S_0} p_j$.
- Fairness: We should not discriminate between bidders. The exact meaning of this requires some thought, but lack of fairness is usually quite clear.
- Incentive Compatibility: We should remember that the bid information is given to us by advertisers. These will react strategically to the auction system used, and optimize their bids as to get highest utility from the system. We should ensure that there are no strategic reasons for advertisers to "underbid" or otherwise strategically declare a bid that is different than their true value.

While the reader may certainly see the need for trade offs between these goals, there is even conceptual difficulty in attempting to handle them separately. We encourage the reader to pause for a while and try to formulate for himself what his optimization trade-off approach will be. To our understanding there is

[2] Less specifically, as in mechanism design theory, we could only ask for the total payments from each bidder i without breakdown by "item price", e.g., as given by the VCG payment rule. This relaxation does not seem to really help, and our subsequent discussion regarding formalizing the auction goals applies also to this less specific "bundle price" setting.

no clear mathematical programming formulation of the optimization goal that makes much sense here. The difficulty is inherent in the combination of budget constraints with valuations, as simple attempts to optimize "social welfare" do not take budgets into proper account and simple attempts to maximize revenue do not give reasonable weight to the valuations. This is especially apparent when looking at scenarios where the budget constraints are the significant ones, which is the typical case. In appendix A we use a simple example to discuss why several natural attempts at formalizing the problem do not really make much sense. We also discuss there the problems with auctioning each slot separately, an approach which would be quite appealing due to its simplicity.

We believe that the approach that makes sense is the classic economic goal of reaching a "Walrasian" market equilibrium:

- Unallocated slots remain at reserve price: $j \in S_0$ implies $p_j = r_j$.
- Each advertiser gets his "demand" at the equilibrium prices, i.e. wins the best package for him. Formally, any set S of slots where $\sum_{j \in S} p_j \leq b_i$, we have that $\sum_{j \in S}(v_i(j) - p_j) \leq \sum_{j \in S_i}(v_i(j) - p_j)$.
- Where there is a range of equilibrium prices, we choose the lowest possible equilibrium prices.

Achieving this goal would seem to be natural and desirable in its own right. Let us say a few words on how it addresses our previous informal list of goals.

1. Efficiency: By the first welfare theorem, any equilibrium allocation will be Pareto-optimal. In particular, every bidder gets the bundles of slots that is optimal for him, under given prices, according to his bid.
2. Revenue: This auction does not always maximize revenue among all auctions. However, at a high level, budgets are exhausted for all bidders that bid "high enough", while fairness and incentive constraints limit what can be taken from "low bidders".
3. Fairness: The auction is obviously anonymous, has the "no envy" property, and all prices are justified by the property that at a lower price the slot would be over-demanded.
4. Incentive Compatibility: in general we know that markets are incentive compatible as long as no single participant has non-negligible effect on market prices. Without this assumption, there only are theoretical results showing incentive compatibility of minimum equilibrium prices in some simple cases: unit demand [4] and multi-unit auctions when valuations constraints are significantly weaker than budget constraints [10]. One can not hope for perfect theoretical incentive compatibility as [10] also show that *no* Pareto-optimal auction can be incentive compatible in the presence of budget limits[3].

Unfortunately, it is theoretically impossible to always reach such an equilibrium, even in this restricted setting, for two main theoretical reasons: The first

[3] In particular due to the non-quasi-linear setting, VCG prices are not incentive compatible, even if they could be computed efficiently.

reason is the computational difficulty: even when slot prices are given, computing the demand of a bidder is equivalent to a knapsack problem. Computing the equilibrium allocation can only be harder, as computing the demand is a special case. This is addressed by taking account of the fact that usually slot prices are relatively small compared to budgets. In this case, we are close to a fractional setting, where the demand of a bidder is efficiently computed by a greedy algorithm. Not only is this greedy algorithm a good approximation, but when more realistic issues are taken into account, in section 3.1, it may be argued that it represents the demand better than the theoretical optimum.

The second reason is that it is well known that a Walrasian equilibrium may not exist unless all demands are "gross substitutes", which they need not be in our case. Thus even ignoring computational issues an equilibrium may not exist at all. Appendix B gives an example. This is handled in our auction by first relaxing the condition that all non-reserve-priced slots must be allocated, and then allocating the relatively few unsold spots in a sub-optimal "remnant inventory sale" round.

2.3 The Simultaneous Ascending Auction

We describe here the basic version of the auction, still dealing only with the basic scenario formalized above. This basic version is the framework for the complete solution, and in the next section we will describe the various changes and enhancements to the basic algorithm. The overall idea is simple:

Initialization:

1. For all slots j, set price to reserve: $p_j \leftarrow r_j$.
2. Start with an empty allocation: For all bidders $i > 0$, $S_i \leftarrow \emptyset$, and $S_0 \leftarrow$ the set of all slots.
3. For all advertisers i, enqueue i into the bidder queue.

Main loop:

While bidder queue not empty do:

1. Dequeue the next bidder i from the bidder que.
2. Compute the demand D of bidder i greedily as follows:
   (a) Sort all slots in T_i with $\tilde{p}_j \leq v_i(j)$ according to decreasing value of $v_i(j)/\tilde{p}_j$, where $\tilde{p}_j = p_j$ for $j \in S_i \cup S_0$ and $\tilde{p}_j = p_j + \delta$ otherwise.
   (b) For all j according to the sorted order, if taking this slot does not exceed budget, $\tilde{p}_j + \sum_{t \in D} \tilde{p}_t \leq b_i$, then acquire it, $D \leftarrow D \cup \{j\}$.
3. For all slots $j \in D \cap S_k$ for some $i \neq k > 0$ do:
   (a) Increase price of j: $p_j \leftarrow p_j + \delta$.
   (b) Remove j from S_k.
   (c) if k is not already in the bidder queue then enqueue k.
4. Update the set of unallocated spots: $S_0 \leftarrow S_0 \cup (S_i - D)$.
5. Update S_i: $S_i \leftarrow D$.

Output:

Each bidder $i > 0$ is allocated all slots j in S_i, at a price of p_j for slot j. Slots in S_0 are left un-allocated at this point.

The key step in this auction is the calculation of the demand D. It is important that this step only depends on bidder i's information as well as the values $\tilde{p}_j$ and not on any other global information. The goal of the step is to find the set of slots that maximize i's utility $\sum_{j \in D}(v_i(j) - \tilde{p}_j)$ subject to the budget constraint that $\sum_{j \in D} \tilde{p}_j \leq b_i$. The greedy algorithm gives the optimal solution to the fractional variant of the problem where a slot may be partially taken at any fraction α for price $\alpha\tilde{p}_j$ and giving value $\alpha v_i(j)$, and thus is practically a good approximation to the optimal set[4]. Notice also the modularity and flexibility of demand computation allowing easy extensions to take into account various other bidder preferences as described in section 3.

This ascending auction algorithm follows the theoretical work starting with [4,5], and described, e.g., in [11]. It is easy to see that it ends with a Walrasian equilibrium (up to the additive δ) if no slots remain un-allocated. This is known to be the case with "gross substitutes" bidders in which case it ends with the minimal equilibrium prices[5]. In general, however, and especially given the complications discussed in section 3, some slots may remain unallocated at this stage and are allocated in a next "remnant inventory" stage.

2.4 Remnant Inventory

The auction main loop ends with some slots unsold, those in S_0. The remnant sale sells these slots. In this stage all bidders participate as before, but their demands take into account the slots that they have already won. The logic of this stage is heuristic, basically attempting to sell as much as possible at prices that are as close as possible to the attempted "equilibrium prices" from the previous round. This stage proceeds by reducing (slightly) all prices p_j of the remaining slots, and re-running the main loop, selling some more slots. This is repeated until all remaining unsold slots are priced at their reserve price which means that they can not be sold at all and are left un-allocated.

This round is certainly a heuristic; a theoretical foundation for handling relatively small deviations from equilibrium would be of considerable interest.

3 Some Complications

3.1 The Imprecise Nature of Budgets

Budgets play a critical role in the problem formulation above, as they do in reality: an advertiser's budget is usually the *main* constraint on his allocated

[4] A dynamic programming algorithm could give a provable tighter approximation, but would require more running time, be much less flexible to addition of further constraints, and even more importantly would likely give a worse model of the bidder's true utility as we discuss in section 3.1.

[5] The minimum is well defined as equilibrium prices turn out to be a lattice [5].

set of spots. In reality, however, budgets are not totally well defined as in our formulation for a host of reasons, including the following significant ones:

1. Google charges TV advertisers according to the actual number of people that watched the ad (as reported by their cable boxes) and these real payments need to be constrained by the budget but are not known at allocation time, when only an estimate is available. Thus the algorithm works with estimated payments that may later turn out to be smaller or larger than the real payments.[6] The implication is that the budget should not be treated like a clear-cut constraint but rather as a "band" in which the higher you get the higher the probability that you are over-budget.
2. The hard budget constraints are usually specified by the advertisers for longer periods of time (month, week, or campaign-length), which encompass multiple auctions. While it is expected that the single-auction daily budget is approximately the appropriate proportion, this is not a hard constraint. Indeed the current Google adwords rules allow exceeding a single day's budget by up to 20%, and only treat the longer-term budgets as "hard".

The implication from this is double: first, since the budget should really be treated as a "smooth" constraint, there is some room for optimizations as well as policy decisions in regards to the exact stopping rule in calculating the demand using the greedy algorithm. A simple example of an optimization is for handling the integrality constraint at the "last spot" — the one that just goes over budget. This may be allowed as long as it does not go over-budget beyond some threshold. An example for a policy decision is to be conservative in optimizations and try to stick closely to proportional daily budgets as to simplify advertiser control of their campaign. The smoothness of the budget constraint further justifies the greedy algorithm for computing demand rather than trying some kind of knapsack algorithm, since the whole justification of the latter is dealing with the sharp integrality constraint at the budget limit, without which the greedy algorithm is optimal (i.e. for the fractional knapsack problem.)

We suggest that some more detailed modeling of budget constraints may be of considerable interest in various settings.

3.2 Crowd Control

The basic formulation of the problem did not place any structural constraints on the set of slots allocated to a single bidder. In reality there are some constraints of this form, where the most significant ones forbid too much "crowding" of slots on the same station. Such constraints come in different flavors: they may forbid a single ad to to appear twice in the same commercial break or within some predefined time gap, they may place the restriction on a single ad, on all ads by the same advertiser, or even on ads by different advertisers in the same

[6] In the algorithm above, p_j is the total price of the ad given the estimated number of impressions for it. The actual bids and payments are on CPM basis, i.e. after the ad is aired will be scaled by (actual number of impressions)/(estimated impressions).

industry. These kind of restrictions can represent the requirements of publishers or of advertisers, and must be respected by the auction. These constraints are easily incorporated into the auction by modifying the greedy *demand* algorithm to take them into account: in each greedy step we check whether taking the slot would violate "crowding" constraints, and skip the slot if this is the case. While the greedy algorithm is no longer theoretically optimal for calculating the demand even in the fractional case under these constraints, we did not observe a significant sub-optimality in practice. Appendix C shortly provides a theoretical analysis of such constraints that seems of general interest.

A constraint between different advertisers in the same industry is conceptually more problematic: it breaks the basic model of a combinatorial auction since it introduces externalities: whether I can take a slot or not depends also on someone else's allocation. While in principle it is possible to "internalize" these externalities into the model by introducing "crowding tokens" which are also put in auction, this would have significant overhead. A simpler, although not "perfectly correct" solution is to simply incorporate these "industry" crowding constraints into the greedy demand logic, slightly breaking the theoretical contract that the demand is a function of solely the current prices. One significant addition to the basic auction algorithm which is needed here is a mechanism that ensures that an advertiser is re-scheduled for calculating his demand whenever an external constraint on him changes. A theoretical analysis and quantification of the effect of "mild" externalities in combinatorial auctions and of various ways of dealing with them seems to be of interest.

3.3 The Nature of Bidders: Accounts, Campaigns, and Creatives

All of our discussion assumes the atomic notion of the "bidders": the entities among which we allocate the slots and which are the strategic participants in the auction. The reality is more complicated: there is an hierarchy of entities among which we allocate. In the case of Google adwords the hierarchy contains three levels.

1. The *Account:* Represents a single advertiser (company).
2. The *Campaign:* Represents an advertising campaign with its own budget and goals. An account may run multiple campaigns.
3. The *Creative:* Represents a single ad. A campaign may run multiple ads.

Now, which of these entities should a bidder be? From one point of view, the allocation is ultimately between ads, so the creative level seems right. From a different point of view the advertiser is really the strategic player in the auction, so the account level seems right. However, it seems that the campaign level is really the preferred answer. Conceptually, a campaign has a goal that it is trying to achieve, and the significant budget constraint usually is the campaign budget. Indeed, bidding is set on a campaign level. Some modification need to be made to the auction in order to handle the other levels of the hierarchy. First, we must sub-allocate each campaign's allocated slots among its creatives. This allocation is not price-based but rather by non-economic criteria usually, more or less, by rotation. Second, we must take care of special relations between campaigns in the same account, in particular they may share an "account budget" — a limit on the combined expenditure of all campaigns in the same account — which in terms of our campaign-based modeling is an externality. Also, as a matter of policy, it might be required that campaigns from the same account do not compete with each other, driving prices up without real competition from another advertiser.

We are not aware of theoretical work that attempts to directly model this "fuzziness" in the nature of the agents themselves, but this certainly seems like an interesting research direction.

3.4 Long vs. Short ads

All of our discussion so far assumed that all ads are of the same length, and thus all advertisers that target the same slot simply compete against each other. In reality, ads come in several standardized lengths: current TV industry standards use an integer multiple of 15 seconds, and so we need to enhance our setting to allow ads whose length is a small integer number of slots (practically between a single slot and eight consecutive slots). The main difficulty arises when different length ads compete for the same slots: should we prefer a $5 bid for 2-slots or a $3 bid for 1-slot? In a totally "liquid" situation, which behaves just like the fractional setting, there would also be another $3 1-slot bid for these 2 slots and thus taking the two $3 bids for a total of $6 is certainly best. In practice this will not always be the case.

At the algorithmic level, the problem is not very difficult due to the consecutive linear nature of the multi-slot bids and can be solved polynomial time using dynamic programming [12][7]. However, the pricing problem here is significant since a bid for two consecutive slots has strong built-in complementarity: a single slot is worthless. The difficulties with such a situation are well known and appear at full strength even with a small example of selling 2 consecutive slots: Assume that Alice has a 2-slot ad at a value of v_A, while Bob and Charlie each have a 1-slot ad at values v_B and v_C, respectively. It terms of maximizing efficiency Alice should win whenever $v_A > v_B + v_C$, and should "logically" pay $v_B + v_C$. But how much should Bob and Charlie pay when $v_B + v_C > v_A$? It is well recognized [13,6] that in such a case the incentive compatible VCG payments are quite problematic: they would have Bob pay $max(0, v_A - v_C)$ and Charlie pay $max(0, v_A - v_B)$. This is awkward for several reasons, e.g., the payments may well be zero and may certainly be such that the combined payment is much less than v_A. A natural approach would be to let Bob and Charlie share paying a sum of v_A in some manner, e.g. proportionally to their bid. But this is strongly non-incentive compatible as it encourages free-riding: reducing my bid will reduce my payment.

Our approach has been to stick with the price-per-slot auction in the main ascending auction stage and then do a correction in the second remnant sale stage. This fits directly into the basic ascending slot price architecture. In places

[7] In fact, this is true even in conjunction with the crowding constraints described above.

with sufficient liquidity, it is optimal in terms of allocation and seems "correct" in terms of pricing. Specifically, during the main ascending stage we maintain a price for each slot and bids that require several continuous slots simply see the sum of the slot prices when they compute their demand. When a "short" ad takes a slot from a longer ad, the remaining slots are left un-allocated. This may lead to slots remaining unallocated at the end of the auction at which point they are sold in the "remanent sale". In fact, the competition between different length ads is usually the main source of un-allocated slots at the end of the first ascending stage[8].

In the remnant sale stage we change the pricing rule from being per-slot to taking the whole ad into account. Since we are already in a situation where it is clear that there are liquidity problems, a short ad can replace a longer one only by paying for the complete price of the replaced ad — even those slots that are not desired by the short one. Theoretically, the allocation achieved is no longer optimal and we also deviate from incentive compatibility, but since typically only a small fraction of slots are left unsold at the remnant stage, this is practically acceptable.

While much analysis of the basic "one two-item bid vs. two one-item bid" has appeared in the literature, we leave a comprehensive theoretic strategic treatment of this scenario when there are many slots and bidders (and so we are "somewhat close" to a fractional setting) as a research problem.

3.5 Auction Overlap

Our basic model considers a single fixed auction: all the input is available, then an algorithm that determines the allocation and pricing is run, and then results are reported to the systems that actually run the ads at the allocated times and that bill the advertisers appropriately. Unfortunately, reality has a significant "online" component: the "input" i.e. the inventory for sale arrives from the different publishers at different times. Some may know their inventory for Tuesday a week before, some may only have it late Monday night, and in some cases some preliminary information is available early and may then be updated later. Similarly, some publishers are willing to get their schedule for Tuesday late Monday night while others need it a few days in advance.

The way that this staggered schedule of availability of the input and the output is handled is as follows. Every time some output (i.e. allocation information for some set of slots) is needed, a new auction is run. At that point in time the auction is run on all inventory whose allocation affects the required set of slots. For inventory that is unavailable at that time, the preliminary available information — a prediction, if needed — is used. Only the allocation of slots that needs to be specified at this time is actually used, and the allocation of other slots is discarded, to be finalized in future auctions which may have better information.

As an example (which is similar in spirit though not details to the case in our auction) suppose that publisher A needs to get his schedule 2 days in advance

[8] As the complementarity between adjacent slots indeed is "farthest away" from the theoretical "gross substitutes" condition.

and publisher B requires only 1 day in advance, but then may also report his inventory to Google only 1 day in advance. We then hold two auctions on every day x: one for day $x + 1$ and the other day $x + 2$. The auction for day $x + 2$ will use the best estimates so far for B's inventory and the final inventory of A. The allocation it produces for B will be simply thrown away, and only the allocation for A will be committed. This will exhaust some of the budgets of advertisers for day $x + 2$. The auction for day $x + 1$ (run on day x) will take into account the budgets that were already spent for day $x + 1$ (by the auction run on day $x - 1$) and will use the final inventory of B to get the allocation for B.

The quality of results obtained by this staggered online algorithm depends of course on the quality of preliminary information. It is not needed really that the preliminary information or prediction get the exact inventory correctly but rather that prices implied by the predicted inventory are close enough to those implied by the actual final inventory.

While there has been some work on online auctions (see survey in [14]), we did not find any work that is directly applicable to our setting. We leave a disciplined theoretical analysis of this type of "staggered" online problem as an open problem.

3.6 Lack of Free Disposal

The model of combinatorial auctions implicitly assumes free disposal: an item which is not sold can simply be thrown away. In reality this need not always be the case. For example, a commercial break *must* be filled — empty airtime is not tolerated. The solution to this is very simple in principle: just have a bunch of "filler" ads ready that can be used to fill any empty slot. Of course preparing, managing, and approving these ads may be non-trivial in practice, but from this paper's auction-centric point of view the problem reduces to filling empty slots with appropriately chosen filler ads. In our auction, Google has "public service" announcements used for this purpose, while providing this public service to the community.

Acknowledgments

Of the many Googlers who contributed to this system, we would especially like to thank the following: Jag Duggal, Daniel Gertsacov, Kaustuv, Ryan Roemer, Steve Stuckenberg, Tal Sela, and Kevin Thompson.

References

1. Google: Adwords tv ads web site, http://www.google.com/adwords/tvads
2. Google: Adwords traditional media blog, http://google-tmads.blogspot.com/search/label/Google%20TV%20Ads
3. Bollapragada, S., Cheng, H., Phillips, M., Garbira, M., Gibbs, T., Humphreville, M.: Nbc's optimization systems increase revenues and productivity. In: Interfaces, pp. 47–60 (2002)

4. Demange, G., Gale, D., Sotomayor, M.: Multi-item auctions. Journal of Political Economy 94, 863–872 (1986)
5. Gul, F., Stacchetti, E.: Walrasian equilibrium with gross substitutes. Journal of Economic Theory 87, 95 (1999)
6. Milgrom, P.: Putting auction theory to work. Cambridge University Press, Cambridge (2004)
7. Cramton, P., Shoham, Y., Steinberg, R.: Combinatorial Auctions. MIT Press, Cambridge (2006)
8. Google: Protocol buffers web site, `http://code.google.com/apis/protocolbuffers`
9. Blumrosen, L., Nisan, N.: On the computational power of iterative auctions. In: ACM EC (2005)
10. Dobzinski, S., Lavi, R., Nisan, N.: Multi-unit auctions with budget constraints. In: FOCS 2008 (2008)
11. Blumrosen, L., Nisan, N.: Combinatorial auctions (a survey). In: Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.) Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
12. Rothkhof, M.H., Pekeč, A., Harstad, R.M.: Computationally manageable combinatorial auctions. Management Science 44(8), 1131–1147 (1998)
13. Rothkopf, M.H., Teisberg, T.J., Kahn, E.P.: Why are vickrey auctions rare? Journal of Political Economy 98, 94–109 (1990)
14. Parkes, D.C.: Online mechanisms. In: Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V. (eds.) Algorithmic Game Theory. Cambridge University Press, Cambridge (2007)
15. Nisan, N.: In: Cramton, P., Shoham, Y., Steinberg, R. (eds.) Combinatorial Auctions. Bidding Languages. MIT Press, Cambridge (2006)
16. Lehamnn, B., Lehmann, D., Nisan, N.: Combinatorial auctions with decreasing marginal utilities. Games and Economic Behavior 55(2), 270–296 (2006)

Appendix A: Difficulties in Problem Formulation

Let us look at a few natural attempts to formulate the desired goals of the auction and see why they do not make much sense. These will lead us to the market equilibrium formulation that we took. For concreteness, let us consider the following simple example:

Running Example: 100 slots are being auctioned among two advertisers: Alice has a \$60 budget and a \$3 value for each slot and Bob has a \$30 budget and a \$6 value per slot.

3.7 Independent Auctions?

The first approach that one may consider is to auction the spots one by one, each time to the highest bidder whose budget has not been exhausted yet. This seems to make much sense as slot values are independent of each other. This approach also has the very strong appeal of simplicity both in terms of implementation and in terms of of explaining it to the advertisers. One may think of various approaches to decide on pricing, but using the second price in each of the slot auctions seems appropriate for incentive compatibility. If we follow what this approach means in our case we see that Bob, the high bidder, will win all the first auctions until his budget is exhausted. Using a second-price rule, he will be charged $3 per slot, so his budget will run out after winning 10 slots. At this point, Alice will win the remaining 90 spots for free (or whatever low reserve price there is). The extreme un-fairness would be even more apparent if Alice only had a $10 budget — she would still win these 90 spots for free. Strategic manipulation can be extremely helpful here: had Bob declared a $2 per-slot bid instead, Alice would win the first slots for $2 each, exhausting her budget after 30 slots, and then Bob could reap the remaining 70 spots for free!

3.8 Maximize Revenue?

Suppose that are goal is maximizing revenue. This is achieved by charging the two advertisers their full budgets. This will also satisfy individual rationality as long as the allocation gives at least 20 slots to Alice and at least 5 to Bob. Thus revenue maximization provides very little guidance on which allocation to choose. So which other criteria should be use? Fairness and efficiency would seem to suggest allocating a slot to the one that has a higher value for it. Should Bob get all but 20 of the slots? This seems quite unfair. Why should he pay much less per slot? It also is clearly not incentive compatible since advertisers are strongly motivated to strategically reduce their declaration of the budget. Another difficulty of this approach is to what extent is taking the whole budget is justified: suppose that Bob only puts in a low $0.1 value for each spot (rather than $6). Can we still charge Alice her full $60 budget, even though the "second price" would only be $10 (10 cents for each of 100 slots)? Should we give some spots to Bob in this case and charge him, hence increasing our revenue further?

3.9 Maximize Efficiency?

Suppose, on the other hand, that our goal is maximizing efficiency. Let us further decide that our resolution of the trade off between different advertisers is the "utilitarian" one of maximizing the sum of values, $\sum_i \sum_{j \in S_i} v_i(j)$. How are the budgets taken into account? As previously, should Bob get all slots? How much should he pay?

Some previous papers have considered the budget limit as an upper bound on the value, $v_i(S) = min(b_i, \sum_{j \in S} v_i(j)$, and thus attempted to maximize $\sum_i (min(b_i, \sum_{j \in S_i} v_i(j))$. If this is done, then again any allocation that gives at least 20 slots to Alice and at least 10 to Bob would achieve this maximization. Again we get very little guidance on which allocation to choose. How much should they pay? VCG payments in this context make no sense since they would be 0 (and in general do not ensure incentive compatibility in our non-quasi-linear setting).

3.10 Market Equilibrium

At this point, we hope that the reader is coming to the realization, like we have, that the correct goal here is a market equilibrium.

Let us see what this equilibrium will look like here. At any price $p \leq 30/34 = 0.882...$ Alice will demand at least 68 slots and Bob at least 34 so there will be over-demand. Just above this price, Bob's demand would be 33 and Alice's 67, so the the lowest equilibrium price would be just above 88.2 cents. Bob would pay $29.1.. for his 33 slots and Alice would pay $59.1.. her 67 slots. This is quite efficient, nearly maximizes revenue, and seems to be quite fair. This case turns out to also be incentive compatible: no advertiser can gain by manipulating his bid.

Appendix B: Example with No Equilibrium Prices

Here is an example for a simple setting where no Walrasian equilibrium exists.

bidder	budget	v(a)	v(b)	v(c)
1	6	5	5	0
2	9	4	4	8
3	7	0	0	7

Assume towards contradiction that an equilibrium exists, and assume without loss of generality that $p_a \leq p_b$. If $p_a + p_b > p_c$ then bidder 2 demands c and $p_c \geq 7$ (otherwise 3 also demands it). But then bidder 1 demands only a, and b is left un-allocated contradicting the requirement that unallocated slots be priced at reserve (0 here). On the other hand, if $p_a + p_b < p_c$ then $p_c \leq 7$ (otherwise neither 2 nor 3 demand c and it is left un-allocated despite its non-zero price). But then both 1 and 2 demand a. Contradiction. The case $p_a + p_b = q_c$ is treated like the first case if c is allocated to 2, and like the second case otherwise, concluding the contradiction.

Appendix C: Additive Valuations with Pair-Wise Constraints

In this appendix we focus, in a general combinatorial auction setting, on constraints that forbid a bidder to win certain given pairs of items. This appendix does not address the interplay with budget constraints (which we have not analyzed and leave as a topic for further study) but rather reverts to the standard quasi-linear setting. A linear valuation with such constraints is given by two elements:

1. A value $v(j)$ for each item $j \in M$, where M is the set of items for sale.
2. A graph $G = (M, E)$, where E are the set of pairs that are forbidden to be taken together.

The valuation of a set is defined as $v(S) = \max_{I \subseteq S} \sum_{j \in I} v(j)$, where I ranges over all sets that are *independent* in G. This is essentially the "OR*" bidding language used on singleton valuations (see [15]). We believe that this is quite a useful "bidding language" in general and should be studied. Here are a few preliminary results, whose proofs are straight forward given the literature and omitted.

- Every linear valuation with pair-wise constraints lies in the class XOS of [16] and hence is sub-additive. There are linear valuations with pair-wise constraints that are not sub-modular and hence not gross-substitutes.
- Answering a value query or a demand query (see [9]) given the description of the valuation as above is NP-hard. The same is true for approximating the value to within $m^{0.5-\epsilon}$ or for finding the welfare-maximizing allocation between such valuations.
- When the graph is restricted to be an interval graph (as it is in our "crowding" constraints) both value and demand queries can be answered exactly in polynomial time (using dynamic programming).

Graph Sparsification in the Semi-streaming Model

Kook Jin Ahn and Sudipto Guha*

Department of Computer and Information Science, University of Pennsylvania,
Philadelphia PA 19104-6389
{kookjin,sudipto}@cis.upenn.edu

Abstract. Analyzing massive data sets has been one of the key motivations for studying streaming algorithms. In recent years, there has been significant progress in analysing distributions in a streaming setting, but the progress on graph problems has been limited. A main reason for this has been the existence of linear space lower bounds for even simple problems such as determining the connectedness of a graph. However, in many new scenarios that arise from social and other interaction networks, the number of vertices is significantly less than the number of edges. This has led to the formulation of the semi-streaming model where we assume that the space is (near) linear in the number of vertices (but not necessarily the edges), and the edges appear in an arbitrary (and possibly adversarial) order.

However there has been limited progress in analysing graph algorithms in this model. In this paper we focus on graph sparsification, which is one of the major building blocks in a variety of graph algorithms. Further, there has been a long history of (non-streaming) sampling algorithms that provide sparse graph approximations and it a natural question to ask: since the end result of the sparse approximation is a small (linear) space structure, can we achieve that using a small space, and in addition using a single pass over the data? The question is interesting from the standpoint of both theory and practice and we answer the question in the affirmative, by providing a one pass $\tilde{O}(n/\epsilon^2)$ space algorithm that produces a sparsification that approximates each cut to a $(1+\epsilon)$ factor. We also show that $\Omega(n \log \frac{1}{\epsilon})$ space is necessary for a one pass streaming algorithm to approximate the min-cut, improving upon the $\Omega(n)$ lower bound that arises from lower bounds for testing connectivity.

1 Introduction

The feasibility of processing graphs in the data stream model was one of the early questions investigated in the streaming model [9]. However the results were not encouraging, even to decide simple properties such as the connectivity of a graph, when the edges are streaming in an arbitrary order required $\Omega(n)$ space. In comparison to the other results in the streaming model, [1,16] which required polylogarithmic space, graph alogithms appeared to difficult in the streaming context and did not receive much attention subsequently.

* Research supported in part by an Alfred P. Sloan Research Fellowship, and NSF Awards CCF-0644119 and IIS-0713267.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 328–338, 2009.

However in recent years, with the remergence of social and other interaction networks, questions of processing massive graphs have once again become prominent. Technologically, since the publication of [9], it had become feasible to store larger quantities of data in memory and the semi-streaming model was proposed in [6,15]. In this model we assume that the space is (near) linear in the number of vertices (but not necessarily the edges). Since its formulation, the model has become more appealing from the contexts of theory as well as practice. From a theoretical viewpoint, the model still offers a rich potential trade-off between space and accuracy of algorithm, albeit at a different threshold than polylogarithmic space. From a practical standpoint, in a variety of contexts involving large graphs, such as image segmentation using graph cuts, the ability of the algorithm to retain the most relevant information in main memory has been deemed critical. In essence, an algorithm that runs out of main memory space would become unattractive and infeasible. In such a setting, it may be feasible to represent the vertex set in the memory whereas the edge set may be significantly larger.

In the semi-streaming model, the first results were provided by [6] on the construction of graph spanners. Subsequently, beyond explorations of connectivity [5], and (multipass) matching [14], there has been little development of algorithms in this model. In this paper we focus on the problem of graph sparsification in a single pass, that is, constructing a small space representation of the graph such that we can estimate the size of any cut. Graph sparsification [2,17] remains one of the major building blocks for a variety of graph algorithms, such as flows and disjoint paths, etc. At the same time, sparsification immediately provides a way of finding an approximate min-cut in a graph. The problem of finding a min-cut in a graph has been one of the more celebrated problems and there is a vast literature on this problem, including both deterministic [7,8] as well as randomized algorithms [10,11,13,12] – see [3] for a comprehensive discussion of various algorithms. We believe that a result on sparsification will enable the investigation of a richer class of problems in graphs in the semi-streaming model.

In this paper we will focus exclusively on the model that the stream is adversarially ordered and a single pass is allowed. From the standpoint of techniques, our algorithm is similar in spirit to the algorithm of Alon-Matias-Szegedy [1], where we simultaneously sample and estimate from the stream. In fact we show that in the semi-streaming model we can perform a similar, but non-trivial, simultaneous sampling and estimation. This is pertinent because sampling algorithms for sparsification exist [2,17], which use $\mathcal{O}(n\text{polylog}(n))$ edges. However these algorithms sample edges in an iterative fashion that requires the edges to be present in memory and random access to them.

Our Results. Our approach is to recursively maintain a summary of the graph seen so far and use that summary itself to decide on the action to be taken on seeing a new edge. To this end, we modify the sparsification algorithm of Benczur and Karger [2] for the semi–streaming model. The final algorithm uses a single pass over the edges and provides $1 \pm \epsilon$ approximation for cut values with high probability and uses $\mathcal{O}(n(\log n + \log m)(\log \frac{m}{n})(1+\epsilon)^2/\epsilon^2)$ edges for n node and m edge graph.

2 Background and Notation

Let G denote the input graph and n and m respectively denote the number of nodes and edges. $VAL(C, G)$ denotes the value of cut C in G. $w_G(e)$ indicates the weight of e in graph G.

Definition 1. *[2] A graph is **k-strong connected** if and only if every cut in the graph has value at least k. **k-strong connected component** is a maximal node-induced subgraph which is k-strong connected. The **strong connectivity** of an edge e is the maximum k such that there exists a k-strong connected component that contains e.*

In [2], they compute the strong connectivity of each edge and use it to decide the sampling probability. Algorithm 1 is their algorithm. We will modify this in section 3.

```
**Benczur-Karger**([2])
**Data**: Graph $G = (V, E)$
**Result**: Sparsified graph $H$
compute the strong connectivity of edge $c_e^G$ for all $e \in G$;
$H \leftarrow (V, \emptyset)$;
**foreach** $e$ **do**
    $p_e = \min\{\rho/c_e, 1\}$;
    with probability $p_e$, add $e$ to $H$ with weight $1/p_e$;
**end**
```

Algorithm 1. Sparsification Algorithm

Here ρ is a parameter that depends on the size of G and the error bound ϵ. They proved the following two theorems in their paper.

Theorem 1. *[2] Given ϵ and a corresponding $\rho = 16(d+2)(\ln n)/\epsilon^2$, every cut in H has value between $(1-\epsilon)$ and $(1+\epsilon)$ times its value in G with probability $1 - n^{-d}$.*

Theorem 2. *[2] With high probability H has $\mathcal{O}(n\rho)$ edges.*

Throughout this paper, $e_1, e_2, \cdots, e_m$ denotes the input sequence. G_i is a graph that consists of e_1,e_2,$\cdots$,e_i. $c_e^{(G)}$ is the strong connectivity of e in G and $w_G(e)$ is weight of an edge e in G. $G_{i,j} = \{e : e \in G_i, 2^{j-1} \leq c_e^{(G_i)} < 2^j\}$. Each edge has weight 1 in $G_{i,j}$. $F_{i,j} = \sum_{k \geq j} 2^{j-k} G_{i,j}$ where scalar multiplication of a graph and addition of a graph is defined as scalar multiplication and addition of edge weights. In addition, $H \in (1 \pm \epsilon)G$ if and only if $(1-\epsilon)VAL(C, G) \leq VAL(C, H) \leq (1+\epsilon)VAL(C, G)$. H_i is a sparsification of a graph G_i, i.e., a sparsified graph after considering e_i in the streaming model.

3 A Semi-streaming Algorithm

We cannot use Algorithm 1 in the streaming model since it is not possible to compute the strong connectivity of an edge in G without storing all the data. The overall idea would be to use a strongly recursive process, where we use an estimation of the connectivity based on the current sparsification and show that subsequent addition of edges does not impact the process. The modification is not difficult to state, which makes us believe that such a modification is likely to find use in practice. The nontrivial part of the algorithm is in the analysis, ensuring that the various dependencies being built into the process does not create a problem. For completeness the modifications are presented in Algorithm 2.

```
Stream-Sparsification
Data: The sequence of edges $e_1, e_2, \cdots, e_m$
Result: Sparsified graph $H$
$H \leftarrow \emptyset$;
foreach $e$ do
    compute the connectivity $c_e$ of $e$ in $H$;
    $p_e = \min\{\rho/c_e, 1\}$;
    add $e$ to $H$ with probability $p_e$ and weight $1/p_e$;
end
```

Algorithm 2. Streaming Sparsification Algorithm

We use $\rho = 32((4+d)\ln n + \ln m)(1+\epsilon)/\epsilon^2$ given $\epsilon > 0$; once again d is a constant which determines the probability of success. We prove two theorems for Algorithm 2. The first theorem is about the approximation ratio and the second theorem is about its space requirement. For the simplicity of proof, we only consider sufficiently small ϵ.

Theorem 3. *Given $\epsilon > 0$, H is a sparsification, that is $H \in (1 \pm \epsilon)G$, with probability $1 - \mathcal{O}(1/n^d)$.*

Theorem 4. *If $H \in (1 \pm \epsilon)G$, H has $\mathcal{O}(n(d \log n + \log m)(\log m - \log n)(1+\epsilon)^2/\epsilon^2)$ edges.*

We use a sequence of ideas similar to that in Benczur and Karger [2]. Let us first discuss the proof in [2].

In that paper, Theorem 1 is proved on three steps. First, the result of Karger [11], on uniform sampling is used. This presents two problems. The first is that they need to know the value of minimum cut to get a constant error bound. The other is that the number of edges sampled is too large. In worst case, uniform sampling gains only constant factor reduction in number of edges.

To solve this problem, Benczur and Karger [2] decompose a graph into k-strong connected components. In a k-strong connected component, minimum-cut is at least k while the maximum number of edges in k-strong connected component(without $(k+1)$-strong connected component as its subgraph) is at most kn. They used the uniform sampling for each component and different sampling rate for different components. In this way, they guarantee the error bound for every cut.

We cannot use Karger's result [11] directly to prove our sparsification algorithm because the probability of sampling an edge depends on the sampling results of previous edges. We show that the error bound of a single cut by a suitable bound on the martingale process. Using that we prove that if we do not make an error until i^{th} edge, we guarantee the same error bound for every cut after sampling $(i+1)^{\mathrm{th}}$ edge with high probability. Using union bound, we prove that our sparsification is good with high probability.

4 Proof of Theorem 3

4.1 Single Cut

We prove Theorem 3 first. First, we prove the error bound of a single cut in Lemma 1. The proof will be similar to that of Chernoff bound [4]. p in Lemma 4 is a parameter and we use different p for different strong connected components in the later proof.

Lemma 1. *Let $C = \{e_{i_1}, e_{i_2}, \cdots, e_{i_l}\}$ with $i_1 < i_2 < \cdots < i_l$ be a cut in a graph G such that $w_G(e_{i_j}) \leq 1$ and $VAL(C, G) = c$. The index of the edges corresponds to the arrival order of the edges in the data stream. Let A_C be an event such that $p_e \geq p$ for all $e \in C$. Let H be a sparsification of G. Then, $\mathbb{P}[A_C \wedge (|VAL(C,H) - c| > \beta c)] < 2\exp(-\beta^2 pc/4)$ for any $0 < \beta \leq 2e - 1$.*

Let $X_j = pw_H(e_{i_j})$ and $\mu_j = \mathbf{E}[X_j] = pw_G(e_{i_j})$. Then, $|VAL(C,H) - c| > \beta c$ if and only if $|\sum_j X_j - pc| > \beta pc$. As already mentioned, we cannot apply Chernoff bound because there are two problems:

1. X_j are not independent from each other and
2. values of X_j are not bounded.

The second problem is easy to solve because we have A_C. Let Y_j be random variables defined as follows:

$$Y_j = \begin{cases} X_j \text{ if } p_{e_{i_j}} \geq p \\ \mu_j \text{ otherwise.} \end{cases}$$

If A_C happens, $Y_j = X_j$. Thus,

$$\begin{aligned}
\mathbb{P}[A_C \wedge (|VAL(C,H) - c| > \beta c)] &= \mathbb{P}[A_C \wedge (|\sum_j X_j - \sum_j \mu_j| > \beta pc)] \\
&= \mathbb{P}[A_C \wedge (|\sum_j Y_j - \sum_j \mu_j| > \beta pc)] \\
&\leq \mathbb{P}[|\sum_j Y_j - \sum_j \mu_j| > \beta pc] \qquad (1)
\end{aligned}$$

The proof of (1) is similar to Chernoff bound [4]. However, since we do not have independent Bernoulli random variables, we need to prove the upperbound of $\mathbf{E}[\exp(t\sum_j Y_j)]$ given t. We start with $\mathbf{E}[\exp(tY_j)]$.

Lemma 2. $\mathbf{E}[\exp(tY_j)|H_{i_j-1}] \leq \exp(\mu_j(e^t - 1))$ *for any t and H_{i_j-1}.*

Proof: There are two cases. Given H_{i_j-1}, $p_{e_{i_j}} \geq p$ or $p_{e_{i_j}} < p$. At the end of each case, we use the fact that $1 + x < e^x$.

Case 1 : If $p_{e_{i_j}} < p$, $Y_j = \mu_j$.

$$\begin{aligned}\mathbf{E}[\exp(tY_j)|H_{i_j-1}] &= \exp(t\mu_j)\\ &< \exp(\mu_j(e^t - 1)).\end{aligned}$$

Case 2 : If $p_{e_{i_j}} \geq p$, $Y_j = X_j$. So $\mathbf{E}[\exp(tY_j)|H_{i_j-1}] = p_{e_{i_j}} \exp(t\mu_j/p_{e_{i_j}}) + (1 - p_{e_{i_j}})$. Let $f(x) = x\exp(t\mu_j/x) + (1 - x)$. Observe that $f'(x) \leq 0$ for $x > 0$. So $f(x)$ is decreasing function. Also we have $\mu_j = pw_G(e_{i_j}) \leq p \leq p_{e_{i_j}}$ since $w_G(e_{i_j}) \leq 1$. Hence,

$$p_{e_{i_j}} \exp(t\mu_j/p_{e_{i_j}}) + (1 - p_{e_{i_j}}) \leq \mu_j \exp(t) + (1 - \mu_j).$$

Therefore,

$$\begin{aligned}\mathbf{E}[\exp(tY_j)|H_{i_j-1}] &\leq \mu_j(\exp(t) - 1) + 1\\ &\leq \exp(\mu_j(e^t - 1)).\end{aligned}$$

From case 1 and 2, $\mathbf{E}[\exp(tY_j)|H_{i_j-1}] \leq \exp(\mu_j(e^t - 1))$ for any H_{i_j-1}. □

Now, we prove the upperbound of $\mathbf{E}[\exp(t\sum_j Y_j)]$.

Lemma 3. *Let* $S_j = \sum_{k=j}^{l} Y_k$. *For any* t *and* H_{i_j-1}, $\mathbf{E}[\exp(tS_j)|H_{i_j-1}] \leq \exp(\sum_{k=j}^{l} \mu_j(e^t - 1))$.

Proof: We prove by induction. For $j = l$, $\mathbf{E}[\exp(tS_j)|H_{i_j-1}] = \mathbf{E}[\exp(tY_l)|H_{i_j-1}]$ $\leq \exp(\mu_l(e^t - 1))$ by Lemma 2.

Assume that $\mathbf{E}[\exp(tS_{j+1})|H_{i_{j+1}-1}] \leq \exp(\sum_{k=j+1}^{l} \mu_k(e^t - 1))$ for any $H_{i_{j+1}-1}$. Then,

$$\begin{aligned}\mathbf{E}[\exp(tS_j)|H_{i_j-1}] &= \sum_y \mathbb{P}[Y_j = y|H_{i_j-1}] \sum_{H_{i_{j+1}-1}} \mathbf{E}[\exp(t(y + S_{j+1}))|H_{i_{j+1}-1}]\mathbb{P}[H_{i_{j+1}-1}|Y_j = y, H_{i_j-1}]\\ &= \sum_y \exp(ty)\mathbb{P}[Y_j = y|H_{i_j-1}] \sum_{H_{i_{j+1}-1}} \mathbf{E}[\exp(tS_{j+1})|H_{i_{j+1}-1}]\mathbb{P}[H_{i_{j+1}-1}|Y_j = y, H_{i_j-1}]\\ &\leq \sum_y \mathbb{P}[Y_j = y|H_{i_j-1}] \exp\left(\sum_{k=j+1}^{l} \mu_k(e^t - 1)\right)\\ &= \exp\left(\sum_{k=j+1}^{l} \mu_k(e^t - 1)\right) \mathbf{E}[Y_j|H_{i_j-1}]\\ &\leq \exp\left(\sum_{k=j}^{l} \mu_k(e^t - 1)\right)\end{aligned}$$

Therefore, $\mathbf{E}[\exp(tS_j)|H_{i_j-1}] \leq \exp(\sum_{k=j}^{n} \mu_k(e^t - 1))$ for any H_{i_j-1} and t. □

Now we prove Lemma 1. Remember that we only need to prove $\mathbb{P}[|\sum_j Y_j - pc| > \beta pc] < 2\exp(-\beta^2 pc/4)$ by (1).

Proof:[Proof of Lemma 1] Let $S = S_1 = \sum_j Y_j$ and $\mu = \sum_j \mu_j = pc$. We prove in two parts: $\mathbb{P}[S > (1+\beta)\mu] \leq \exp(-\beta^2\mu/4)$ and $\mathbb{P}[S < (1-\beta)\mu] \leq \exp(-\beta^2\mu/4)$.

We prove $\mathbb{P}[S > (1+\beta)\mu] < \exp(-\beta^2\mu/4)$ first. By applying Markov's inequality to $\exp(tS)$ for any $t > 0$, we obtain

$$\begin{aligned}\mathbb{P}(S > (1+\beta)\mu) &< \frac{\mathbf{E}[\exp(tS)]}{\exp(t(1+\beta)\mu)} \\ &\leq \frac{\exp(\mu(e^t-1))}{\exp(t(1+\beta)\mu)}.\end{aligned}$$

The second line is from Lemma 3. From this point, we have identical proof as Chernoff bound [4] that gives us bound $\exp(-\beta^2\mu/4)$ for $\beta < 2e - 1$. To prove that $\mathbb{P}[S < (1-\beta)\mu] < \exp(-\beta^2 pc/4)$ we applying Markov's inequality to $\exp(-tS)$ for any $t > 0$, and proceed similar to above. Using union bound to these two bounds, we obtain a bound of $2\exp(-\beta^4\mu/4)$. □

4.2 *k*-Strong Connected Component

Now we prove the following lemma given a k-strong connected component and parameter p. This corresponds to the proof of uniform sampling method in [11].

Lemma 4. *Let Q be a k-strong component such that each edge has weight at most 1. H_Q is its sparsified graph. Let $\beta = \sqrt{4((4+d)\ln n + \ln m)/pk}$ for some constant $d > 0$. Suppose that A_Q be an event such that every edge in Q has sampled with probability at least p. Then, $\mathbb{P}[A_Q \wedge (H_Q \notin (1 \pm \epsilon)Q)] = \mathcal{O}(1/n^{2+d}m)$.*

Proof: Consider a cut C whose value is αk in Q. If A_Q holds, every edge in C is also sampled with probability at least p. By Lemma 1, $\mathbb{P}[A_Q \wedge |VAL(C, H_Q) - \alpha k| > \beta\alpha k] \leq 2\exp(-\beta^2 p\alpha k/4) = 2(n^{4+d}m)^{-\alpha}$. Let $P(\alpha) = 2(n^{4+d}m)^{-\alpha}$.

Let $F(\alpha)$ be the number of cuts with value less or equal to αk. By union bound, we have

$$\mathbb{P}[A_Q \wedge (H_Q \notin (1 \pm \epsilon)Q)] \leq P(1)F(1) + \int_1^\infty P(\alpha)\frac{dF}{d\alpha}d\alpha.$$

The number of cuts whose value is at most α times minimum cut is at most $n^{2\alpha}$. Since the value of minimum cut of Q is k, $F(\alpha) \leq n^{2\alpha}$. Since P is a monotonically increasing function, this bound is maximized when $F(\alpha) = n^{2\alpha}$. Thus,

$$\begin{aligned}\mathbb{P}[A_Q \wedge (H_Q \notin (1 \pm \epsilon)Q)] &\leq F(1)P(1) + \int_1^\infty P(\alpha)\frac{dF}{d\alpha}d\alpha \\ &\leq n^2 P(1) + \int_1^\infty P(\alpha)(2n^{2\alpha}\ln n)d\alpha \\ &\leq \frac{2}{n^{2+d}m} + \int_1^\infty \frac{\ln n}{n^{\alpha(2+d)}m^\alpha}d\alpha \\ &= \mathcal{O}(1/n^{2+d}m).\end{aligned}$$

□

4.3 Error Bound for H_i and H

Lemma 5. *The probability of i being the first integer such that $H_i \notin (1 \pm \epsilon)G_i$ is $\mathcal{O}(1/n^d m)$.*

Proof: If $H_j \in (1 \pm \beta)G_j$ for all $j < i$, $c_{e_j} \leq (1+\epsilon)c_{e_j}^{(G_j)} \leq (1+\epsilon)c_{e_j}^{(G_i)}$. Remember that $c_e^{(G)}$ denotes the strong connectivity of e in graph G.

$$\begin{aligned} H_i &= \sum_{j=-\infty}^{\infty} H_{i,j} \\ &= \sum_{j=-\infty}^{\infty} \left(H_{i,j} + \frac{1}{2}F_{i,j+1} \right) - \sum_{j=-\infty}^{\infty} \frac{1}{2}F_{i,j+1} \end{aligned}$$

$H_{i,j} + (1/2)F_{i,j+1}$ is a sparsification of $G_{i,j} + (1/2)F_{i,j+1} = F_{i,j}$. $F_{i,j}$ consists of 2^{j-1}-strong connected components. For every $e \in G_{i,j}$, $c_e^{(G_i)} < 2^j$. So it is sampled with probability at least $p = \rho/(1+\epsilon)2^j$. If we consider one 2^{j-1}-strong connected component and set $\rho = 32((4+d)\ln n + \ln m)(1+\epsilon)/\epsilon^2$, by Lemma 4, every cut has error bound $\epsilon/2$ with probability at least $1 - \mathcal{O}(1/n^{2+d}m)$. Since there are less than n^2 such distinct strong connected components, with probability at least $1 - \mathcal{O}(1/n^d m)$, $H_{i,j} + (1/2)F_{i,j+1} \in (1 \pm \beta)F_{i,j}$ for every i, j. Hence,

$$\begin{aligned} H_i &\in \sum_{j=-\infty}^{\infty} (1 \pm \epsilon/2)F_{i,j} - \sum_{j=-\infty}^{\infty} \frac{1}{2}F_{i,j+1} \\ &\subseteq (2 \pm \epsilon)G_i - G_i \\ &= (1 \pm \epsilon)G_i. \end{aligned}$$

Therefore, $\mathbb{P}[(\forall j < i.H_j \in (1 \pm \epsilon)G_j) \wedge (H_i \notin (1 \pm \epsilon)G_i)] = \mathcal{O}(1/n^d m)$. □

From Lemma 5, Theorem 3 is obvious. $\mathbb{P}[H \notin (1 \pm \epsilon)G] \leq \sum_{i=1}^{m} \mathbb{P}[(\forall j < i.H_j \in (1 \pm \epsilon)G_j) \wedge (H_i \notin (1 \pm \epsilon)G_i)] = \mathcal{O}(1/n^d)$.

5 Proof of Theorem 4

For the proof of Theorem 4, we use the following property of strong connectivity.

Lemma 6. *[2] If the total edge weight of graph G is $n(k-1)$ or higher, there exists a k-strong connected components.*

Lemma 7. *$H \in (1 \pm \epsilon)G$, total edge weight of H is at most $(1+\epsilon)m$.*

Proof: Let C_v be a cut $(\{v\}, V - \{v\})$. Since $H \in (1 \pm \epsilon)G$, $VAL(C_v, H) \leq (1 + \epsilon)VAL(C_v, G)$. Total edge weight of H is $(\sum_{v \in V} VAL(C_v, H))/2$ since each edge is counted for two such cuts. Similarly, G has $(\sum_{v \in V} VAL(C_v, H))/2 = m$ edges. Therefore, if $H \in (1 \pm \epsilon)G$, total edge weight of H is at most $(1+\epsilon)m$. □

Let $E_k = \{e : e \in H \text{ and } c_e \leq k\}$. E_k is a set of edges that sampled with $c_e = k$. We want to bound the total weight of edges in E_k.

Lemma 8. $\sum_{e\in E_k} w_H(e) \le n(k+k/\rho)$.

Proof: Let H' be a subgraph of H that consists of edges in E_k. H' does not have $(k+k/\rho+1)$-strong connected component. Suppose that it has. Then there exists the first edge e that creates a $(k+k/\rho+1)$-strong connected component in H'. In that case, e_i must be in the $(k+k/\rho+1)$-strong connected component. However, since weight e is at most k/ρ, that component is at least $(k+1)$-strong connected without e. This contradicts that $c_e \le k$. Therefore, H' does not have any $(k+k/\rho+1)$-strong connected component. By Lemma 6, $\sum_{e\in E_k} w_H(e) \le n(k+k/\rho)$. □

Now we prove Theorem 4.

Proof:[Proof of Theorem 4] If the total edge weight is the same, the number of edges is maximized when we sample edges with smallest strong connectivity. So in the worst case,

$$\sum_{e\in E_k - E_{k-1}} w_H(e) = nk(1+\rho) - n(k-1)(1+\rho) = n(1+\rho).$$

In that case, k is at most $(1+\epsilon)m/n(1+1/\rho)$. Let this value be k_m. Then, total number of edges in H is

$$\begin{aligned}\sum_{i=1}^{k_m} \frac{n(1+1/\rho)}{i/\rho} &= n(\rho+1)\sum_{i=1}^{k_m}\frac{1}{i}\\ &= O(n(\rho+1)\log(k_m))\\ &= O(n\rho(\log m - \log n))\\ &= O(n(d\log n + \log m)(\log m - \log n)(1+\epsilon)^2/\epsilon^2).\end{aligned}$$

□

6 Space Lower Bounds

First, we prove a simple space lowerbound for weighted graphs, where the lowerbound depends on ϵ.

Theorem 5. *For $0<\epsilon<1$, $\Omega(n(\log C + \log\frac{1}{\epsilon}))$ bits are required in order to sparsify every cut of a weighted graph within $(1\pm\epsilon)$ factor where C is maximum edge weight and 1 is minimum edge weight.*

Proof: Let F be a set of graphs such that there is a center node u and other nodes are connected to u by an edge whose weight is one of $1, \left(\frac{1+\epsilon}{1-\epsilon}\right), \left(\frac{1+\epsilon}{1-\epsilon}\right)^2, \cdots, C$. Then, $|F| = (\log_{\left(\frac{1+\epsilon}{1-\epsilon}\right)} C)^{n-1}$. For $G, G' \in F$, they must have different sparsifications. So we need $\Omega(\log|F|)$ bits for sparsfication. It is easy to show that $\log|F| = \Omega(n(\log C + \log\frac{1}{\epsilon}))$. □

Now we use the same proof idea for unweighted simple graphs. Since we cannot assign weight as we want, we use $n/2$ nodes as a center instead of having one center node. In this way, we can assign degree of a node from 1 to $n/2$.

Theorem 6. *For* $0 < \epsilon < 1$, $\Omega(n(\log n + \log \frac{1}{\epsilon}))$ *bits are required in order to sparsify every cut of a graph within* $(1 \pm \epsilon)$.

Proof: Consider bipartite graphs where each side has exactly $n/2$ nodes and each node in one side has a degree 1, $\left(\frac{1+\epsilon}{1-\epsilon}\right)$, $\left(\frac{1+\epsilon}{1-\epsilon}\right)^2, \cdots$, or $n/2$. For each degree assignment, there exists a graph that satisfies it. Let F be a set of graphs that has different degree assignments. Then, $|F| = \left(\log_{\left(\frac{1+\epsilon}{1-\epsilon}\right)} \frac{n}{2}\right)^{n-1}$. $G, G' \in F$ cannot have the same sparsification. So we need at least $\Omega(\log |F|) = \Omega(n(\log n + \log \frac{1}{\epsilon}))$ bits. □

Another way of viewing the above claim is a direct sum construction, where we need to use $\Omega(\log \frac{1}{\epsilon})$ bits to count upto a precision of $(1 + \epsilon)$.

7 Conclusion and Open Problems

We presented a one pass semi-streaming algorithm for the adversarially ordered data stream model which uses $O(n(d \log n + \log m)(\log m - \log n)(1 + \epsilon)^2/\epsilon^2)$ edges to provide ϵ error bound for cut values with probability $1 - O(1/n^d)$. If the graph does not have parallel edges, the space requirement reduces to $O(dn \log^2 n(1 + \epsilon)^2/\epsilon^2)$. We can solve the minimum cut problem or other problems related to cuts with this sparsification. For the minimum cut problem, this provides one-pass $((1 + \epsilon)/(1 - \epsilon))$-approximation algorithm.

A natural open question is to determine how the space complexity of the approximation depends on ϵ. Our conjecture is that the bound of n/ϵ^2 is tight up to logarithmic factors.

References

1. Alon, N., Matias, Y., Szegedy, M.: The Space Complexity of Approximating the Frequency Moments. J. Comput. Syst. Sci. 58(1), 137–147 (1999)
2. Benczúr, A.A., Karger, D.R.: Approximating s-t minimum cuts in O(n2) time. In: STOC 1996: Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, pp. 47–55. ACM, New York (1996)
3. Chekuri, C.S., Goldberg, A.V., Karger, D.R., Levine, M.S., Stein, C.: Experimental study of minimum cut algorithms. In: SODA 1997: Proceedings of the eighth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 324–333. Society for Industrial and Applied Mathematics (1997)
4. Chernoff, H.: A measure of the asymptotic efficiency for tests of a hypothesis based on the sum of observations. Annals of Mathematical Statistics 23, 493–509 (1952)
5. Demetrescu, C., Finocchi, I., Ribichini, A.: Trading off space for passes in graph streaming problems. In: SODA, pp. 714–723 (2006)
6. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theor. Comput. Sci. 348(2), 207–216 (2005)
7. Gomory, R.E., Hu, T.C.: Multi-terminal network flows. J. Soc. Indust. Appl. Math. 9(4), 551–570 (1961)
8. Hao, J., Orlin, J.B.: A faster algorithm for finding the minimum cut in a graph. In: SODA 1992: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 165–174. Society for Industrial and Applied Mathematics (1992)

9. Henzinger, M., Raghavan, P., Rajagopalan, S.: Computing on data streams (1998)
10. Karger, D.R.: Global min-cuts in rnc, and other ramifications of a simple min-out algorithm. In: SODA 1993: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 21–30. Society for Industrial and Applied Mathematics (1993)
11. Karger, D.R.: Random sampling in cut, flow, and network design problems. In: STOC 1994: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing, pp. 648–657. ACM Press, New York (1994)
12. Karger, D.R.: Minimum cuts in near-linear time. J. ACM 47(1), 46–76 (2000)
13. Karger, D.R., Stein, C.: A new approach to the minimum cut problem. J. ACM 43(4), 601–640 (1996)
14. McGregor, A.: Finding Graph Matchings in Data Streams. In: Proc. of APPROX-RANDOM, pp. 170–181 (2005)
15. Muthukrishnan, S.: Data streams: Algorithms and Applications. Now publishers (2006)
16. Ian Munro, J., Paterson, M.: Selection and Sorting with Limited Storage. Theor. Comput. Sci. 12, 315–323 (1980)
17. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. In: STOC 2008: Proceedings of the 40th annual ACM symposium on Theory of computing, pp. 563–568. ACM Press, New York (2008)

Sort Me If You Can: How to Sort Dynamic Data

Aris Anagnostopoulos[1,⋆], Ravi Kumar[2], Mohammad Mahdian[2], and Eli Upfal[3,⋆⋆]

[1] Sapienza University of Rome, Rome, 00185, Italy
aris@cs.brown.edu
[2] Yahoo! Research, Sunnyvale, CA 94089, USA
{ravikumar,mahdian}@yahoo-inc.com
[3] Brown University, Providence, RI 02912, USA
eli@cs.brown.edu

Abstract. We formulate and study a new computational model for dynamic data. In this model the data changes gradually and the goal of an algorithm is to compute the solution to some problem on the data at each time step, under the constraint that it only has a limited access to the data each time. As the data is constantly changing and the algorithm might be unaware of these changes, it cannot be expected to always output the exact right solution; we are interested in algorithms that guarantee to output an approximate solution. In particular, we focus on the fundamental problems of sorting and selection, where the true ordering of the elements changes slowly. We provide algorithms with performance close to the optimal in expectation and with high probability.

1 Introduction

In the classic paradigm, an algorithm received all the input at the start of the computation and computed a function of that input. As computing became more interactive, researchers developed the theory of online algorithms, focusing on the tradeoff between the timely availability of the input and the performance of the algorithm. In this paper we study another important aspect of online, interactive computing: computing and maintaining global information on a data set that is constantly *changing*. While algorithms and models to study dynamic data have been in vogue, our work formulates and studies a new model of computing in the presence of constantly changing data.

For concreteness we present our work through one specific motivation, the popular online voting website Bix (bix.com), owned by Yahoo!; this partially inspired us to study the particular problem of sorting. We comment later on more general applications. The Bix website hosts online contests for various themes such as the most entertaining sport or the most dangerous animal or the best presidential nominee, in which users vote to select the best amongst a pre-specified set of candidates. For a given contest, Bix displays a pair of candidates to a user visiting the website and asks the user to rank-order this pair. As the contest progresses, Bix aggregates all the pairwise comparisons

⋆ Part of this work was done while the author was at Yahoo! Research.

⋆⋆ Supported in part by NSF award DMI-0600384, ONR Award N000140610607, and Yahoo! Faculty Research Grant.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 339–350, 2009.

provided by users to pick the leader (or the top few leaders) of the contest thus far; the goal is to reflect the current aggregated opinion as faithfully as possible. For simplicity, we will ignore issues such as malicious user behavior and assume each user is able to compare any pair of candidates. In fact, we will assume something more general: each user has access to the global total order ("the public opinion") and when Bix shows a pair of candidates, the user consults this total order to rank-order the given pair.

There are two factors that make this setting both interesting and challenging. First, as the contest progresses, users' voting patterns might change, perhaps slowly, at an aggregate level. This can be caused by an intrinsic shift in public opinion about the candidates or factors external to the contest. While one cannot assume there is a fixed total order that the contest is trying to uncover, it is reasonable to assume that the total order changes slowly over time. Second, whenever a user visits their website, Bix has to choose a pair of candidates to show to the user in order to elicit the comparison. A visiting user is thus a valuable resource and hence Bix has to judiciously utilize this by showing a pair of candidates that yields the most value. Note that this is not a trivial problem: for example, it is not hard to show that asking the user to rank a random pair of candidates is quite "wasteful" and leads to considerably weaker guarantees.[1]

One way to model the above scenario is as follows. We have a set of n elements and an underlying total order π^t, at time t, on the elements. The ordering slowly changes over time and we model the slow change by requiring that the change from π^t and π^{t+1} is local. The goal is to design an algorithm that, at any point in time, tracks the top few elements of the underlying total order or more generally, maintains a total order $\tilde{\pi}^t$ that is close to π^t. The only capability available to the algorithm is pairwise comparison probes: at any time t, given one or more pair of elements, it can obtain the pairwise ranking of them according to the underlying total order currently in effect, (i.e., π^t). Clearly, there is a tradeoff between the number of probes that can be made at time t and the quality of $\tilde{\pi}^t$ (e.g., if the number of probes is large enough, then $\tilde{\pi}^t = \pi^t$ is easily achievable.)

Another motivation for the sorting problem is that of ranking in settings such as web search, recommendation systems, and online ad selection. A significant factor in ranking is the use of historic data. However, what may have been a good ranking in the past may not remain so perpetually, and the ranking changes are typically gradual over time (e.g., the query "vacation spots" might connote differently depending on the time of the year). The ranking system would like to track the changing perception of ranking by selecting what feedback (in the form of clicks) to request from the user. In addition to the above applications, which are mostly in the Internet domain, the problem has applications in sociology under the topic of the method of "paired comparisons" in the measurement of social values [5, Ch. 7].[2]

Of course, except for the aforementioned motivations for the sorting problem, similar issues arise in scenarios other than sorting. Consider, for example, a web crawler, whose goal is to track the highest quality pages on the web. The notion of quality, however, is (slowly) time-varying and the crawling algorithm, which is usually

[1] In the language of the model defined in Section 2, this algorithm leads to a guarantee of $O(n^2)$ for the Kendall tau distance (only a constant factor better than an oblivious algorithm that always outputs the same ranking), whereas we are able to achieve $O(n \ln \ln n)$.

[2] We thank Matthew Salganik for pointing out this application.

resource-constrained, has only limited access to the web graph at any point in time. The goal of the crawler would then be to track pages whose quality is reasonably close to the current best. Another graph application is maintaining routing tables with fastest (least congested) routes. The load on routes changes gradually, and the router receives new information on route's load only when a packet is sent along that route. Yet another setting can be that of a company that wants to track popular social network users with lots of friends, so as to use this information for viral marketing. Social networking systems such as Facebook allow to query and find the contacts of a given user (unless the user explicitly disallows) but limit the number of queries so as to prevent abuse. Overall, our setting is fairly general and can capture real-life scenarios such as continually updated remote databases, hashing, load balancing, polling, etc.

A general framework. The nature of the problems described above suggests the following general framework to study dynamic data. Let $\mathcal{U}$ and $\mathcal{V}$ be (possibly infinite) universe of objects. Let $f : \mathcal{U} \rightarrow \mathcal{V}$ be a function. Let $d : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}^+$ and $d' : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}^+$ be pairwise distance functions. $U^t \in \mathcal{U}$ is the object at time t while $V^t \in \mathcal{V}$ will be the estimate of the output of function f at time t.

(1) We have an implicit sequence of objects $U^1, U^2, \ldots$ such that $d(U^t, U^{t+1})$ is small, that is, the object changes slowly over time. The change can be arbitrary, or stochastic (which is the case that we consider in this paper).

(2) At each t, portions of the object U^t can be accessed by a certain number of probes.

(3) The goal is to output a sequence $V^1, V^2, \ldots$ such that for each t, $d'(f(U^t), V^t)$ is small, that is, we have a good approximation to the function of the true object at each point in time.

In the case of the Bix sorting problem that is the main focus of this paper, $\mathcal{U} = \mathcal{V} = S_n$, the set of permutations on n elements, $d = d'$ is the Kendall tau distance, and f is the identity function. For the selection problems we have that $\mathcal{V}$ is the set of elements, d' is the absolute rank difference between two elements, and f is an element. The slow changing of the objects in (1) is captured by permitting, say, only pairwise exchanges (corresponding to Kendall distance of 1) and the access to the object in (2) is captured by rank-ordering a given pair of elements according to the current total order. Even though in this paper we only focus on ranking and selection problems, this framework applies to many other settings as well, such as graph algorithms [1].

Related work. Models for dealing with dynamic and uncertain data have been extensively studied in the algorithmic community, from various points of view. This includes the multi-arm bandit algorithms that deal with explore-exploit tradeoffs, online algorithms that deal with future information, dynamic graph algorithms that deal with fast updates in response to graph changes, data stream algorithms that deal with limited computational resources such as space, stochastic optimization algorithms, etc. However, none of these captures the two crucial aspects of the above scenario: the slow changing of the underlying object and the probe model of exposing only a limited portion of the object to an algorithm. We are only aware of one other work that studies similar tradeoffs, namely the work of Slivkins and Upfal [4], which studies them in the more restricted setting of the multi-armed bandit model.

Our results. For the problem of maintaining a sorted order using a single probe at each time step when the permutation changes slowly and randomly and where the notion of distance is Kendall tau (number of pairwise disagreements), we give an algorithm that guarantees that for every time step t, the distance between the underlying true ordering and the ordering maintained by the algorithm is at most $O(n \ln \ln n)$, in expectation and with high probability. This builds upon an algorithm that has a distance guarantee of $O(n \ln n)$, in expectation and with high probability. We also show an $\Omega(n)$ lower bound on the expected distance between the true ordering and the order maintained by any algorithm.

To show the upper bound result, we first develop an algorithm that is based on periodically running the quicksort algorithm on the data. We use quicksort-specific properties to show that this algorithm can guarantee a distance of $O(n \ln n)$. We then give a more sophisticated algorithm that runs a copy of the above quicksort-based algorithm in parallel with multiple copies of faster though less accurate "local quicksorts." These local quicksorts will be able to give us the desired distance guarantee of $O(n \ln \ln n)$ in the first few runs; however, their weakness is that they could accumulate the errors and lead to considerably worse distance guarantees later. This weakness is overcome by occasionally resetting the algorithm using the slower quicksort, which is run in parallel.

We then consider selection problems: finding an element of a given rank. We provide algorithms that track the target elements to within distance 1. The basic idea is similar to the one we used for sorting: we adapt a static algorithm to the online setting by repeated executions. Furthermore, to ensure that the result returned is always close to the true value, we decompose the algorithm into two processes that are executed independently and in parallel, where the slower process prepares the data structures that the faster process uses over and over to compute the output. For the special case of finding the minimum element, we give a simpler algorithm by modeling the evolution of the process as a Markov chain.

2 Sorting Dynamic Elements

Consider a set $U = \{u_1, \dots, u_n\}$. Throughout most of this paper, our focus is on the problem of sorting the elements of U. In a static setting, where the correct ordering of the elements of U is given by a permutation π, there are numerous well-known sorting algorithms that can find the permutation π after comparing $O(n \ln n)$ pairs in U [2]. We are interested in a dynamic setting, where the true ordering π changes over time. To make this precise, consider a discretized time horizon with time steps indexed by positive integers. Let π^t be the true ordering at time t. We assume that the true ordering changes gradually, and we model this by assuming that for every $t > 1$, π^t is obtained from π^{t-1} by swapping a random pair of *consecutive* elements.

Our objective is to give an algorithm that can estimate the true ordering π^t. Unlike the familiar notion of algorithms that terminate in finite time, the algorithms we study run for ever; we often refer to them as *protocols*. In every time step t, the algorithm can select two elements of U to compare. The ordering of these two elements according to π^t is given to the algorithm, and then the algorithm computes an estimate $\tilde{\pi}^t$ of the true ordering. The algorithm has memory, that is, it is allowed to store any information, and the information will be carried over to the next time step. Note that our definition assumes that the rate of comparisons performed by the algorithm is equal to the rate of change in the true ordering (i.e., one swap and one comparison probe per time step). This assumption is merely for simplicity: all our results work in the more general setting where corresponding to every change in the true ordering, the algorithm is allowed to perform a number of comparisons given by a parameter β (β can be more or less than 1); the proofs are essentially exactly the same, with the bounds multiplied by functions of β. Furthermore, notice that we did not impose any constraint on either the amount of memory required by the algorithm or its running time. While such constraints seem natural in practice, it turns out that the running time and the memory are not major concerns, at least for the algorithms that we propose in this paper. Also, we need to specify whether the algorithm knows the initial ordering π^1. For convenience, we assume that the algorithm knows π^1, although our results hold without this assumption as well.[3]

Notice that unlike in the static setting where the algorithm can find the permutation π after finite time, in the dynamic setting the algorithm can never expect to find the exact true ordering π^t. Therefore, we need a way to measure how close the estimate is to the true ordering. For this purpose, we use the classical *Kendall tau* distance function between permutations. For a permutation π we write $x <_\pi y$ if x is ordered before y according to permutation π. The Kendall tau distance $\mathrm{KT}(\pi_1, \pi_2)$ between permutations π_1 and π_2 is defined as follows:

$$\mathrm{KT}(\pi_1, \pi_2) = |\{(x, y) : x <_{\pi_1} y \wedge \ y <_{\pi_2} x\}| \, .$$

The maximum Kendall tau distance between two permutations (and in fact the distance between two random permutations) is $O(n^2)$. In fact, no algorithm can guarantee that in every time step the distance between π^t and $\tilde{\pi}^t$ is less than $O(n)$ (Section 2.1). Our main result in Section 2.3 shows that there is an algorithm that can guarantee with high probability that this distance is at most $O(n \ln \ln n)$. We start with an easier result of $O(n \ln n)$ in Section 2.2, which will be used in our main result.

2.1 Lower Bound

We first prove an $\Omega(n)$ lower bound on the expected Kendall tau distance between the estimated order computed by any algorithm for our problem and the actual order at any time t.

Theorem 1. *For every $t > n/8$, $KT(\tilde{\pi}^t, \pi^t) = \Omega(n)$ in expectation and whp.* [4]

Proof. Consider the time interval $I = [t - n/8, t]$. Let B be the set of items involved in any comparison by the algorithm in this interval, $|B| \leq n/4$. Let $\bar{B} = U \setminus B$. At any time $\tau \in I$, the elements of B are adjacent in π^τ to up to $n/2$ elements in $\bar{B}$. Thus, for any $\tau \in [t - n/8, t]$, there are at least $n/4$ pairs of adjacent elements of $\bar{B}$ in π^τ, each of these pairs is swapped with probability c/n, where $c > 0$ is a constant. Thus, during the interval $[t - n/8, t]$ the expected number of pairs in $\bar{B}$ that are swapped

[3] We only need to be careful to require $t \geq n \ln n$ in our upper bounds (Theorems 2 and 3) if the algorithm does not know π^1.

[4] We say that an event holds "with high probability", abbreviated whp., if it holds with probability at least $1 - n^{-c}$ for some constant c, for sufficiently large n.

is $(c/32)n$. The expected number of pairs that are swapped and then swapped back is $O(c^2n^2/n^2) = O(1)$. Since no element of $\bar{B}$ is involved in any comparison the algorithm cannot identify swapped between pairs in $\bar{B}$, implying the result. □

2.2 Algorithm with $O(n \ln n)$ Distance Guarantee

In this section we first give an algorithm that guarantees the following: for every time step t, the distance between the orderings π^t and $\tilde{\pi}^t$ is $O(n \ln n)$, with high probability. We will use this result in the next section to get an improved bound of $O(n \ln \ln n)$.

The algorithm proceeds in phases, where each phase consists of $O(n \ln n)$ time steps (in expectation and whp.). In each phase, the algorithm runs a randomized quicksort algorithm to sort all elements. At any time step, the algorithm outputs the ordering that is obtained at the end of the *last* phase. Notice that since this algorithm outputs the same permutation for $O(n \ln n)$ steps, it cannot provide a distance guarantee better than $O(n \ln n)$. The following theorem shows that the distance guarantee of this algorithm is in fact $\Theta(n \ln n)$ whp.

Theorem 2. *For every t, $\mathrm{KT}(\tilde{\pi}^t, \pi^t) = O(n \ln n)$ in expectation and whp.*

Before proving the above theorem, we note that in our algorithm, the quicksort algorithm *may not* be replaced by an arbitrary $O(n \ln n)$ sorting algorithm. The reason being, in our setting, the algorithm can receive inconsistent data (since the true ordering is changing), and such inconsistencies can lead to large errors in general. In the case of quicksort, we will use its specific properties to argue that the inconsistencies can result in only a small number of *additional* errors (these errors will correspond to the set B in the following proof).

Proof (of Theorem 2). Consider one phase of the algorithm from time t_0 to t_1. We have that $t_1 - t_0 = \Theta(n \ln n)$, in expectation and whp.

To bound the Kendall tau distance we have to bound the number of pairs (u_i, u_j) that are ordered differently in the two permutations $\tilde{\pi}^t$ and π^t. We divide these pairs into two disjoint sets, A and B, where the set A contain the pairs for which the algorithm's order at time t_1 is in accordance with the true ordering at some time point $t \in [t_0, t_1)$:

$$A = \{(u_i, u_j) \mid u_i <_{\tilde{\pi}^{t_1}} u_j, u_i >_{\pi^{t_1}} u_j, \exists t \in [t_0, t_1) \text{ s. t. } u_i <_{\pi^t} u_j\},$$

and the set B contains the pairs for which there was a disagreement between the algorithm's order estimate (at time t_1) and the true order throughout the execution of the algorithm in this phase:

$$B = \{(u_i, u_j) \mid u_i <_{\tilde{\pi}^{t_1}} u_j, \forall t \in [t_0, t_1)\ u_i >_{\pi^t} u_j\}.$$

Since $\mathrm{KT}(\tilde{\pi}^t, \pi^t) = |A \cup B| = |A| + |B|$, Lemmas 1 and 2 will complete the proof. □

First we bound the cardinality of A by the running time of the algorithm.

Lemma 1. *$|A| = O(n \ln n)$ in expectation and whp.*

Proof. For the set A, note that if we let A' be the set of pairs for which the true order changed in $[t_0, t_1)$, that is,

$$A' = \{(u_i, u_j) \mid u_i <_{\pi^{t_1}} u_j, \exists t \in [t_0, t_1) \text{ s. t. } u_i >_{\pi^t} u_j\},$$

then we have that $A \subseteq A'$. Now notice that since the true order of the pair (u_i, u_j) was swapped during $[t_0, t_1)$, it has to be the case that at some point in $[t_0, t_1)$, the pair (u_i, u_j) was chosen to swap. Since only one pair swaps its ordering at each timestep and since $t_1 - t_0 = O(n \ln n)$ in expectation and whp., we have that $|A| \leq |A'| \leq t_1 - t_0 = O(n \ln n)$ in expectation and whp. □

For the set B the counting is more involved. By definition, for a pair $(u_i, u_j) \in B$ we have that $u_i > u_j$ according to the true ordering during $(t_0, t_1]$, however, at t_1 the algorithm concluded otherwise. This means that during one of the recursive calls of the quicksort algorithm, elements u_i and u_j belonged to the same subarray that was then sorted, a pivot element u_k was chosen ($u_k \neq u_i, u_j$), and after element u_k was compared with all the elements of the subarray, the result was $u_i < u_k$ and $u_j > u_k$. For this to have happened, the element u_k would have to be swapped with *each* of the elements u_i and u_j at least once while it was a pivot. After the element u_k terminates being a pivot, the algorithm's perception of the ordering between u_i and u_j does not change. (Note that the above arguments crucially rely on the fact that the algorithm is quicksort.)

From the previous discussion we see that if we can bound the number of swaps of the pivot elements during the period they were acting as pivots, then we will be able to bound the number of pairs in the set B. Since the probability that a pivot element is chosen at a given time step is small (at most $2/n$), we expect the set B to be small. We prove this formally below.

Lemma 2. $|B| = O(n \ln n)$ *in expectation and whp.*

Proof. We will charge the error due to pair (u_i, u_j) to the corresponding pivot u_k. Let X_i be the number of steps that element u_i was a pivot during $[t_0, t_1)$; note that $X_i \leq n$. Let $\mathcal{E}$ be the event that

$$\sum_{i=1}^{n} X_i \leq c_0 n \ln n, \tag{1}$$

for some constant $c_0 > 0$. Since the running time of quicksort is $O(n \ln n)$ in expectation and whp., $\mathcal{E}$ holds whp. Also, the running time of quicksort is $O(n^2)$ in the worst case. The event $\neg\mathcal{E}$ will only contribute a negligible (inverse polynomial) amount to the calculations below; therefore, for ease of exposition, we will condition on $\mathcal{E}$ being true for the rest of the proof.

Since $X_i \leq n$ and $\sum X_i \leq c_0 n \ln n$, by convexity, $\sum X_i^2$ is maximized if $c_0 \ln n$ of the X_i's are equal to n and the rest are equal to 0. Hence,

$$\sum_{i=1}^{n} X_i^2 \leq c_0 n^2 \ln n. \tag{2}$$

Let Y_i be the number of steps that element i was a pivot element and it was chosen to swap. Given X_i, we have that $Y_i \sim \text{Binomial}(X_i, p)$ where $p = 2/n$ (with the exception of the case that the pivot is or becomes the first or last element in the order, in which case $p = 1/n$). We argued earlier that for the pair (u_i, u_j) to become misordered, the corresponding pivot was swapped with both u_i and u_j. Therefore, if a pivot swapped Y_i times, then it could have led to at most Y_i^2 misordered pairs. We can then bound the number of pairs in the set B by $S \triangleq \sum_{i=1}^{n} Y_i^2 \geq |B|$. The proof is complete if we upper bound $\mathbf{E}[S]$. Now,

$$\begin{aligned}\mathbf{E}[S] &= \mathbf{E}\left[\sum_{i=1}^{n} Y_i^2\right] = \mathbf{E}\left[\mathbf{E}\left[\sum_{i=1}^{n} Y_i^2 \middle| X_i\right]\right] = \mathbf{E}\left[\sum_{i=1}^{n} \mathbf{E}\left[Y_i^2 | X_i\right]\right] \\ &= \mathbf{E}\left[\sum_{i=1}^{n} (\mathbf{Var}\left[Y_i | X_i\right] + \mathbf{E}[Y_i | X_i]^2)\right] = \mathbf{E}\left[\sum_{i=1}^{n} (X_i p(1-p) + X_i^2 p^2)\right] \\ &= \mathbf{E}\left[p(1-p)\sum_{i=1}^{n} X_i + p^2 \sum_{i=1}^{n} X_i^2\right] \overset{(1),(2)}{\leq} (2c_0(1-p) + 4c_0)\ln n \leq c_1 \ln n,\end{aligned}$$

for some constant $c_1 > 0$.

To bound the probability that the set B is large, first note that given X_i's, the Y_i's are independent binomial random variables. We apply Azuma's inequality and finish the proof.[5] For some sufficiently large constant $c_2 > 0$, we have

$$\mathbf{Pr}(S - \mathbf{E}[S] > c_2 n \ln n) \leq \exp\left(-\frac{2c_2^2 n^2 \ln^2 n}{\sum_{i=1}^{n} X_i^2}\right) \overset{(2)}{\leq} n^{-2c_2^2/c_1}.$$

The following lemma is also proved similarly and will be used later. The proof will appear in the full version of the paper.

Lemma 3. *Given an element u_i, the number of pairs (u_i, u_j) that the permutations π^{t_1} and $\tilde{\pi}^{t_1}$ rank differently is bounded by $\tilde{c} \ln n$ in expectation and whp., for some constant $\tilde{c}$ and sufficiently large n.*

2.3 Main Result

Now we present a more complicated protocol that maintains an error of $O(n \ln \ln n)$. The main idea is that after the quicksort execution, which due to its running time has as a result an error of $O(n \ln n)$, the rank of each element in the algorithm's estimate is within $O(\ln n)$ of its actual rank. Thus, by performing several ($O(n/\ln n)$) local sorts in blocks of size $m = \Theta(\ln n)$ we can correct the ordering. The total running time is $O(n/\ln n) \cdot (\ln n) \ln \ln n$, therefore after all the sorts terminate the total error will be bounded by $O(n \ln \ln n)$.

[5] The following is a consequence of Azuma's inequality [3]. Assume that $0 < X_i < d_i$ are independent random variables, and let $S = \sum_{i=1}^{n} X_i$. Then

$$\mathbf{Pr}(S - \mathbf{E}[S] > \lambda) \leq \exp\left(-2\lambda^2 / \sum_{i=1}^{n} d_i^2\right).$$

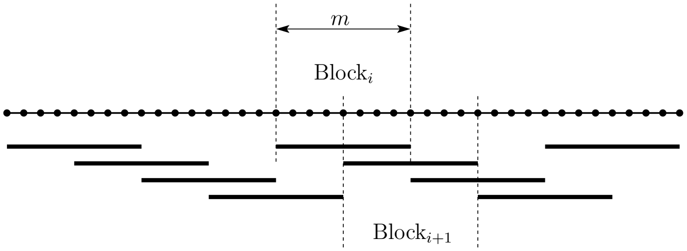


Fig. 1. The set of elements ordered according to $\tilde{\pi}^{t_0}$, and the partition into blocks

There are some issues that we have to address though. First, since elements might have moved to neighboring blocks, we make the blocks overlapping thus allowing the comparison of all neighboring elements (see Figure 1). First we sort the first m elements. From the resulting order we maintain the first $m/2$ of the elements. The second half of the block is sorted along with the next $m/2$ elements. Again we maintain the first $m/2$ elements and proceed in the same way.

Second, while we would like to sequentially execute a full set of local quicksorts after the termination of the previous one so as to maintain the error of $O(n \ln \ln n)$, eventually elements will move far. Thus it is necessary to occasionally execute a full quicksort to recover the global order. The problem, however, is that during the execution of the global quicksort the error will become $n \ln n$. Therefore, we use the following idea: execute two sets of quicksorts independently. During the odd timesteps we execute a regular quicksort, and after its termination we restart, as in Section 2.2. The previous analysis applies to this case as well with the difference that in every step there are two pairs whose order swaps. During the even steps, we execute the set of $\Theta(n/\ln n)$ quicksorts on overlapping blocks of length $m = \Theta(\ln n)$. The input to the set of quicksorts is the output of the last full quicksort that has terminated. After the termination of the set of quicksorts we rerun them, again with the same input. The two processes are executed independently with their own data structures. In every time step, the "output" of the protocol is the output of the latest successfully completed set of quicksorts. In Figure 2 we present a schematic representation of the algorithm. The proof of the following theorem will appear in the full version of the paper.

Theorem 3. *For every t,* $\mathrm{KT}(\tilde{\pi}^t, \pi^t) = O(n \ln \ln n)$ *in expectation and whp.*

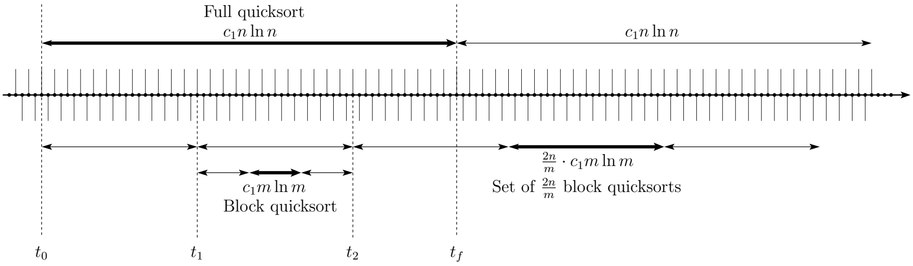


Fig. 2. The periods of the execution of the sorting algorithm

3 Selection Problems

As we mentioned earlier, the dynamic data setting can capture several scenarios. In this section we illustrate this by providing two more examples. First we show a simple algorithm for finding the element with minimum rank; for a fairly realistic application of this setting, consider the social-network example presented in the introduction. We then present a more general algorithm that can be used to find the element of a given rank. By combining this algorithm with the previous result on sorting, one can find the top-k ranked elements.

3.1 Finding the Minimum

Assume the same dynamic perturbation model as before where each pair swaps in every time step with probability $\alpha/(n-1)$, where $\alpha > 0$ is a constant ($\alpha = 1$ in the simplest case). Instead of sorting all the elements we only want to estimate the smallest element. The following simple algorithm outputs at any given step an element that is either the minimum or very close to minimum. The algorithm maintains the current minimum estimate m and in every step compares it with an element u_i chosen uniformly at random from all the elements. If $u_i < m$, it replaces m with u_i. The basic idea to prove the following theorem is to model the process as a Markov chain (details omitted in this version).

Theorem 4. *Let m_t be the rank of the estimate at time t. In the steady state $\mathbf{Pr}(m_t \geq i) \leq \left(\frac{\alpha}{1+\alpha}\right)^i$, and $\mathbf{E}[m_t] = 1 + \alpha$.*

3.2 Finding the Element of a Given Rank

In this section we give an algorithm for solving the problem of finding the element of rank k for $k = 1, 2, \ldots, n$. Given k, our goal is to find an element u_i that minimizes the distance $|\pi^t(u_i) - k|$, where $\pi^t(u_i)$ is the rank of u_i at time t. For $k = 1$ the problem is that of finding the minimum, while for $k = \lceil n/2 \rceil$ the problem is that of finding the median. To make the exposition clearer we present the case of the median; the algorithm and the proof can be easily generalized for any k. Figure 3 is a dynamic version of the median algorithm in [3], with a few modifications to adapt it to our dynamic setting. As in the case of the elaborate sorting algorithm, we run two algorithms in an interleaved manner. In the odd steps we prepare a set C that will contain the median in any step of the next execution of the while loop. In the even steps we use the set C computed in the previous step to output the median estimate. During a sorting phase of the set C, the output estimate is the element $\tilde{\mu}$ computed in the previous run of the sorting. Let t_0 be the last time point of the first period. We show that the difference in the rank of the element returned by the algorithm is negligible.

Theorem 5. *Let $\tilde{\mu}$ be the output of algorithm Median. For every time $t \geq t_0 = O(n)$ we have that $\mathbf{Pr}\left(\left|\pi(\tilde{\mu}) - \frac{n}{2}\right| = 0\right) \geq 1 - o(1)$, and $\mathbf{E}\left[\left|\pi(\tilde{\mu}) - \frac{n}{2}\right|\right] = o(1)$.*

```
1.  Algorithm Median($U$)
2.    Input: A set of elements $U$
3.    while (true)
4.      Execute in odd steps:
5.      Pick a (multi-)set $R$ of $\frac{n}{36 \ln n}$ elements from $U$ chosen independently uniformly at
          random with replacement
6.      quicksort($R$)
7.      Let $d$ be the $(\frac{n}{72 \ln n} - \sqrt{n})$th smallest element in the sorted set $R$
8.      Let $u$ be the $(\frac{n}{72 \ln n} + \sqrt{n})$th smallest element in the sorted set $R$
9.      By comparing every element in $U$ to $d$ and $u$, compute the set
          $C = \{x \in U : d \leq x \leq u\}$ and the number $\ell_d = |\{x \in U : x < d\}|$
10.     Execute in even steps using the set $C$ computed last
11.     quicksort($C$)
12.     $\tilde{\mu} \leftarrow (\lfloor n/2 \rfloor - \ell_d + 1)$th element in the sorted order of $C$
13.   end while
```

Fig. 3. Algorithm for computing the median

Proof. The proof is based on the proof of the static version presented, for example, in [3]; we will outline only the modification specific to the dynamic case.

We partition time into periods of length $\Theta(n)$, where each period corresponds to a full execution of steps 4–9 in Figure 3. (Executing the full set of steps 4–9 (odd time steps) requires time $\Theta(n)$ while the set of steps 10–12 (even time steps) requires $\Theta(\sqrt{n} \ln^2 n)$, whp.) In the odd time steps of a period we compute a set C to be used to compute the median in the next period. In the even time steps we use the set C computed in the previous period.

We first note that the length of each period is linear with high probability, therefore in a given period the rank of a given element (and in particular that of the median) changes by a constant in expectation. Furthermore, with high probability no element moves more than $c \ln n$ places during a period, for some constant c.

The analysis of the static case as presented in [3] reduces to proving that the following two facts (adapted to our case) hold whp.:

1. The set C computed at a given period contains all the elements that are medians during the next period.
2. $|C| = O(\sqrt{n} \ln n)$ whp.

With a similar argument as the one used in [3] and taking into account that the rank of element d during two periods does not change more than $2c \ln n$ whp., we have that in order to maintain $\pi^t(d) < \lceil n/2 \rceil$ it suffices that at most $\frac{n}{72 \ln n} - \sqrt{n}$ samples in R had rank smaller than $\frac{n}{2} - 2c \ln n$, when they were selected. We define $X_i = 1$ if the ith sample had rank smaller than $\frac{n}{2} - 2c \ln n$, and 0 otherwise. Then we have that $\mathbf{Pr}(X_i = 1) = \frac{1}{2} - \frac{2c \ln n}{n}$ and $\mathbf{E}\left[\sum X_i\right] = \frac{n}{72 \ln n} - \frac{c}{18}$. We can apply a Chernoff bound and obtain:

$$\mathbf{Pr}\left(\sum_{i=1}^{|R|} X_i < \frac{n}{72 \ln n} - \sqrt{n}\right) = \mathbf{Pr}\left(\sum X_i - \mathbf{E}\left[\sum X_i\right] < \frac{c}{18} - \sqrt{n}\right)$$

$$\leq e^{-\frac{72 \ln n}{n}\left(\sqrt{n} - \frac{c}{18}\right)^2} \leq \frac{1}{n^3}.$$

A similar argument shows that we maintain that $\pi^t(u) > \lceil n/2 \rceil$ throughout the execution of the entire period, therefore, the set C created at step 8 will contain whp. all elements that are medians in the next period.

Next we need to show that $|C| = O(\sqrt{n} \ln n)$. The argument must take into account that the elements change ranks and is similar to the one just presented. The reader can refer to [3] for the type of events that need to be defined and we do not repeat here the details. Thus C can be sorted in $O(\sqrt{n} \ln^2 n)$ time.

We have now established that whp. $|C| = O(\sqrt{n} \ln n)$ and that it contains all elements that are medians during the next period. Since sorting in step 8 takes $O(\sqrt{n} \ln^2 n)$ steps, the probability that either the median at the beginning of a sorting phase, or the $O(\ln n)$ pivots that it is compared to during the sorting move during the sorting phase is bounded by $O(\ln^3 n/\sqrt{n})$. Thus, with probability $1 - O(\ln^3 n/\sqrt{n})$ the sorting returns the correct median at that step. The probability that the median change place during the next sorting round (before a new median is computed) is bounded by $O((\sqrt{n} \ln^2 n)/n)$. Thus, at any given step, with probability $1 - o(n^{-1/2+\epsilon})$ the algorithm returns the correct median. The expectation result is obtained by observing that when the output is not the correct median, its distance to the correct median is with high probability $O(\ln n)$.

□

4 Conclusions

In this paper we study a new computational paradigm for dynamically changing data. This paradigm is rich enough to capture many natural problems that arise in online voting, crawling, social networks, etc. In this model the data gradually changes over time and the goal of an algorithm is to compute some property of it by probing, under the constraint that the amount of access to the data at each time step is limited. In this simple framework, we consider the fundamental problems of sorting and selection, where the true ordering slowly changes over time and the algorithm can probe the true ordering once each time step using a pair of elements it chooses. We obtain an algorithm that maintains, at each time step, an ordering that is at most $O(n \ln \ln n)$–Kendall-tau distance away from the true ordering, with high probability. For selection problems, we provide algorithms that track the target element to within distance 1. Revisiting classical algorithmic problems in this paradigm will be an interesting direction for future line of research [1].

References

1. Anagnostopoulos, A., Kumar, R., Mahdian, M., Upfal, E.: Dynamic graph algorithms (manuscript, 2009)
2. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press, Cambridge (2001)
3. Mitzenmacher, M., Upfal, E.: Probability and Computing. Cambridge University Press, Cambridge (2005)
4. Slivkins, A., Upfal, E.: Adapting to a changing environment: The Brownian restless bandits. In: Proc. 21st Annual Conference on Learning Theory, pp. 343–354 (2008)
5. Thurstone, L.L.: The Measurement of Values. The University of Chicago Press (1959)

Maximum Bipartite Flow in Networks with Adaptive Channel Width

Yossi Azar[1], Aleksander Mądry[2,⋆], Thomas Moscibroda[3],
Debmalya Panigrahi[2,⋆⋆], and Aravind Srinivasan[4,⋆⋆⋆]

[1] Microsoft Research, Redmond, WA 98052 and Tel Aviv University, Tel Aviv, Israel
azar@tau.ac.il

[2] Massachusetts Institute of Technology, Cambridge, MA 02139
{madry,debmalya}@mit.edu

[3] Microsoft Research, Redmond, WA 98052
moscitho@microsoft.com

[4] Dept. of Computer Science and Institute for Advanced Computer Studies,
University of Maryland, College Park, MD 20742
srin@cs.umd.edu

Abstract. Traditionally, combinatorial optimization problems (such as maximum flow, maximum matching, etc.) have been studied for networks where each link has a fixed capacity. Recent research in wireless networking has shown that it is possible to design networks where the capacity of the links can be changed *adaptively* to suit the needs of specific applications. In particular, one gets a choice of having *few* high capacity outgoing links or *many* low capacity ones at any node of the network. This motivates us to have a re-look at the traditional combinatorial optimization problems and design algorithms to solve them in this new framework. In particular, we consider the problem of *maximum bipartite flow*, which has been studied extensively in the traditional network model. One of the motivations for studying this problem arises from the need to maximize the throughput of an infrastructure wireless network comprising base-stations (one set of vertices in the bipartition) and clients (the other set of vertices in the bipartition). We show that this problem has a significantly different combinatorial structure in this new network model from the classical one. While there are several polynomial time algorithms solving the maximum bipartite flow problem in traditional networks, we show that the problem is NP-hard in the new model. In fact, our proof extends to showing that the problem is APX-hard. We complement our lower bound by giving two algorithms for solving the problem approximately. The first algorithm is deterministic and achieves an approximation factor of $O(\log N)$, where there are N nodes in the network, while the second algorithm (which is our main contribution) is randomized and achieves an approximation factor of $\frac{e}{e-1}$.

[⋆] Research supported by a Fulbright Science and Technology Award, by NSF contract CCF-0829878 and by ONR grant N00014-05-1-0148.

[⋆⋆] Part of this work was done while visiting Microsoft Research, Redmond. Research supported in part by NSF contract CCF-0635286.

[⋆⋆⋆] Part of this work was done while visiting Microsoft Research, Redmond. Research supported in part by NSF ITR Award CNS-0426683 and NSF Award CNS-0626636.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 351–362, 2009.

1 Introduction

Combinatorial optimization has been an important tool in the analysis of computer networks. Traditionally, optimization problems such as maximum flow, maximum matching, etc. have been studied for networks where each link has a fixed capacity.[1] However, recent research in wireless networking has shown that it is possible to design networks where the capacity of the links can be changed *adaptively* to suit the needs of specific applications [4][19]. (We call this phenomenon *adaptive channel width.*) Specifically, wider communication channels increases channel capacity (a result that is also predicted by Shannon's capacity formula [20]), but reduces the transmission range thereby disconnecting distant nodes. Thus, one gets a choice of having *few* high capacity outgoing links or *many* low capacity ones at each node of the network. This motivates us to have a relook at traditional combinatorial optimization problems and design algorithms to solve them in this new framework. In this paper, we focus on the *maximum bipartite flow* problem in networks with adaptive channel width.

Consider a directed bipartite graph on vertices $X \cup Y$, where all the edges are directed from vertices in X to vertices in Y and have capacities. In addition each node in X and Y has a node capacity (corresponding to a budget for nodes in X and demand for nodes in Y). This is equivalent to adding a vertex s (called the *supersource*) which is connected by a directed edge[2] to each vertex in X with edge capacity corresponding to the node capacity; similarly, adding a vertex t (called the *supersink*) such that each vertex in Y is connected to it by a directed edge with the corresponding capacity. In the traditional model, the maximum flow permitted on an edge is equal to its capacity. The *maximum bipartite flow* problem then requires us to find the maximum flow from s to t subject to the capacity constraints. On the other hand, in networks with adaptive channel width, the edges from X to Y do not have fixed capacities; rather one needs to determine the *assigned capacity* at each node $x \in X$. Choosing a high assigned capacity disconnects x from some nodes in Y but produces a high capacity edge to the remaining nodes, while choosing a low assigned capacity implies that all edges from x to nodes in Y have low capacity. The goal is to choose assigned capacities such that the achievable flow in the resulting capacitated network is maximized.

The main practical motivation for studying this problem comes from the need to maximize throughput in infrastructure wireless networks. In particular, let X be the set of base-stations and Y be the set of clients in an infrastructure wireless network. Suppose that the base-stations are equipped with the ability to adapt channel width. We will show that the problem of maximizing throughput in such an infrastructure wireless network is identical to solving the maximum flow problem on bipartite graphs with adaptive channel width. We formally define our throughput maximization problem and its connection to the maximum bipartite flow problem below.

[1] In this paper, we will use the terms *bandwidth* and *capacity* interchangeably.

[2] In this paper, we will use the terms *directed edge*, *link* and *arc* interchangeably.

Problem Definition. We are given a set of base-stations $\mathcal{B}$ and a set of clients $\mathcal{C}$ with $|\mathcal{B}| = n$ and $|\mathcal{C}| = m$. Each base-station $B \in \mathcal{B}$ has a *budget* $\beta(B)$, which is the total capacity that the base-station can deliver to its clients. On the other hand, each client $C \in \mathcal{C}$ has a *demand* $\alpha(C)$, which is the total bandwidth it would like to be allocated from all the base-stations together.

For each base-station and client pair (henceforth, called a *base-client pair*) (B, C), there is a *critical capacity* $\eta(B, C)$, which corresponds to the maximum bandwidth of a link from B to C. To each base station $B \in \mathcal{B}$, the algorithm selects an *assigned capacity* $\tau(B)$ that determines the *capacity* of a link (B, C) (denoted by $\psi_\tau(B, C)$) as follows

$$\psi_\tau(B, C) := \begin{cases} \tau(B) & , \tau(B) \le \eta(B, C) \\ 0 & , \text{otherwise.} \end{cases}$$

Once the capacities of all links have been fixed, we want to find a *flow* $f(B, C)$ for each link (B, C) such that neither any link capacity is violated (*capacity constraint*), i.e.

$$f(B, C) \le \psi_\tau(B, C),$$

nor any base-station budget is violated (*budget constraint*), i.e.

$$\sum_{C \in \mathcal{C}} f(B, C) \le \beta(B).$$

The goal is to find the capacity assignment τ, and corresponding flow f that maximizes the sum of *satisfied demands* of all the clients, where the satisfied demand $\alpha_{\tau,f}(C)$ of a client C is given by

$$\alpha_{\tau,f}(C) = \min\big(\sum_{B \in \mathcal{B}} f(B, C), \alpha(C)\big).$$

Note that given any τ and f, there always exists a flow f' which satisfies the budget and capacity constraints, achieves the same value of total satisfied demand and additionally satisfies the following *demand constraints*,

$$\sum_{B \in \mathcal{B}} f(B, C) \le \alpha(C).$$

As a result, we will focus on flows that obey demand constraints along with budget and capacity constraints.

The benefit of this assumption is that our problem now corresponds to the maximum bipartite flow problem in networks with adaptive channel width. Recall that in this flow problem, we have two sets of nodes X and Y with edges directed from X to Y, along with a supersource s and a supersink t. To draw the correspondence, let X be the set of base-stations and Y the set of clients. The edge from s to any $x \in X$ (called a *budget arc*) has capacity $\beta(x)$, that from any $x \in X$ to any $y \in Y$ has critical capacity $\eta(x, y)$ and that from any $y \in Y$ to t (called a *demand arc*) has capacity $\alpha(y)$ (refer to Figure 1). We call

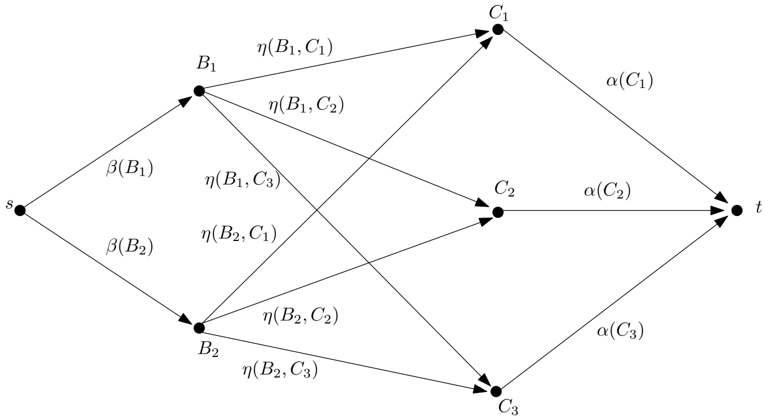


Fig. 1. The *augmentation graph* corresponding to an instance of the problem

this graph the *augmentation graph* of the given problem instance. Our task is to choose assigned capacities (i.e. the function τ) for vertices in X; this fixes the capacities of all the arcs from X to Y. Our goal is to choose τ so that the maximum flow in the resulting capacitated network is maximized.

Related Work. Classically, the maximum bipartite flow problem has been solved as a special case of the more general maximum flow problem on arbitrary graphs. Suppose the input graph $G = (V, E)$ has maximum flow of c from the source to the sink. Ford and Fulkerson gave the first algorithm for the maximum flow problem in the 1950s, which had a running time of $O(|E|c)$ [9]. Since then, several algorithms have been developed with better time bounds [6][7][8][11] finally culminating in an $\tilde{O}(|E| \min(|E|^{1/2}, |V|^{2/3}) \log c)$ algorithm due to Goldberg and Rao [10], which is currently the fastest known deterministic algorithm for maximum flow. It may be noted here that a substantial amount of work has also been done for developing randomized algorithms for maximum flow [14][16][15][17], but these algorithms apply only to undirected networks. On the other hand, the maximum bipartite flow problem with unit capacities (which is equivalent to *maximum bipartite matching*) can be solved in $O(|E|\sqrt{|V|})$ time [13].

To summarize the above discussion, the maximum bipartite flow problem in directed graphs with capacitated edges is solvable in polynomial time; however, there is no algorithm which solves this problem faster than in general directed graphs. As we will see, this is in sharp contrast to what we observe in networks with adaptive channel width. In such networks, the maximum bipartite flow problem is NP-hard (in fact, it is APX-hard); further, we give a randomized approximation algorithm achieving an approximation factor of $\frac{e}{e-1}$ which does not appear to extend easily to general directed networks.

We also briefly mention a related class of well-studied problems, namely *unsplittable* flow problems. In these problems, typically there are one or more pairs of source and sink vertices with specific demands, and the goal is to connect the source-sink pairs using paths such that the satisfied demand is maximized while not violating any capacity constraint. These problems are typically NP-hard, and several variants have been studied extensively [18][21][12] [2][3][5]. Interestingly, though we have a single source and a single sink, and flow is allowed to re-distribute arbitrarily at a node, the techniques we use to give an approximation algorithm for our problem bear similarities with the techniques usually used for solving unsplittable flow problems. Specifically, both problems use a suitable linear programming relaxation which is then rounded ensuring that certain cuts are large in the rounded solution.

Our Results. Our first claim is that the maximum bipartite flow problem in networks with adaptive channel width is APX-hard, i.e., it is unlikely that a polynomial-time algorithm can approximate the problem within a certain constant factor. Specifically, we describe an L-reduction from the APX-hard Maximum Bounded 3-Dimensional Matching problem (Max-3DM) to the channel width assignment problem. We prove the following theorem (details of the construction are omitted due to space constraints).

Theorem 1. *The maximum bipartite flow problem in networks with adaptive channel width is APX-hard.*

Our next contribution is a greedy combinatorial algorithm which achieves an approximation factor of $O(\log N)$, where $N = \max(m, n)$. The algorithm first categorizes links according to their critical capacity in geometrically spaced intervals. Now, observe that for any interval, we can set the assigned capacities at the nodes such that all the links in that interval have capacity equal to their critical capacities (while all other links have potentially no capacity at all). The algorithm needs to decide which interval to choose. For this purpose, a maximum flow algorithm is run on the entire graph, assuming that each link has capacity equal to its critical capacity. This outputs a flow on each link. The algorithm greedily chooses the interval which carries the greatest amount of flow on its links. The details of the algorithm are omitted due to space constraints.

Finally, our main result is a randomized algorithm for this problem.

Theorem 2. *There is a randomized algorithm for the maximum bipartite flow problem in networks with adaptive channel width that has an expected approximation factor of* $\frac{e}{e-1}$.

Our algorithm uses a linear programming relaxation of the problem. Recall that the celebrated Menger's theorem implies that maximum flow from s to t equals the minimum $s - t$ cut. So, an algorithm for the problem should aim to choose assigned capacities so as to maximize the minimum $s - t$ cut in the resulting capacitated network. Now, let us consider any linear programming formulation of the problem; such a fractional linear program can be interpreted as a polytope, where its optimal solution is a convex combination of the vertices of the polytope. Each vertex of the polytope represents a particular choice of assigned capacities and therefore, a particular capacitated graph (call them *vertex graphs*); these correspond to the integral solutions we will round our solution to. The natural linear program that we consider first simply ensures that for each cut, the convex combination of the values of the cut in the vertex graphs is large. However, since each vertex graph may have a different minimum $s{-}t$ cut, this does not guarantee that sizes of these minimum $s-t$ cuts are large. In fact, this linear program has an integrality gap of $\Omega(\log N/\log\log N)$. To overcome this problem, we design a more sophisticated linear program and a corresponding randomized rounding technique that ensures that the minimal $s-t$ cuts in the vertex graphs are large.

2 Algorithms for Maximum Bipartite Flow

As mentioned previously, a greedy combinatorial algorithm for our problem achieves an approximation factor of $O(\log N)$, where $N = \max(m, n)$.

2.1 A Linear Program

Our goal now is to improve upon this combinatorial algorithm. Without loss of generality, we may assume that the assigned capacity chosen at any base-station B in an optimal solution is one among the critical capacities of its outgoing edges, i.e. $\tau(B) \in \{\eta(B,C) : C \in \mathcal{C}\}$. If this is not the case, then the assigned capacity can be increased to the closest value from the set $\{\eta(B,C) : C \in \mathcal{C}\}$ without changing the flow on any link. This allows us introduce the *boolean capacity choice function* $p(B,C)$, which is 1 if $\tau(B) = \eta(B,C)$, and 0 otherwise. Clearly, for any base-station B, $p(B,C) = 1$ for exactly one client C (called the *choice constraint*). We also introduce another new notation, $\mathcal{C}_B(C)$ which represents the set of clients for which the critical capacity of their link to base-station B is less than that for client C, i.e.

$$\mathcal{C}_B(C) = \{C' \in \mathcal{C} : \eta(B,C') \leq \eta(B,C)\}.$$

A natural formulation of the problem is via the following integer linear program (ILP), where constraints (1), (2), (3) and (4) correspond to budget, demand, capacity and choice constraints respectively.

$$\texttt{maximize } \textstyle\sum_{B\in\mathcal{B}}\sum_{C\in\mathcal{C}} f(B,C) \texttt{ subject to}$$

$$\sum_{C\in\mathcal{C}} f(B,C) \leq \beta(B), \ \ \forall B \in \mathcal{B} \tag{1}$$

$$\sum_{B\in\mathcal{B}} f(B,C) \leq \alpha(C), \ \ \forall C \in \mathcal{C} \tag{2}$$

$$f(B,C) \leq \sum_{C'\in\mathcal{C}_B(C)} p(B,C')\eta(B,C'), \ \ \forall B \in \mathcal{B}, \ \ \forall C \in \mathcal{C} \tag{3}$$

$$\sum_{C\in\mathcal{C}} p(B,C) = 1, \ \ \forall B \in \mathcal{B} \tag{4}$$

$$p(B,C) \in \{0,1\}, \ \ \forall B \in \mathcal{B}, \ \ \forall C \in \mathcal{C} \tag{5}$$

$$f(B,C) \geq 0, \ \ \forall B \in \mathcal{B}, \ \ \forall C \in \mathcal{C}. \tag{6}$$

LP Relaxation. To make this ILP tractable, we relax constraint (5) and allow the function p to assume values between 0 and 1 (call this the *fractional program* or FLP). The natural interpretation is that $p(B, C)$ denotes the *goodness* of $\eta(B, C)$ as the assigned capacity of base-station B. Mathematically, it can be thought of as the probability with which $\eta(B, C)$ should be the assigned capacity of base-station B.

Unfortunately, it turns out that this natural linear programming relaxation fails to provide us with an approximation guarantee that is significantly better than the one achieved by the combinatorial algorithm. To understand why this is the case, let us note that for a given choice of values of $p(B, C)$, this linear program is solving the max-flow problem in the augmentation graph where the capacity u of a link (B, C) is the following: if the assigned capacity $\tau(B)$ at base-station B is chosen according to the probability distribution given by $p(B, C)$, then

$$u = \mathbb{E}[\psi_\tau(B, C)].$$

Therefore, by max-flow/min-cut duality, the approach used in the LP boils down to choosing $p(B, C)$ in such a way that the minimal expected capacity among all s-t-cuts in the augmented graph is maximized. Given some final choice of $p(B, C)$ computed by the linear program, it is tempting to round it by choosing assigned capacities according to $p(B, C)$, and then solving the max-flow problem in the resulting graph, hoping that the capacity of minimal cut will be close to the expected one. Clearly, when we focus on one particular cut, say the one that separates base-stations from clients, it will be true, but this does not necessarily mean that for all cuts, such a promise will hold simultaneously. It may happen that for the choice of assigned capacities that we obtain, it will always be the case that for part of the clients the capacity of links leading to them in the resulting graph will be much below the expectation, while for the other part it will excessively large, and this excess will be wasted due to the bottlenecks imposed by not large enough capacity of demand arcs for the respective clients. Thus, even if on expectation each client has reasonable capacity of links leading to it, the rounding procedure might not provide us with a particularly good solution. Therefore, our analysis of the approximation guarantee given by this LP would need to argue that with good probability all cuts are preserved up to some ratio, and in fact, using Chernoff bounds, we can prove that this is indeed true for the ratio $O(\log(m + n)/ \log\log(m + n))$. Unfortunately, we can show through an integrality gap example (omitted due to space constraints) that this unsatisfactorily large ratio is not only a shortcoming of our particular rounding procedure, but it is in fact all that we can achieve through *any* rounding algorithm for this LP.

2.2 An Alternative Linear Program

In this section, we will describe a more sophisticated ILP which overcomes the shortcomings of the previous ILP. Note that our goal is to choose assigned capacities such that the augmentation graph has a large maximum flow, or equivalently by Menger's theorem, a large minimum cut. The previous FLP ensures that each cut has large capacity in expectation and therefore the minimum among the expected capacities of the cuts is large; this however does not guarantee that the expected capacity of the minimum cut is large. We need this stronger guarantee from our LP. To achieve this goal, we design an LP which yields a family of flows corresponding to the different choices of assigned capacities, and ensures that the expected value of these flows is large. This clearly implies that the expected capacity of the minimum cut is large, and therefore provides stronger guarantees than the previous LP. Precisely, we consider the following ILP.

$$\texttt{maximize } \textstyle\sum_{B\in\mathcal{B}}\sum_{C',C\in\mathcal{C}} f_{C'}(B,C) \texttt{ subject to}$$

$$\sum_{C\in\mathcal{C}} f_{C'}(B,C) \le p(B,C')\beta(B),\ \ \forall B\in\mathcal{B},\ \ \forall C'\in\mathcal{C} \tag{7}$$

$$\sum_{B\in\mathcal{B}}\sum_{C'\in\mathcal{C}} f_{C'}(B,C) \le \alpha(C),\ \ \forall C\in\mathcal{C} \tag{8}$$

$$f_{C'}(B,C) \le \begin{cases} 0, & \text{if } \eta(B,C) < \eta(B,C') \\ p(B,C')\min\{\eta(B,C'),\alpha(C)\}, & \text{otherwise} \end{cases}$$
$$\forall B\in\mathcal{B},\ \ \forall C,C'\in\mathcal{C} \tag{9}$$

$$\sum_{C\in\mathcal{C}} p(B,C) = 1,\ \ \forall B\in\mathcal{B} \tag{10}$$

$$p(B,C)\in\{0,1\},\ \ \forall B\in\mathcal{B},\ \ \forall C\in\mathcal{C} \tag{11}$$

$$f_{C'}(B,C)\ge 0,\ \ \forall B\in\mathcal{B},\ \ \forall C,C'\in\mathcal{C}. \tag{12}$$

The key to understanding this ILP is the rounding technique that we employ in our approximation algorithm; so let us describe our algorithm first. We relax the integrality constraint , i.e. constraint (11) and allow the variables p to take any value between 0 and 1, both inclusive. We solve the resulting FLP, and then round the solution to obtain an integral solution. It is in this rounding procedure that the crux of our algorithm lies. We choose assigned capacities according to $p(B,C)$, noting that for a fixed base-station B, $p(B,C)$ is a valid probability distribution. Now, for any base-station B, if the assigned capacity $\tau(B)=\eta(B,C')$, then for each link (B,C), we add a flow of $g_{C'}(B,C)\equiv f_{C'}(B,C)/p(B,C')$ to the $s-B-C-t$ path in the augmentation graph. Crucially, this does not violate the budget constraint at any base-station B since the total outflow at B is $\sum_{C\in\mathcal{C}} g_{C'}(B,C)$, which is at most $\beta(B)$ by constraint (7); neither does it violate the capacity constraint on any link (B,C) since constraint (9) ensures that the flow on link (B,C) is at most $\psi_\tau(B,C)$. Hence, we focus on analyzing violations of the demand constraints. The total inflow at C is $\sum_{B\in\mathcal{B}} g_{C'(B)}(B,C)$. Unfortunately, assigning these flow values simultaneously for all base-stations might lead to an overflow in a demand arc (i.e., a demand constraint violation). For a client with overflow, we decrease the incoming flows arbitrarily until the flow on the link to t exactly matches its capacity (we call this the *truncation step*). Since such a truncated flow is feasible, our ultimate goal is to prove that the truncation step decreases the initial flow only by a constant fraction in expectation.

Let $F(B, C)$ be the random variable denoting the flow on link (B, C); clearly, $F(B, C) = g_{C'}(B, C)$ with probability $p(B, C')$ and its expectation

$$\mathbb{E}[F(B, C)] = f(B, C) \equiv \sum_{C' \in \mathcal{C}} f_{C'}(B, C) = \sum_{C' \in \mathcal{C}} p(B, C') g_{C'}(B, C).$$

Constraint (8) states that the expected inflow $\sum_{B \in \mathcal{B}} f(B, C)$ at client C is at most its demand $\alpha(C)$. Also, for a given C, the $F(B, C)$ values are independent. Finally, constraint (9) enforces that $F(B, C) \le \alpha(C)$ irrespective of the choice of the assigned capacity at base-station B, (i.e. inflow due to a single base-station at a client never exceeds the demand of the client). Thus, we ensure that there is some restriction on the wasted capacity, i.e. the capacity in the base-client links which are left unused due to truncation; such a restriction was absent in the previous formulation and, as we will see, this additional condition will be sufficient for our purpose.

Note. The rounding procedure can be simplified in an actual implementation. Once we obtain the assigned capacities of all the base-stations using randomized rounding as described above, we can run a maximum flow algorithm on the augmentation graph. Note that this achieves at least as much (and potentially more) flow as that achieved by the rounding procedure described above. So an actual implementation of our algorithm will rather employ a maximum flow subroutine than the above procedure for determining flows. However, we assume that our algorithm uses the above procedure since it would be simpler to analyze—all bounds proved using this assumption hold for an actual implementation using maximum flow as well.

Before moving on to the analysis of the algorithm, let us verify that the new ILP does represent the original problem. To do this, let us fix some optimal solution (τ^*, f^*) for the original problem. Consider now a solution to our ILP defined as follows. For each base-station B we set $p(B, C) = 1$ if B chooses $\eta(C)$ as its assigned capacity i.e. if $\tau^*(B) = \eta(B, C)$; otherwise $p(B, C) = 0$. Next, for each link (B, C), we set $f_{C'}(B, C) = f^*(B, C)$ if $\tau^*(B) = \eta(B, C')$, and $f_{C'}(B, C) = 0$ otherwise. Observe that all the constraints are preserved, and the objective value corresponding to this solution has value equal to that for the optimal solution. The converse direction is similar and we omit it for brevity.

2.3 Analysis

If there are n base-stations and m clients, then the algorithm clearly runs in time polynomial in $N = \max(n, m)$. So, we focus on proving guarantees on the approximation factor of the algorithm. By the discussion in the previous section, we know that in our rounding procedure the difference between the objective value of the solution to the FLP and the actual flow that we obtain, consists solely of the amount of initial flow that we have to truncate due to overflows at clients. Thus our main task is to prove upper bounds on the expected overflow. Let $F(C) \equiv \sum_{B \in \mathcal{B}} F(B, C)$ be the random variable denoting total inflow at C before truncation and $T(C) \equiv \min(F(C), \alpha(C))$ be the random variable representing the inflow at C after truncation. We would like to show that

$$\mathbb{E}[T(C)] \geq (1 - 1/e)\mathbb{E}[F(C)]. \tag{13}$$

Then,

$$\mathbb{E}[\sum_{C \in \mathcal{C}} T(C)] \geq (1 - 1/e)\mathbb{E}[\sum_{C \in \mathcal{C}} F(C)] \geq (1 - 1/e)T^*, \tag{14}$$

where T^* is the total flow in an optimal integral solution. This proves Theorem 2, which was stated in Section 1.

To establish inequality (13), we will need the following theorem (a similar proof appears in [1]).

Theorem 3. *Suppose we have a sequence of independent discrete random variables $X_1, X_2, \ldots, X_n$ such that each $0 \leq X_i \leq 1$. Furthermore, suppose $X = \sum_{i=1}^n X_i$ and $\mathbb{E}[X] \leq 1$. If $Y = \min(X, 1)$, then*

$$\mathbb{E}[Y] \geq (1 - 1/e)\mathbb{E}[X].$$

We first use this theorem to prove inequality (13), and then give a proof of the theorem itself. If, for client C, we define $X_i = F(B_i, C)/\alpha(C)$ (where $\mathcal{B} = \{B_1, \ldots, B_n\}$) and $Y = T(C)/\alpha(C)$, then such X_is and Y satisfy the assumptions of the theorem. Thus, we can conclude that

$$\mathbb{E}[T(C)] = \alpha(C)\mathbb{E}[Y] \geq (1 - 1/e)\alpha(C)\mathbb{E}[X] = (1 - 1/e)\mathbb{E}[F(C)].$$

Proof (Theorem 3). Our proof has the following outline. We assume for the sake of contradiction that there exists a sequence $\{\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n\}$ of discrete random variables such that

$$\mathbb{E}[\hat{Y}] = \mathbb{E}[\min(\sum_i \hat{X}_i, 1)] < (1 - 1/e)\mathbb{E}[\hat{X}] = (1 - 1/e)\mathbb{E}[\sum_i \hat{X}_i].$$

We call such a sequence $(\hat{X}_i)$ a *nemesis sequence*. First, we prove that we can assume without loss of generality, that $\hat{X}_i$s are 0-1 random variables. Then, we prove our theorem for 0-1 random variables, thus arriving at a contradiction for the general case.

Let $S(\hat{X}_i)$ be the number of distinct values other than 0 and 1 for which $\hat{X}_i$ has non-zero probability. Now, let us consider a nemesis sequence $(\hat{X}_i)$ that minimizes $\sum_i S(\hat{X}_i)$. We will prove that if $\sum_i S(\hat{X}_i) > 0$ then there exists another nemesis sequence $(\widetilde{X}_i)$ with $\sum_i S(\widetilde{X}_i) < \sum_i S(\hat{X}_i)$. The minimality of $(\hat{X}_i)$ implies there exists a nemesis sequence with $\sum_i S(\hat{X}_i) = 0$, i.e. $(\hat{X}_i)$ is a sequence of 0-1 variables.

If $\sum_i S(\hat{X}_i) > 0$, then there exists some k such that $S(\hat{X}_k) > 0$, which in turn means that there exists some $0 < a < 1$ such that $Pr[\hat{X}_k = a] = p > 0$. Suppose that this $\hat{X}_k$ takes value of 0 and 1 with probability $q \geq 0$ and $r \geq 0$ respectively (note that p, q and r do not necessarily sum to 1). Now, consider another random variable $\widetilde{X}_k$ that is distributed identically to $\hat{X}_k$ except that the probabilities of a, 0 and 1 are changed to 0, $q+(1-a)p$ and $r+ap$ respectively. Note that $\mathbb{E}[\widetilde{X}_k]=\mathbb{E}[\hat{X}_k]$, $0 \leq \widetilde{X}_k \leq 1$ and $S(\widetilde{X}_k)=S(\hat{X}_k)-1$. So, if we define $\widetilde{X}_i=\hat{X}_i$ for $i \neq k$, then $\mathbb{E}[\hat{X}]=\mathbb{E}[\widetilde{X}] \leq 1$. We would like to compare $\mathbb{E}[\hat{Y}]$ to $\mathbb{E}[\widetilde{Y}] \equiv \mathbb{E}[\min\{\widetilde{X},1\}]$. Note that by our definition, for any $\delta \geq 0$,

$$Pr[\widetilde{X}-\widetilde{X}_k=\delta]=Pr[\hat{X}-\hat{X}_k=\delta].$$

Thus, to prove that $\mathbb{E}[\widetilde{Y}] \leq \mathbb{E}[\hat{Y}]$, it is sufficient to prove that

$$\mathbb{E}[\widetilde{Y}|\widetilde{X}-\widetilde{X}_k=\delta] \leq \mathbb{E}[\hat{Y}|\hat{X}-\hat{X}_k=\delta],$$

for all $\delta \geq 0$.

Clearly, if $\delta \geq 1$ then

$$\mathbb{E}[\widetilde{Y}|\widetilde{X}-\widetilde{X}_k=\delta]=1=\mathbb{E}[\hat{Y}|\hat{X}-\hat{X}_k=\delta];$$

so the inequality holds. On the other hand, for $\delta < 1$,

$$\begin{aligned}\mathbb{E}[\widetilde{Y}|\widetilde{X}-\widetilde{X}_k=\delta]-\mathbb{E}[\hat{Y}|\hat{X}-\hat{X}_k=\delta] &= \mathbb{E}[\min\{\widetilde{X}_k,1-\delta\}]-\mathbb{E}[\min\{\hat{X}_k,1-\delta\}]\\ &= ap(1-\delta)-p\min\{a,1-\delta\} \leq 0.\end{aligned}$$

Thus, $\mathbb{E}[\widetilde{Y}] \leq \mathbb{E}[\hat{Y}]$, which proves that $\{\hat{X}_i\}$ had to be a zero-one nemesis sequence.

Now, when $\{\hat{X}_i\}$ is zero-one,

$$\begin{aligned}\mathbb{E}[\hat{Y}] &= Pr[\hat{X} \geq 1]\\ &= 1-\prod_i(1-Pr[\hat{X}_i=1])\\ &\geq 1-(1-\sum_i \mathbb{E}[\hat{X}_i]/n)^n\\ &\geq 1-e^{-\mathbb{E}[\hat{X}]}\\ &\geq (1-1/e)\mathbb{E}[\hat{X}],\end{aligned}$$

as desired, where in the first inequality we used the fact that

$$\sum_i Pr[\hat{X}_i=1]=\sum_i \mathbb{E}[\hat{X}_i]=\mathbb{E}[\hat{X}],$$

and the arithmetic/geometric mean inequality.

To conclude, we may note that this theorem is tight for n i.i.d. 0-1 random variables X_i with $Pr[X_i=1]=1/n$.

3 Conclusion

The ability to adaptively change channel widths in wireless networks introduces interesting algorithmic problems. In this paper, we have studied a throughput maximization problem in infrastructure wireless networks that was identical to the maximum flow problem in bipartite graphs with adaptive channel width. An interesting open question is to to find maximum flow in a *general* network with adaptive channels.

Acknowledgements

We thank the anonymous referees for their helpful comments.

References

1. Andelman, N., Mansour, Y.: Auctions with budget constraints. In: 9th Scandinavian Workshop on Algorithm Theory, pp. 26–38 (2004)
2. Azar, Y., Regev, O.: Combinatorial algorithms for the unsplittable flow problem. Algorithmica 44(1), 49–66 (2006)
3. Chakrabarti, A., Chekuri, C., Gupta, A., Kumar, A.: Approximation algorithms for the unsplittable flow problem. Algorithmica 47(1), 53–78 (2007)
4. Chandra, R., Mahajan, R., Moscibroda, T., Raghavendra, R., Bahl, P.: A case for adapting channel width in wireless networks. In: SIGCOMM, pp. 135–146 (2008)
5. Chekuri, C., Khanna, S., Shepherd, F.B.: An O(sqrt(n)) approximation and integrality gap for disjoint paths and unsplittable flow. Theory of Computing 2(1), 137–146 (2006)
6. Dinic, E.A.: Algorithm for solution of a problem of maximum flow in a network with power estimation. Soviet Math. Doklady (Doklady) 11, 1277–1280 (1970)
7. Edmonds, J., Karp, R.M.: Theoretical improvements in algorithmic efficiency for network flow problems. Journal of the ACM 19(2), 248–264 (1972)
8. Even, S., Tarjan, R.E.: Network flow and testing graph connectivity. SIAM J. Comput. 4(4), 507–518 (1975)
9. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Canadian Journal of Mathematics 8, 399–404 (1956)
10. Goldberg, A.V., Rao, S.: Beyond the flow decomposition barrier. J. ACM 45(5), 783–797 (1998)
11. Goldberg, A.V., Tarjan, R.E.: A new approach to the maximum-flow problem. J. ACM 35(4), 921–940 (1988)
12. Guruswami, V., Khanna, S., Rajaraman, R., Shepherd, F.B., Yannakakis, M.: Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. J. Comput. Syst. Sci. 67(3), 473–496 (2003)
13. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM J. Comput. 2(4), 225–231 (1973)
14. Karger, D.R.: Using random sampling to find maximum flows in uncapacitated undirected graphs. In: STOC, pp. 240–249 (1997)
15. Karger, D.R.: Better random sampling algorithms for flows in undirected graphs. In: SODA, pp. 490–499 (1998)
16. Karger, D.R., Levine, M.S.: Finding maximum flows in undirected graphs seems easier than bipartite matching. In: STOC, pp. 69–78 (1998)
17. Karger, D.R., Levine, M.S.: Random sampling in residual graphs. In: STOC, pp. 63–66 (2002)
18. Kleinberg, J.M.: Single-source unsplittable flow. In: FOCS, pp. 68–77 (1996)
19. Moscibroda, T., Chandra, R., Wu, Y., Sengupta, S., Bahl, P., Yuan, Y.: Load-Aware Spectrum Distribution in Wireless LANs. In: ICNP (2008)
20. Shannon, C.E.: Communication in the presence of noise. Proc. Institute of Radio Engineers 37(1), 10–21 (1949)
21. Srinivasan, A.: Improved approximations for edge-disjoint paths, unsplittable flow, and related routing problems. In: FOCS, pp. 416–425 (1997)

Mediated Population Protocols*

Ioannis Chatzigiannakis[1,2], Othon Michail[2], and Paul G. Spirakis[1,2]

[1] Research Academic Computer Technology Institute (CTI), P.O. Box 1382, N. Kazantzaki Str., 26500 Patras, Greece
[2] Computer Engineering and Informatics Department (CEID), University of Patras, 26500, Patras, Greece
{ichatz,spirakis}@cti.gr, michailo@ceid.upatras.gr

Abstract. We extend here the Population Protocol model of Angluin et al. [2] in order to model more powerful networks of very small resource-limited artefacts (agents) that are possibly mobile. The main feature of our extended model is to allow edges of the communication graph, G, to have *states* that belong to a constant size set. We also allow edges to have *readable only costs*, whose values also belong to a constant size set. Our protocol specifications are still independent of the population size and do not use agent ids, i.e. they preserve *uniformity* and *anonymity*. Our Mediated Population Protocols (MPP) can stably compute graph properties of the communication graph. We show this for the properties of maximal matchings (in undirected communication graphs), also for finding the transitive closure of directed graphs and for finding all edges of small cost. We demonstrate that our mediated protocols are stronger than the classical population protocols, by presenting a MPP for a non-semilinear predicate. To show this fact, we state and prove a general theorem about the composition of two stably computing mediated population protocols. We also show that all predicates stably computable in our model are (non-uniformly) in the class $NSPACE(|E(G)|)$.

1 Introduction

1.1 Population Protocols

In a seminal work, Angluin et al. in [2] proposed the *population protocol model* in order to represent sensor networks consisting of extremely limited agents that may move and interact in pairs. Due to their severe limitations, the agents can be represented as a population V of $|V| = n$ finite state machines. A common assumption is that a global start signal initiates computation by informing the agents to sense their environment in order to receive a piece of the input. Two agents communicate when they come sufficiently close to each other. The movement is carried out by an *adversary scheduler*. A strong global *fairness condition* is imposed on the adversary to ensure the protocol makes progress.

* This work has been partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (FRONTS).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 363–374, 2009.

Formally, a population protocol $\mathcal{A}$ consists of finite *input and output alphabets* X and Y, a finite set of *states* Q, an *input function* $I : X \to Q$ mapping inputs to states, an *output function* $O : Q \to Y$ mapping states to outputs, and a *transition function* $\delta : Q \times Q \to Q \times Q$. The critical assumption that diversifies the population protocol model from traditional distributed systems is that the protocol specifications are independent of the population size (that is, need $\mathcal{O}(1)$ total memory capacity in each agent), which is known as the *uniformity* property of population protocols. Moreover, population protocols are *anonymous* since there is no room in the state of an agent to store a unique identifier.

A network communication graph $G = (V, E)$ (see [1]) describes the permissible ordered pairwise interactions. G is assumed to be a directed graph with no multi-edges or self-loops (simple) and its n nodes are numbered 1 through n. An edge $(u, \upsilon) \in E$ indicates the possibility of a communication between u and υ, in which u is the *initiator* and υ the *responder*. The *basic* population protocol model assumes the all-pairs family of directed communication graphs, denoted $\mathcal{G}^d_{All}$, which simply contains for each n the complete directed graph on n vertices.

A *population configuration* is a mapping $C : V \to Q$ providing a snapshot of the population states. $C \to C'$ denotes that configuration C can go to C' through a single interaction, i.e. there is an edge $e = (u, \upsilon) \in E$ such that $\delta(C(u), C(\upsilon)) = (C'(u), C'(\upsilon))$ and $C'(\omega) = C(\omega)$ for all $\omega \in V - \{u, \upsilon\}$. In this case we say that C goes to C' via encounter $e = (u, \upsilon)$, and to emphasize that, we write $C \xrightarrow{e} C'$. C' is said to be *reachable* from C, denoted $C \xrightarrow{*} C'$, if C can be converted to C' in one or more steps.

An *execution* is a finite or infinite sequence of configurations $C_0, C_1, C_2, \ldots$, where C_0 is an initial configuration and $C_i \to C_{i+1}$, for all $i \geq 0$. At any point during the execution of a population protocol, each agent's state determines its output at that time (the output of u under configuration C is $O(C(u))$, for each $u \in V$). An infinite execution is fair if for every possible transition $C \to C'$, if C occurs infinitely often in the execution, then C' occurs infinitely often. A *computation* is an infinite fair execution.

Generally, population protocols do not halt. Instead, halting is replaced by an interesting property called *stability*. Stability was defined in [2] to be a situation where computation reaches a configuration C, after which, no matter how the computation proceeds, no agent will be able to change its output value. Such a configuration C is called an *output-stable* configuration. A protocol $\mathcal{A}$ (stably) *computes* a function f that maps multisets of elements of X to Y if, for every such multiset x and every computation that starts from the initial configuration corresponding to x, the output value of every agent eventually stabilizes to $f(x)$.

For the basic population protocol model there exists an exact characterization of the computable predicates: they are precisely the *semilinear predicates* or equivalently the predicates definable by first-order logical formulas in *Presburger arithmetic* [2,4,5]. In the *stabilizing inputs* variant of the basic model [1], convergence to a stable common output value is only required after the inputs stop changing (here, each agent has an input field that can change over time).

The results of [4] show that again the semilinear predicates are all that can be computed by this model.

1.2 Other Previous Work

The motivation given for the population protocol model was the study of sensor networks in which passive agents were carried along by other entities. Much work has been devoted to the, now, well known fact that the set of computable predicates of the basic (complete interaction graph) population protocol model and most of its variants is exactly equal or closely related to the set of *semilinear predicates*. Also, in [2], the *probabilistic population protocol* model was proposed, in which the scheduler selects randomly and uniformly the next pair to interact. More recent work has concentrated on performance, supported by this random scheduling assumption. [6] and [9] considered a huge population hypothesis (population going to infinity), and studied the dynamics, stability and computational power of probabilistic population protocols by exploiting the tools of continuous nonlinear dynamics. In [6] it was also proven that there is a strong relation between classical finite population protocols and models given by ordinary differential equations. Distributed computing with advice (in the spirit of our model) was considered in [10]. Moreover, several extensions of the basic model have been proposed to more accurately reflect the requirements of practical systems. In [1] they studied what properties of restricted communication graphs are stably computable, gave protocols for some of them, and proposed the model extension with *stabilizing inputs*. Finally, some works incorporated agent failures and gave to the agents slightly increased memory capacity. For an excellent introduction to the subject see [5].

1.3 Our Approach

We extend the population protocol model with communication links satisfying the following properties: Each $e \in E$ is equipped with a buffer of $\mathcal{O}(1)$ total storage capacity. Before an interaction $e = (u, v)$, the interacting pair reads the contents of the corresponding buffer, that is, the state of e, to provide it to the transition function. After an interaction $e = (u, v)$, the interacting pair updates the contents of the corresponding buffer, according to the new state of e, returned by the transition function. Since the memory of the edges is constant and independent of the population size, the protocol specifications are independent of n, i.e. *both uniformity and anonymity are preserved.*

By letting the edges keeping states, the first gain is that we manage to get more computational power in comparison to the basic population protocol model. The additional computational power is limited (as expected) due to the fact that this additional set of states is again of constant cardinality (and usually we try to keep it low, i.e. one or two bits suffice). We prove that the derived model is able to compute at least one non-semilinear predicate.

The new model also gives rise to many other novel computational possibilities. By defining a natural relaxation of stability, that we call *r-stability*, we obtain protocols that are able to locate subgraphs of the communication graph. By associating each edge with a read-only cost, under certain assumptions, we are able to devise protocols that solve optimization problems concerning the communication graph. This cost can be assumed to be stored (together with the state) in the constant size buffer of the edge.

2 The New Model

2.1 Mediated Population Protocols

A *mediated population protocol* $\mathcal{A}$ consists of finite *input and output alphabets* X and Y, a finite set of *agent states* Q, an *agent input function* $I : X \rightarrow Q$ mapping inputs to agent states, an *agent output function* $O : Q \rightarrow Y$ mapping agent states to outputs, a finite set of *edge states* S, an *edge input function* $\iota : X \rightarrow S$ mapping inputs to edge states, an *edge output function* $\omega : S \rightarrow Y$ mapping edge states to outputs, an *output instruction* r, a finite totally ordered *cost set* K, a *cost function* $c : E \rightarrow K$ assigning a cost to each edge of the communication graph and a *transition function* $\delta : Q \times Q \times K \times S \rightarrow Q \times Q \times K \times S$ (from now on we will always assume that the cost remains the same after applying δ and so we will omit specifying an output cost). If $\delta(q_i, q_j, x, s) = (q'_i, q'_j, s')$ (which, according to our assumption, is equivalent to $\delta(q_i, q_j, x, s) = (q'_i, q'_j, x, s')$), we call $(q_i, q_j, x, s) \rightarrow (q'_i, q'_j, s')$ a *transition*, and we define $\delta_1(q_i, q_j, x, s) = q'_i$, $\delta_2(q_i, q_j, x, s) = q'_j$ and $\delta_3(q_i, q_j, x, s) = s'$. We will call δ_1 the *initiator's acquisition*, δ_2 the *responder's acquisition*, and δ_3 the *edge acquisition* after the corresponding interaction.

In most cases we will assume that $K \subset \mathbb{Z}^+$ and that $c_{max} = \max_{w \in K} \{w\} = \mathcal{O}(1)$. Generally, if $c_{max} = \max_{w \in K} \{|w|\} = \mathcal{O}(1)$ then any agent is capable of storing at most k cumulative costs (at most the value kc_{max}), for some $k = \mathcal{O}(1)$, and we say that the cost function is *useful* (note that a cost range that depends on the population size could make the agents incapable for even a single cost storage and any kind of optimization would be impossible).

A mediated population protocol runs in a communication graph $G = (V, E)$, where V is a population of n agents, and E is an irreflexive binary relation on V, of cardinality denoted by m. In the case of an undirected graph we only require that E is also symmetric, and that for all $(u, v), (v, u) \in E$, (u, v) and (v, u) share the same buffer (in the undirected case we also assume in this work that if $\delta(q_i, q_j, x, s) = (q'_i, q'_j, s')$, then $\delta(q_j, q_i, x, s) = (q'_j, q'_i, s')$, for all $(q_i, q_j, x, s) \in Q \times Q \times K \times S$). In both cases (directed and undirected), a $(u, v) \in E$ means that interaction (u, v) is permitted in which u is the *initiator* and v the *responder*.

A *network configuration* is a mapping $C : V \cup E \rightarrow Q \cup S$ specifying the agent state of each agent in the population and the edge state of each edge in the communication graph. If we restrict our attention on the states of the agents only, we can use the mapping $C_V : V \rightarrow Q$ which is called the *population configuration* and similarly $C_E : E \rightarrow S$ is the *edge configuration*. Let C and C' be network configurations, and let u, v be distinct agents. We say that C goes to C' via encounter $e = (u, v)$, denoted $C \xrightarrow{e} C'$, if $C'(u) = \delta_1(C(u), C(v), x, C(e))$, $C'(v) = \delta_2(C(u), C(v), x, C(e))$, $C'(e) = \delta_3(C(u), C(v), x, C(e))$, and

$C'(z) = C(z)$, for all $z \in (V - \{u, v\}) \cup (E - e)$. We say that C can go to C' in one step, if $C \xrightarrow{e} C'$ for some encounter $e \in E$. We write $C \xrightarrow{*} C'$ if there is a sequence of configurations $C = C_0, C_1, \ldots, C_t = C'$, such that $C_i \to C_{i+1}$ for all i, $0 \leq i < t$, in which case we say that C' is *reachable* from C. The definitions of *execution* and *computation* are the same as in the population protocol model but concern network configurations. Note that the mediated population protocol *code* is of *constant size* and, thus, can be stored in each agent (device) of the population.

2.2 Output Interpretation

Following the definition of stability proposed in [2], we say that a network configuration C is *agent output-stable*, if $O(C'(v)) = O(C(v))$ for all C' where $C \xrightarrow{*} C'$ and for all $v \in V$ (we can also write $O(C'_V) = O(C_V)$ if we extend O to a mapping from population configurations to output assignments like in [2]). Moreover, a configuration C is said to be *edge output-stable*, if $\omega(C'(e)) = \omega(C(e))$ for all C' where $C \xrightarrow{*} C'$ and for all $e \in E$ (i.e. if $\omega(C'_E) = \omega(C_E)$) and *globally output-stable*, if it is both edge output-stable and agent output-stable.

The instruction r that we have included in the model definition is simply an instruction that tells the output viewer how to interpret the output of a protocol. For example, if a protocol is supposed to compute a predicate by reaching an agent output-stable configuration where all agents agree on the correct output value, then instruction r would be: "*Get any $u \in V$ and view its output*". In this case, an agent output-stable configuration is said to be an r-stable configuration.

A configuration C is *r-stable* if one of the following holds: If the problem concerns a subgraph to be found, then C should fix a subgraph that will not change in any C' reachable from C. If the problem concerns a function to be computed by the agents, then an r-stable configuration drops down to an agent output-stable configuration.

We will say that a protocol $\mathcal{A}$ *stably solves* a problem Π, if for every instance I of Π and every computation of $\mathcal{A}$ on I, the network reaches an r-stable configuration C that gives the correct solution for I if interpreted according to the output instruction r. If instead of a problem Π we have a function f to be computed, we will say that $\mathcal{A}$ *stably computes* f. In the special case where Π is an optimization problem, a protocol that stably solves Π will be called an *optimizing population protocol* for problem Π.

3 Some Graph Protocols

3.1 Maximal Matching

We first give a mediated population protocol $MaximalMatching$ that stably solves the following problem:

Problem 1. (*Maximal matching*) Given an undirected communication graph $G = (V, E)$, find a maximal matching, i.e., a set $E' \subseteq E$ such that no two members of

E' share a common end point in V and, moreover, there is no $e \in E - E'$ such that e shares no common end point with every member of E'.

MaximalMatching

- $X = \{0\}$, $Y = \{0, 1\}$,
- $Q = \{q_0, q_1\}$, $S = \{0, 1\}$,
- $I(0) = q_0$,
- $\iota(0) = 0$, $\omega(0) = 0$, $\omega(1) = 1$,
- r: "*Get each* $e \in E$ *for which* $\omega(s_e) = 1$ *(where* s_e *is the state of* e*)*",
- δ: $(q_0, q_0, 0) \rightarrow (q_1, q_1, 1)$

Note that we have omitted specifying costs, since there is no need for them here, and δ is of the simplified form $\delta : Q \times Q \times S \rightarrow Q \times Q \times S$. Moreover, any other possibly interacting pair not appearing in δ, e.g. $(q_0, q_1, 0)$ and $(q_1, q_1, 1)$, is assumed throughout this work to be included as an identity rule, that is, a rule that leaves all interacting components unaffected (e.g. $(q_0, q_1, 0) \rightarrow (q_0, q_1, 0)$ and $(q_1, q_1, 1) \rightarrow (q_1, q_1, 1)$). We, also, don't specify an agent output function because the protocol's correctness concerns only the edge output function.

Theorem 1. *MaximalMatching stably solves the maximal matching problem.*

Proof. Let M be the set of edges in state 1. M is theoretically updated after each interaction, i.e., at any point, any edge in state 1 belongs to M. For an edge $e = (u, v)$ to become a member of M, both its endpoints during the interaction must be in state q_0. If this holds, the edge gets in M and u, v go to state q_1 to indicate that they both have an edge incident to them that belongs to M. From now on, no edges adjacent to e can get in M, simply because their end point on which they coincide with e is in state q_1. This, together with the fact that two interactions happening in parallel cannot concern adjacent edges, proves that M is always a matching. Moreover, an edge not conflicting with M will eventually get in M (if no conflict arises in the meanwhile), since it will be in state 0 and both its end points will be in q_0. The latter proves that M is maximal. □

3.2 Transitive Closure with the Help of a Leader

Assume that $G = (V, E)$ is a graph from the $\mathcal{G}^d_{All}$ family, that is, the all-pairs family of directed communication graphs, and that a protocol has computed a subgraph of G, $G' = (V', E')$, by letting the selected edges (edges in E') be in state 1, while all the remaining edges (i.e. all $e \in E - E'$) are in state 0. Note that V' simply contains all nodes that are incident to at least one $e \in E'$. We want to solve the following problem:

Problem 2. (**Transitive Closure**) Given a communication graph $G = (V, E)$ in $\mathcal{G}^d_{All}$ with a subgraph $G' = (V', E')$ precomputed in the above manner, find the transitive closure of G', that is, find a new edge set E^* that will contain a directed edge (u, v) joining any nodes u, v for which there is a non-null path from u to v in G' (note that always $E' \subseteq E^*$).

We assume a *controlled input assignment* $W : E \to X$ that allows us to give input 1 to any edge belonging to E' and input 0 to any other edge. Moreover, we assume that initially all agents are in state q_0, except for an elected leader (by any leader election protocol) that is in state l. The assumption of a leader and the remark that this helps the protocols was first used in [2] and extensively studied in [3]. We devise a protocol, *TranClos*, with the following specification:

TranClos

- $X = Y = \{0, 1\}$,
- $Q = \{l, q_0, q_1, q_1', q_2, q_2', q_3\}$, $S = \{0, 1\}$,
- *controlled input assignment*: "$W(e') = 1$*, for all* $e' \in E'$*, and* $W(e) = 0$*, for all* $e \in E - E'$",
- $\iota(x) = x$, for all $x \in X$, $\omega(s) = s$, for all $s \in S$,
- r: "*Get each* $e \in E$ *for which* $\omega(s_e) = 1$ *(where* s_e *is the state of* e*)*",
- δ:

$$\begin{array}{rlrl}
(l, q_0, 0) \to (q_0, l, 0) & & (q_2, q_0, 1) \to (q_2', q_3, 1) & \\
(l, q_0, 1) \to (q_1, q_2, 1) & & (q_1, q_3, x) \to (q_1', q_0, 1), & \text{for } x \in \{0, 1\} \\
(q_1, q_2, 1) \to (q_0, l, 1) & & (q_1', q_2', 1) \to (q_0, l, 1) &
\end{array}$$

Theorem 2. *Protocol TranClos stably solves the transitive closure problem.*

For the proof see [8].

3.3 Edges of Minimum Cost

Let us illustrate the incorporation of edge costs in the case of optimization problems, by a simple optimizing population protocol for the following problem:

Problem 3. (*Edges of minimum cost*) Given an undirected connected communication graph $G = (V, E)$ and a useful cost function $c : E \to K$ on the set of edges, where $K \subset \mathbb{Z}^+$, design a protocol that finds the minimum cost edges of E.

MinEdges

- $X = Y = \{0, 1\}$,
- $Q = K \cup \{q_0\}$, $S = \{0, 1\}$,
- $I(x) = q_0$, for all $x \in X$,
- $\iota(x) = 0$, for all $x \in X$, $\omega(s) = s$, for all $s \in S$,
- r: "*Get each* $e \in E$ *for which* $\omega(s_e) = 1$ *(where* s_e *is the state of* e*)*",
- δ:

$$\begin{aligned}
(q_0, q_0, c, d) &\to (c, c, 1) \\
(c_i, c_j, c, d) &\to (c, c, 1), \text{ if } c \leq \min\{c_i, c_j\} \\
&\to (\min\{c_i, c_j\}, \min\{c_i, c_j\}, 0), \text{ if } c > \min\{c_i, c_j\} \\
(c_i, q_0, c, d) &\to (c, c, 1), \text{ if } c \leq c_i \\
&\to (c_i, c_i, 0), \text{ if } c > c_i
\end{aligned}$$

Theorem 3. *MinEdges is an optimizing population protocol for Problem 3.*

Proof. We need to show that the system reaches an r-stable configuration C, where if E_{out} is the subset of E specified by instruction r, we have $e \in E_{out}$ if and only if $c(e) = c_{opt}$, where $c_{opt} = \min_{e \in E}\{c(e)\}$.

All rules of δ together with the fairness assumption ensure that every agent will eventually get c_{opt} (a less cost encountered always replaces the current cost of an agent). At that point the system will be in an agent output-stable configuration, since there is no cost less than c_{opt} in order to replace it. From now on, after every interaction (u, υ), where $e = \{u, \upsilon\}$, $E_{out} \leftarrow E_{out} \cup \{e\}$, if no interacting agent's cost is less than $c(e)$ (that is, if $c(e) = c_{opt}$) and $E_{out} \leftarrow E_{out} - \{e\}$, otherwise.

It follows that, eventually, the system will enter a configuration C where $e \in E_{out}$ will imply that $c(e) = c_{opt}$ and $e \notin E_{out}$ that $c(e) > c_{opt}$. At that time, no edge will be able to enter or leave E_{out}, and since E_{out} is the set specified by r, C will be an r-stable configuration as needed. These together imply that *MinEdges* is indeed an optimizing population protocol for Problem 3. □

4 Computability

4.1 All-Pairs Directed Communication Graphs

We now investigate some aspects of the computational power of the mediated population protocol (MPP) model and show that in the special case of the all-pairs family of directed communication graphs it is in fact stronger than the basic model proposed in [2].

Definition 1. *The MPP model with the additional constraint that it runs on the all-pairs family of directed communication graphs ($\mathcal{G}^d_{All}$), will be called the* basic MPP model*.*

Definition 2. *We say that a predicate is* strongly stably computable *by the MPP model, if it is stably computable in the classical sense of stable computation, that is, all agents eventually agree on the correct output value.*

On the other hand, if we say that a predicate is stably computable by the MPP model (without including the word "strongly"), it is not obvious if all agents agree on the same output value or not. Finally, when we say that a predicate is stably computable by the population protocol model, it always means that all agents eventually agree on the correct output value, since in this case we always follow the classical definition of stable computation, as appears in [2].

Theorem 4. *The population protocol model is a special case of the MPP model.*

Proof. Ignoring the edge functions, the edge states, the edge costs, and the output instruction r in the mediated population protocol model, makes the two models equivalent. □

All the following corollaries are immediate consequences of Theorem 4, since it shows that the population protocol model can be simulated by the mediated population protocol model. There is nothing that the population protocol model does that the MPP model is not capable of doing.

Corollary 1. *Any predicate stably computable by the population protocol model is also strongly stably computable by the mediated population protocol model.*

Corollary 2. *Any predicate stably computable by the basic population protocol model is also strongly stably computable by the basic MPP model.*

Corollary 3. *Any predicate stably computable by the population protocol model with stabilizing inputs is also strongly stably computable by the similar extension of the mediated population protocol model with stabilizing inputs.*

It is well known that Presburger arithmetic does not allow multiplication of variables. Moreover, any semilinear predicate can be described by first-order logical formulas in Presburger arithmetic and it is known that a predicate is computable in the basic population protocol model if and only if it is semilinear. To demonstrate that the basic MPP model is stronger, it suffices to show that there is at least one non-semilinear predicate that is strongly stably computable under this model.

It is obvious that the predicate "the number of c's is the product of the number of a's and the number of b's" is not semilinear. This holds, because multiplication of variables cannot be described by first-order logical formulas in Presburger arithmetic. Let N_q denote the multiplicity of state q in the input configuration multiset. Then, $N_c = N_a \cdot N_b$ is a shorthand of the above predicate and the mediated protocol $VarProduct$ that we will now describe, stably computes it in $\mathcal{G}^d_{All}$.

VarProduct

- $X = \{a, b, c, 0\}$, $Y = \{0, 1\}$,
- $Q = \{a, \dot{a}, b, c, \bar{c}, 0\}$, $S = \{0, 1\}$,
- $I(x) = x$, for all $x \in X$, $O(a) = O(b) = O(\bar{c}) = O(0) = 1$, and $O(c) = O(\dot{a}) = 0$,
- $\iota(x) = 0$, for all $x \in X$,
- r: "*If there is at least one agent with output 0, reject, else accept.*",
- δ: $(a, b, 0) \rightarrow (\dot{a}, b, 1)$, $(c, \dot{a}, 0) \rightarrow (\bar{c}, a, 0)$, $(\dot{a}, c, 0) \rightarrow (a, \bar{c}, 0)$

Theorem 5. *Protocol $VarProduct$ stably computes (according to our relaxed definition of stable computation) predicate $N_c = N_a \cdot N_b$ in $\mathcal{G}^d_{All}$.*

Proof Sketch. Notice that the number of links leading from agents in state a to agents in state b equals $N_a \cdot N_b$. For each a the protocol tries to erase b c's. Each a is able to remember the b's that has already counted (for every such b a c has been erased) by marking the corresponding links. If the c's are less than the product then at least one $\dot{a}$ remains and if the c's are more at least one c

remains. In both cases at least one agent that outputs 0 remains. If $N_c = N_a \cdot N_b$ then every agent eventually outputs 1. For a full proof see [8].

It is easy to see that $VarProduct$'s states in every computation eventually stop changing. Any protocol with the above property will be called a protocol with *stabilizing states*. Keep in mind that original stable computation of population protocols requires that all agents agree on their output value, and that it is correct. But $VarProduct$ does not seem to strongly stably compute $N_c = N_a \cdot N_b$, since the agents do not always agree on their final output value. Now we are about to prove that with a slight addition it does.

Note that instruction r defines a semilinear predicate on multisets of states (members of Q). To see this, simply write r formally as $(N_c > 0) \vee (N_{\dot{a}} > 0)$. The fact that it is semilinear suffices to prove that there is a population protocol $\mathcal{B}'$ with stabilizing inputs from the set Q of $VarProduct$'s states, that stably computes it. This follows from a result in [4], stating that any population protocol for fixed inputs can be adapted to work with stabilizing inputs. Moreover, Corollary 3 implies that there is a mediated protocol $\mathcal{B}$ with stabilizing inputs that is equivalent to $\mathcal{B}'$ (the one that ignores edges and does the same things as $\mathcal{B}'$), that is, it strongly stably computes the predicate defined by r. Finally, $VarProduct$ has stabilizing states, so its composition with $\mathcal{B}$ (their product construction) provides stabilizing inputs to $\mathcal{B}$. If the protocol's answer is now taken from $\mathcal{B}$'s output, then it is trivial to see that their composition strongly stably computes $N_c = N_a \cdot N_b$.

We state now a composition theorem to generalize this remark. Its proof can be found in [8]. In fact, our composition theorem holds for any family of directed and connected communication graphs $\mathcal{G}$.

Theorem 6. *Any MPP $\mathcal{A}$, that stably computes a predicate p with stabilizing states in some family of directed and connected communication graphs $\mathcal{G}$, containing an instruction r that defines a semilinear predicate t on multisets of $\mathcal{A}$'s agent states, can be composed with a provably existing MPP $\mathcal{B}$, that strongly stably computes t with stabilizing inputs in $\mathcal{G}$, to give a new MPP $\mathcal{C}$ satisfying the following properties:*

- *$\mathcal{C}$ is formed by the composition of $\mathcal{A}$ and $\mathcal{B}$,*
- *its input is $\mathcal{A}$'s input,*
- *its output is $\mathcal{B}$'s output, and*
- *$\mathcal{C}$ strongly stably computes p in $\mathcal{G}$.*

Definition 3. *Let* SEM *be the class of predicates stably computable, according to the classical sense of stable computation, by the basic population protocol model (precisely the semilinear predicates), and* MP *the class of number predicates strongly stably computable by the basic mediated population protocol model.*

Theorem 7. *SEM is a proper subset of MP.*

Proof. Corollary 2 implies that $SEM \subseteq MP$. Theorem 5 shows that there is a non-semilinear predicate, $p : N_c = N_a \cdot N_b$, that does not belong to SEM but is stably computable by the basic MPP model (according to our definition of stable computation). The mediated protocol $VarProduct$ that stably computes p, contains an output instruction r that defines a semilinear predicate t on multisets of $VarProduct$'s states. Consequently, Theorem 6 applies and we get that p is also strongly stably computable by the basic MPP model (i.e. stably computable according to the classical sense of stable computation), in other words, p belongs to MP, and the theorem follows. □

4.2 Any Family of Communication Graphs : Non-uniform Upper Bounds on Computability

Definition 4. *Let* UMP *be the class of predicates stably computable by the MPP model in any family $\mathcal{G}$ of undirected, connected communication graphs, and* DMP *the class of predicates stably computable by the MPP model in any family $\mathcal{G}'$ of directed, connected communication graphs.*

Let m denote the number of edges of any communication graph G.

Theorem 8. *All predicates in DMP and UMP are in the class $NSPACE(m)$.*

Proof. Let $\mathcal{A}$ be a mediated protocol that stably computes such a predicate p in some family of graphs $\mathcal{G}$, and let $G \in \mathcal{G}$ be any graph of this family. Since G is always connected, we have that $m \geq n-1$. A network configuration can be represented explicitly, by storing a state per node and a state per edge of G. This takes $\mathcal{O}(m)$ space. Note that w.l.o.g. we can talk about languages instead of predicates. So, $\mathcal{A}$ stably computes the language L corresponding to p (its *support*).

We will now present a nondeterministic Turing machine $M_{\mathcal{A}}$ that decides L in space $\mathcal{O}(m)$. $M_{\mathcal{A}}$ works as follows: To accept input x, $M_{\mathcal{A}}$ must verify two conditions: That there exists a configuration C reachable from $I(x)$ (the initial configuration corresponding to x), in which all relevant states satisfy the output instruction r, and that there is no configuration C' reachable from C, in which r is violated.

The first condition is verified by guessing and checking a sequence of network configurations, starting from $I(x)$ and reaching such a C. $M_{\mathcal{A}}$ guesses a C_{i+1} each time, verifies that $C_i \to C_{i+1}$ (begins from $C_0 = I(x)$, i.e. $i = 0$) and, if so, replaces C_i by C_{i+1}, otherwise drops this C_{i+1}. The second condition is the complement of a similar reachability problem. But $NSPACE$ is closed under complement for all space functions $\geq \log n$ (see [11]). Thus, $M_{\mathcal{A}}$ decides L in $O(m)$ space. □

Note that, as far as a DMP is concerned, even a "standard" (not mediated) population protocol whose G is a directed line can simulate a deterministic linear space Turing machine [2]. Thus, by applying Savitch's theorem [12], one can say informally that DMP is between $NSPACE(\sqrt{n})$ and $NSPACE(m)$. However, for UMP, we only know that $SEM \subset UMP \subseteq NSPACE(m)$.

5 Future Work

We are currently experimenting with Population Protocols (see [7]) and investigating the possible graph properties that can be computed by Mediated Population Protocols.

Acknowledgements. We thank Maria Serna for posing the question of Transitive Closure and its importance.

References

1. Angluin, D., Aspnes, J., Chan, M., Fischer, M.J., Jiang, H., Peralta, R.: Stably computable properties of network graphs. In: Proc. Distributed Computing in Sensor Systems: 1st IEEE International Conference, pp. 63–74 (2005)
2. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M.J., Peralta, R.: Computation in networks of passively mobile finite-state sensors. In: 23rd Annual ACM Sympsium on Principles of Distributed Computing (PODC), pp. 290–299. ACM Press, New York (2004)
3. Angluin, D., Aspnes, J., Eisenstat, D.: Fast computation by population protocols with a leader. Distributed Computing 21(3), 183–199 (2008)
4. Angluin, D., Aspnes, J., Eisenstat, D.: Stably computable predicates are semilinear. In: Proc. 25th Annual ACM Symposium on Principles of Distributed Computing, pp. 292–299 (2006)
5. Aspnes, J., Ruppert, E.: An introduction to population protocols. Bulletin of the European Association for Theoretical Computer Science 93, 98–117 (2007); Mavronicolas, M. (ed.). Columns: Distributed Computing
6. Bournez, O., Chassaing, P., Cohen, J., Gerin, L., Koegler, X.: On the convergence of population protocols when population goes to infinity. To appear in Applied Mathematics and Computation (2009)
7. Chatzigiannakis, I., Michail, O., Spirakis, P.G.: Experimental verification and performance study of extremely large sized population protocols. FRONTS Technical Report FRONTS-TR-2009-3 (January 2009), http://fronts.cti.gr/aigaion/?TR=61
8. Chatzigiannakis, I., Michail, O., Spirakis, P.G.: Mediated Population Protocols. FRONTS Technical Report FRONTS-TR-2009-8 (February 2009), http://fronts.cti.gr/aigaion/?TR=65
9. Chatzigiannakis, I., Spirakis, P.G.: The dynamics of probabilistic population protocols. In: Taubenfeld, G. (ed.) DISC 2008. LNCS, vol. 5218, pp. 498–499. Springer, Heidelberg (2008)
10. Fraigniaud, P., Gavoille, C., Ilcinkas, D., Pelc, A.: Distributed computing with advice: Information sensitivity of graph coloring. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 231–242. Springer, Heidelberg (2007)
11. Immerman, N.: Nondeterministic space is closed under complementation. SIAM J. Comput. 17(5), 935–938 (1988); (see also page 153 C. H. Papadimitriou "Computational Complexity")
12. Savitch, W.J.: Relationship between nondeterministic and deterministic tape classes. J. CSS 4, 177–192 (1970) (see also page 149-150 C. H. Papadimitriou "Computational Complexity")

Rumor Spreading in Social Networks*

Flavio Chierichetti, Silvio Lattanzi, and Alessandro Panconesi

Dipartimento di Informatica
Sapienza Università di Roma

Abstract. Social networks are an interesting class of graphs likely to become of increasing importance in the future, not only theoretically, but also for its probable applications to ad hoc and mobile networking. Rumor spreading is one of the basic mechanisms for information dissemination in networks, its relevance stemming from its simplicity of implementation and effectiveness. In this paper, we study the performance of rumor spreading in the classic preferential attachment model of Bollobás et al. which is considered to be a valuable model for social networks. We prove that, in these networks: (a) The standard `PUSH-PULL` strategy delivers the message to all nodes within $O(\log^2 n)$ rounds with high probability; (b) by themselves, `PUSH` and `PULL` require polynomially many rounds. (These results are under the assumption that m, the number of new links added with each new node is at least 2. If $m = 1$ the graph is disconnected with high probability, so no rumor spreading strategy can work.) Our analysis is based on a careful study of some new properties of preferential attachment graphs which could be of independent interest.

1 Introduction

Rumor spreading is one of the basic mechanisms for information dissemination in networks. In this paper we analyze the performance of rumor spreading in the Preferential Attachment model [7]. We show that, while neither `PUSH` nor `PULL` by themselves guarantee fast information dissemination, with `PUSH-PULL` the information reaches all nodes in the network within $O(\log^2 n)$ rounds with high probability, n being the number of nodes in the network.

The study of information dissemination in social networks is an important endeavour, encompassing a variety of questions ranging from the purely technological to the spread of viruses and the diffusion of ideas in human communities. In order to gain insight into these and other related questions, a lot of activity has been devoted to studying stochastic generative models for social networks. The well-known preferential attachment model, defined precisely in the next section, is considered to be able to capture many of their salient features. Thus, it seems worthwhile to study how fundamental primitives of information dissemination behave in such a model. Rumor spreading is one of the very basic such primitives. Its simple, basic character makes it useful as a protocol and interesting theoretically, for one can hope to gain insight on more complex

* This research was partially supported by the ENEA project Cresco, by the Italian Ministry of University and Research under Cofin 2006 grant Web Ram and by the Italian-Israelian FIRB project (RBIN047MH9-000).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 375–386, 2009.

phenomena by studying it. Perhaps surprisingly there seem to be no precise analysis in the literature of the speed with which rumor spreading terminates in the preferential attachment model (to the best of our knowledge) and thus this is the aim of this paper. Before discussing the relevant state-of-the-art, let us first describe our results precisely.

There is a danger of terminological confusion surrounding the term gossip that we shall now try to avoid. In this paper we study the *randomized gossip protocol*, also referred to in the literature with the terms rumor spreading or randomized broadcast (see for instance [11,19]). It should not be confused with the *gossip problem*, in which each node is to broadcast a piece of information and one seeks the most effective protocols to do it (see for instance [20]). The randomized gossip protocol is a widely used protocols in ad hoc networks to implement a broadcast service, and is defined as follows. Its aim is to broadcast a message, i.e. to deliver to every node in the network a message originating from one source node. The randomized gossip protocol proceeds in a sequence of synchronous rounds. At round $t \geq 0$, every node that knows the message, selects a random neighbour and sends message. This is the so-called PUSH strategy. The PULL variant is specular. At round $t \geq 0$ every node that does not yet have the message selects a neighbour uniformly at random and asks for the information, which is transferred provided that it is in possession of the queried neighbour. Finally, the PUSH-PULL strategy is a combination of both. At round $t \geq 0$, each node selects a random neighbour to perform a PUSH if it has the information or a PULL in the opposite case. One of the most studied question concerning rumor spreading is the following: how many rounds will it take for one of the above strategies to disseminate the information to all nodes in the graph, assuming a worst-case source?

We study this question for the preferential attachment model and show the following:

- regardless of the starting node, the PUSH strategy requires, with $\Omega(1)$ probability, polynomially many rounds;
- there are starting nodes such that the PULL strategy requires, with $\Omega(1)$ probability, polynomially many rounds;
- regardless of the starting node, the PUSH-PULL strategy requires, with probability $1 - o(1)$, $O(\log^2 n)$ many rounds.

From the technical point of view, Preferential Attachment networks (henceforth PA network) are quite interesting because in them coexist subgraphs that are very hard for rumor spreading, such as many high-degree stars, the so-called hubs, with subgraphs that are very good, such as small-degree expanders. These two act as opposing forces and only a careful analysis can ascertain which one will prevail. To this aim, we establish several strong structural properties of PA graphs that we believe will be useful for further study. In particular we show that a linear size portion of the graph behaves like a low-degree expander. This expander is not a subgraph made of nodes and edges. Rather it is a sort of cluster graph obtained by collapsing connected components into macronodes that are connected by short paths (as opposed to single edges). This core acts as a sort of fast information exchanger– it can be easily reached by every node and it can itself reach every node.

We now turn to a discussion of the relevant literature.

2 Related Work

The literature on the gossip protocol and social networks is huge and we confine ourselves to what appears to be more relevant to the present work.

Clearly, at least diameter-many rounds are needed for the gossip protocol to reach all nodes. It has been shown that $O(n \log n)$ rounds are always sufficient for a connected graph of n nodes [11]. The problem has been studied on a number of graph classes, such as hypercubes, bounded-degree graphs, cliques and Erdös-Rényi random graphs (see [15,11,18]).

It is a common assumption that graphs generated by the model intuitively introduced by Barabási and Albert is a good representation of social networks [1]. In this model nodes arrive one after the other. Roughly speaking, when a new node arrives, m nodes are chosen randomly as neighbours, with probability proportional to their degree. Bollobás et al. formalize this model in [7]. We will use the terms "preferential attachment" and refer to it as the *PA model*. Analogously, we will speak of *PA graphs* and *PA networks*. The PA model has been the object of a great deal of rigorous study by a number of authors [2,4,5,6,7,8,14]. For instance, it is known that (a) its degree distribution follows a power-law [7], (b) its maximum degree is $\geq n^{\frac{1}{2}-\epsilon}$ [13], (c) its diameter is $\Theta(\log n / \log \log n)$ (for $m \geq 2$), and (d) its cover time is $\Theta(n \log n)$ (again for $m \geq 2$).

An interesting property of the PA graphs is their robustness with respect to node deletion. In [4,14] the authors study the size of the largest component of the PA graphs after random and adversarial node deletions. In [2] the authors study the virus-spreading problem on graphs very similar to PA graphs, relating it to spreading of computer viruses over the internet.

It is natural to ask whether expansion and high conductance imply that rumor spreading is fast. The graph in Figure 1 has high edge expansion, but rumor spreading takes linearly many rounds. The graph consists of $\sqrt{n}$ many independent sets, each of size $\sqrt{n}$. These independent sets are arranged in a cycle. Two neighbouring independent sets form a complete bipartite graph. The central node is connected to one vertex in each independent set.

The graph also has high edge expansion but PUSH-PULL requires polynomially many rounds in spite of the diameter being constant. Whether high conductance by itself implies the success of PUSH-PULL in general appears to be an intriguing and difficult open problem.

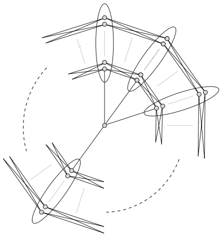


Fig. 1. Slow rumor spreading in spite of high edge expansion

Mihail et al. [16] study the edge expansion and the conductance of graphs that are very similar to PA graphs. We shall refer to these as "almost" PA-graphs [2]. They show that edge expansion and conductance are constant in these graphs, when $m \geq 2$.

High conductance implies that *non-uniform* rumor spreading succeeds. By non-uniform we mean that, for every ordered pair of neighbours i and j, node i will select j with probability p_{ij} for the rumor spreading step (in general, $p_{ij} \neq p_{ji}$). Boyd et al. [3] consider the "averaging" problem on general graphs, which is closely related to the convergence of PUSH-PULL. A corollary of their main results is that, if the p_{ij} are suitably chosen, non-uniform PUSH-PULL rumor spreading succeeds within $O(\log n)$ rounds in almost-PA graphs. They also show that this distribution can be found efficiently using local computations in these graphs, but their method requires $\Omega(\log n)$ steps. While the contribution of [3] is noteworthy, this corollary is in our context somewhat trivial. That such a probability distribution exists is straightforward. Because of their high conductance, almost-PA graphs have diameter $O(\log n)$. Thus, in a synchronous network, it is possible to elect a leader in $O(\log n)$ many rounds and set up a BFS tree originating from it. By assigning probability 1 to the edge between a node and its parent one has the desired non uniform probability distribution. Thus, from the point of view of this paper the existence of non uniform problem is rather uninteresting. Boyd et. al [3] also show that this distributions can be found efficiently using local computations, but their method requires $\Omega(\log n)$ many steps. The local computations of each node, at every step, include a broadcasting of some value to all neighbours. Local broadcasting, used for diameter (that is, $O(\log n)$) many rounds, is a trivial information-dissemination strategy.

Also, Mosk-Aoyama and Shah [17] consider the problem of computing separable functions. In particular, they consider the uniform rumor spreading problem on graphs weighted by a high-conductance doubly-stochastic matrix "that assigns equal probability to each of the neighbours of any node" (that is, if p_{ij} is the probability that node i initiates a connection to node j in the generic round t, then $\forall ij \in E(G)$ $p_{ij} = p_{ji} = \Delta^{-1}$, where Δ is the maximum degree in the graph). Their work implies that if the conductance (or the edge expansion) of a graph is $\Omega(1)$ then rumor spreading ends in $O(\Delta \log^2 n)$ many rounds — this, while being a good bound for constant-degree graphs, is polynomially large for PA graphs (where the bound would be larger than $\Omega(n^{\frac{1}{2}-\epsilon})$).

3 Preliminaries

Preferential attachment graphs were intuitively introduced in [1] and formally defined in [7], from which the following definition is taken.

Definition 1. [PA model]. *Let G_n^m, m being a fixed parameter, be defined inductively as follows:*

- *G_1^m consists of a single vertex with m self-loops.*
- *G_n^m is built from G_{n-1}^m by adding a new node u together with m edges $e_u^1 = (u, v_1), \ldots, e_u^m = (u, v_m)$ inserted one after the other in this order. Let M_i be the sum of the degree of all the nodes right before the edge e_u^i is added. The endpoint*

v_i is selected with probability $\frac{\deg(v_i)}{M_i+1}$, with the exception of node u that is selected with probability $\frac{\deg(u)+1}{M_i+1}$.

In other words, if a vertex $v \neq u$ has degree d when e^i_u is inserted, it will be chosen with probability proportional to d, while u will be chosen with probability proportional to its current degree plus one. Note that definition allows for self-loops. The rich-get-richer effect is clear, since the higher the degree of a node, the higher the probability that it will be chosen as the endpoint of a new edge.

In the sequel we will say that an event holds *with high probability* if it occurs with probability $1 - o(1)$, where $o(1)$ will be a quantity going to zero as n, the number of vertices of the graph under consideration, grows.

Definition 2. *Consider a rumor spreading strategy (*`PUSH`*,* `PULL` *or* `PUSH-PULL`*). Given a (random) graph of n nodes, we say that the strategy* succeeds *if the message is delivered within poly-logarithmically, in n, many rounds, regardless of the starting node, with probability $1 - o(1)$. It* fails *if, with non-vanishing probability, there exist a node from which the message requires polynomially-many rounds to be delivered to all nodes (i.e. it requires $\Omega(n^\alpha)$ many rounds for some $\alpha > 0$).*

4 Rumor Spreading in Social Networks

We begin by showing some simple lower bounds for the performance of `PUSH` and `PULL` acting alone, and that the condition $m \geq 2$ is necessary. This discussion will provide the motivation to analyse the `PUSH-PULL` strategy.

The requirement $m \geq 2$ is due to the fact that G^1_n is disconnected with high probability. The next lemma is implicit in [6].

Lemma 1. *G^1_n is connected with probability*

$$\frac{\sqrt{\pi}}{2} \cdot \frac{\Gamma(n)}{\Gamma(n+1/2)} = \Theta\left(\frac{1}{\sqrt{n}}\right),$$

where Γ denotes the gamma function ($\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt$).

Proof. The graph is connected iff no node, except the first, has a self-loop. The probability of this event is $\prod_{i=2}^{n} \frac{2i-2}{2i-1}$, which is equivalent to the expression in the claim. □

Next we characterize the performance of `PUSH` and `PULL`. Fix $\epsilon > 0$. We say that a node is of *high degree* if its degree is $> n^\epsilon$. The next lemma says that, for a suitably small ϵ, there are lots of high degree nodes in the graph. More precisely, with high probability, the sum of their degrees is $\Omega(n^{1-\epsilon})$. To prove the lemma we need a sharp estimate of $E[X^n_k]$, where X^n_k is the number of nodes of degree k in G^1_n. [7] gives the bound $E[X^t_i] = (1 \pm o(1))\frac{4t}{i(i+1)(i+2)}$ but this is not sufficient for our purposes. We require a bound of the form $E[X^t_i] \leq \frac{4t}{i(i+1)(i+2)} + c$, for some absolute constant c (say, $c = 2$).

Lemma 2. *Fix $\epsilon > 0$ sufficiently small. Then, with high probability, the sum of the degrees of nodes that in G^1_n have degree $> n^\epsilon$, is $\Omega(n^{1-\epsilon})$.*

The proof is omitted from this extended abstract for lack of space.

Theorem 1. *Both* PUSH *and* PULL *fail on* $\{G_n^m\}$.

Proof. If $m = 1$ the claim is implied by Lemma 1. We assume $m \geq 2$. Let us consider G_{2n}^m. We show next that, if we consider the nodes inserted after time n, at least $\Omega(n^\alpha)$ of them are connected only to high degree nodes, for some constant $\alpha > 0$.

We will think of every edge uv as a pair of "half-edges", going out of u and v respectively. A half-edge is *good* if, at time n, it goes out of a high degree node. Consider now a node u arrived at time $t > n$. Choosing a neighbour for u is equivalent to choosing a half-edge uniformly at random in G_{t-1}^m. We say that u is *good* if the half-edges it chooses are all good. Note that the events of being good for each of the nodes from $n + 1$ to $2n$ are independent.

As in [7], we can think of G_n^m in the following, equivalent way. Generate G_{nm}^1 and then collapse into one node each sequence of m consecutive nodes. Clearly, the degree of a node in G_n^m is at least as large as the degree of the nodes G_{nm}^1 it comes from. The sum of degrees of high degree nodes in G_n^m is no less than the analogous sum in G_{nm}^1.

By Lemma 2, the probability of a node being good is at least

$$\left(\Omega\left(\frac{n^{1-\epsilon}}{n}\right)\right)^m \geq \Omega(n^{-m\epsilon}).$$

We choose ϵ in such a way that $(m + 1)\epsilon < 1$. Say, $0 < \epsilon \leq 1/(m + 2)$.

Now take the last $\Theta(n^{(m+1)\epsilon})$ nodes. Among those, by Chernoff-Hoeffding bound, at least $\Omega(n^\epsilon)$ are good, with high probability. Let v_t be one of them and suppose that it was inserted at time $t \geq 2n - \Theta(n^{(m+1)\epsilon})$.

The probability that v_t is not chosen as a neighbour by nodes inserted at later times is at least

$$\prod_{i=mt+1}^{2mn}\left(1 - \frac{m}{2i-1}\right) > \prod_{i=mt+1}^{2mn}\left(1 - \frac{m}{2mt}\right) \geq \left(1 - \frac{1}{2n}\right)^{m\Theta(n^{(m+1)\epsilon})} = 1 - o(1).$$

In other words, with high probability, v_t is only connected to m nodes of high degree.

So, suppose the PULL algorithm is being used. If the source of the message is v_t, then the message will not be passed to any other node in time $< o(n^\epsilon)$ with high probability because its m neighbours all have high degree — PULL does not work.

Analogously, if we place the message in $u \neq v_t$, since the only way to reach v_t is via m high degree nodes, PUSH will not be able to route the message to v_t in $o(n^\epsilon)$ time. □

The previous theorem provides strong motivation to analyse the PUSH-PULL strategy.

5 Push and Pull Acting Together

In view of Lemma 1 we assume $m \geq 2$ from now on. Our aim is to show the following.

Theorem 2. *Let $m \geq 2$. Then,* PUSH-PULL *succeeds for $\{G_n^m\}$ within $O(\log^2 n)$ many rounds.*

The proof will be based on several structural facts concerning PA graphs. Some are taken from the existing literature, while new ones will have to be established. We will use the following from [6].

Theorem 3. *Let $m \geq 2$. The diameter of G_n^m is $O(\log n / \log \log n)$ with high probability.*

Consider a connected graph with n nodes in which every node has degree $O(\log n)$ and that has $O(\log n)$ diameter. It is easy to see that in such a graph rumor spreading succeeds within $O(\log^2 n)$ rounds. The next two lemmas say that a PA graph contains a linear size subgraph of this kind. Their proofs are postponed to the next section. Here they are used to prove Theorem 2.

Lemma 3. *Let $m \geq 2$ and let $\epsilon > 0$ be sufficiently small. Then, there exists a set of vertices $W \subseteq V(G_n^m)$, such that: (a) W only contains nodes added after time ϵn and before time $n/2$, (b) $|W| \geq \epsilon n$, and (c) every pair of vertices $x, y \in W$ are connected by a path of length $O(\log n)$ consisting solely of edges inserted between time ϵn and $3/4n$.*

The next lemma roughly says that if a vertex is not a hub by time ϵn it will never be (this includes the case of nodes inserted after that time).

Lemma 4. *Let $m \geq 2$ and fix any $\epsilon > 0$. Then, with high probability the following holds: (a) Every node added after time ϵn has degree $O(\log n)$ in G_n^m, and (b) For every $c' > 0$ there exists $c > c'$ such that, if a node has degree $\leq c' \log n$ in $G_{\epsilon n}^m$ its degree in G_n^m will be $< c \log n$.*

The following lemma follows from lemmas 3 and 4. It says that hubs are at distance at most 2 from W.

Lemma 5. *Let W be as in Lemma 3 and let $m \geq 2$. Then, there exists a sufficiently large constant c such that, with high probability, every node $v \in V(G_n^m)$ with degree $\geq c \log n$ is at distance ≤ 2 from W.*

Proof. Let H be the set of vertices inserted before time ϵn and let R be the set of vertices inserted after time $\frac{3}{4}n$. Recall that W is a subset of the vertices inserted after H and before R. A vertex $v \in R$ is *good* if it is connected to W with its first edge. Our aim is to show that R contains $\Theta(n)$ good vertices. Now, given any vertex in W we will consider only the m half-edges going out of it when this vertex joined the graph. Even with this limitation, the first edge choice of $v \in R$ hits W with probability at least

$$\frac{|W|m}{2mn}.$$

By Lemma 3(b) this is at least some constant $1 > p > 0$. By our previous choice concerning the half-edges of vertices in W, being good is an independent Bernoulli trial. It follows from the Chernoff-Hoeffding bounds (and stochastic domination) that the number of good nodes is $\Theta(n)$ with high probability.

So far we have exposed only the random choice of the first edge of every good node. Thus, there remain $\Theta(n)$ other edges going out of good nodes that are to be fixed. We will use this to prove that every vertex in H whose degree is $\geq c \log n$ in G_n^m is connected to a good node with probability at least $1 - O\left(n^{-2}\right)$. Once this is done the claim will follow from the union bound, since $|H| < n$.

Let v' be of degree $\geq c \log n$ (the value of c will be fixed later). By Lemma 4 this vertex must have been added before time ϵn, i.e. $v' \in H$, and must have had degree $\geq c' \log n$ at that time. Let z be a good node inserted at time $t > \frac{3}{4}n$. The second edge choice of z selects v' with probability at least

$$\frac{\text{(degree of } v' \text{ at time } t\text{) - } m}{2m(\#\text{ edges at time } t)} \geq \frac{c' \log n - m}{2mn} = \frac{c'' \log n}{n}$$

These choices are Bernoulli trials with total expectation $\geq c''' \log n$ (that is the number of good nodes times $c'' \frac{\log n}{n}$). It follows from the Chernoff-Hoeffding bounds that c (and consequently c', c'', c''') can be chosen in such a way that the probability that v' has no neighbour among the good vertices is at most $O\left(\frac{1}{n^2}\right)$. The claim follows by union bound. □

Given the previous lemmas, we can prove Thm. 2 in the following way.

Proof (of Thm. 2). Let us partition $V(G_n^m)$ into three classes. Let $H \subseteq V$ be composed of the nodes of degree $\Omega(\log n)$, let $W \subseteq V$ be as in lemma 3 (note that by lemma 4, w.h.p. $W \cap H = \varnothing$) and let $R = V - H - W$.

Suppose that the information starts in some node of H. Then, by the PULL strategy the information will be taken by at least one node in W in time $O(\log^2 n)$ (by the maximum degree of nodes in W, a coupon collector argument and lemma 5).

Suppose instead that the information starts in some node of R. The distance between one node of R and one node in $V - R$ is at most $O(\log n)$ by the diameter bound of theorem 3. Also, each of the nodes in R has degree $O(\log n)$ by definition. Thus, in time $O(\log^2 n)$ (= maximum distance × maximum degree, see [11]) the information will reach W (either directly or by passing through H) by the PUSH and PULL strategies.

So, we can assume that after $O(\log^2 n)$ steps the information is in W. Each node added after time ϵn has degree $O(\log n)$ and the diameter of W, even after projecting on $W \cup R$, is $O(\log n)$. Thus if at some point a node in W has the information, after $O(\log^2 n)$ steps the information will have reached all nodes in W by the PUSH strategy.

By lemma 5, if each node in W has the information, after $O(\log^2 n)$ steps the information will have been passed to each of the nodes in H by the PUSH strategy.

After each node in $W \cup H$ has the information, the PULL strategy employed will give the information to each of the nodes in R in time $O(\log^2 n)$. □

6 Proofs

In this section we prove the two main lemmas from the previous section, Lemma 3 and 4. Recall that Lemma 4 was a statement about degrees in G_n^m. The next lemma is the key technical ingredient.

Lemma 6. *Consider a Polya urn process lasting for n steps. Suppose that this Polya urn starts with $R_0 \geq \alpha n$ red balls and $B_0 \leq g(n)$ blue balls, for $\alpha > 0$ and some function $g(n) \in o(n)$. Then, with probability $1 - o(1/n)$, the number of blue balls after the n-th extraction, B_n, will be $B_n \leq c \max\{g(n), \log n\}$, for some constant $c > 0$.*

Proof. The probability that, overall, k blue balls will be added to the urn is

$$\begin{aligned} P[B_{n+B_0+R_0} = k] &= \binom{n}{k} \cdot \frac{B_0 \cdots (B_0 + k - 1) \cdot R_0 \cdots (R_0 + n - k - 1)}{(B_0 + R_0) \cdots (B_0 + R_0 + n - 1)} \\ &= \frac{(B_0 + k - 1)!}{k!\,(B_0 - 1)!} \cdot \frac{(R_0 + n - k - 1)!}{(R_0 - 1)!\,(n - k)!} \cdot \frac{n!\,(B_0 + R_0 - 1)!}{(B_0 + R_0 + n - 1)!} \\ &= \frac{\binom{B_0+k-1}{k} \cdot \binom{R_0+n-k-1}{n-k}}{\binom{B_0+R_0+n-1}{n}} \\ &= f(k; B_0 + R_0 + n - 2, B_0 + k - 1, n) \frac{B_0 + R_0 - 1}{B_0 + R_0 + n - 1}. \end{aligned}$$

Where $f(k; s, t, u)$ is the probability of getting exactly k good elements from a sample without replacement of u elements, from a set of s elements, t of which are good.

Consider the numerator of the second to the last expression, is $h(k) = \binom{B_0+k-1}{k} \cdot \binom{R_0+n-k-1}{n-k}$. By simple algebraic manipulation, we obtain that for integer $k \geq c' \cdot g(n)$, for some $c' > 0$, it holds that $h(k) > h(k + 1)$. Therefore, the whole expression is decreasing for k in that range.

Now, let us bound the "bad" event using r for $r = c' \cdot (g(n) + \log n)$, for some sufficiently large constant $c' > 0$,

$$\begin{aligned} P[B_{n+B_0+R_0} \geq r] &= \sum_{i=r}^{n+B_0} P[B_{n+B_0+R_0} = i] \\ &\leq (n + B_0) P[B_{n+B_0+R_0} = r] \\ &= (n + B_0) f(r; B_0 + R_0 + n - 2, B_0 + r - 1, n) \frac{B_0 + R_0 - 1}{B_0 + R_0 + n - 1} \\ &\leq (n + B_0) \sum_{i=r}^{\infty} f(i; B_0 + R_0 + n - 2, B_0 + r - 1, n) \frac{B_0 + R_0 - 1}{B_0 + R_0 + n - 1} \end{aligned}$$

where the last step allows us to use the well-known tail bound for the hypergeometric sum [10]. Let P denote the probability that at least k good elements are in a sample (without replacement) of u elements, from a set of s elements, t of which are good. Then

$$P \leq 2 \exp\left(-\frac{(k - 1 - ut/s)^2}{k - 1} \right).$$

Note that, in our case,

$$ut/s \leq \frac{n\,(B_0 + r - 1)}{B_0 + R_0 + n - 2} \leq (1 \pm o_n(1)) \frac{c' + 1}{1 + \alpha} \left(g(n) + \log n\right).$$

So, we get that $k - 1 - ut/s \geq (1 \pm o_n(1)) \frac{c'\alpha - 1}{1+\alpha} (g(n) + \log n)$. Thus it is possible to make this lower bound arbitrarily large, by choosing c' bounded away from $\frac{1}{\beta}$. The statement follows. □

We show how Lemma 6 implies lemma 4. Take any node v having degree less than $c \log n$ at time ϵn. The graph process can be seen as the following urn process. At time ϵn, the urn contains a number of blue balls equal to d_v, the degree of v, and $\epsilon n - d_v$ red balls. The urn process works as follows. At each new time step, a red ball is added to the urn. Then a random ball is extracted. The time step ends after this ball, and a new one of the same color, are added to the urn.

The number of blue balls of the a Polya urn process (with the same starting conditions, and the same running time) dominates the number of blue balls of the urn process just described.

This proves the second part of lemma 4. The first part follows by noting that, just after having added the generic node j, the degree of j is upper bounded by $2m$. Suppose that v was added after time ϵn. Then, the Polya urn process of lemma 6, with an initial urn of $\log n$ blue balls and ϵn red ones, trivially dominates the degree of v.

We now move on to proving Lemma 3. The next lemma says that, given any positive integer k, any tree can be partitioned into connected components of size k (with the exception of one component) and diameter at most $2k$. Later we will use this to cluster a linear size subgraph of G_n^m.

Lemma 7. *Fix some integer $k \geq 1$. The nodes of any tree can be partitioned into connected components called macronodes in such a way that: (a) each macronode, except at most one, contains k nodes, and (b) the diameter (in the tree) of each macronode is at most $2k$.*

Proof. Fix a root, and label each node with the number of nodes of the subtree rooted there. Pick a node v having the smallest label ℓ greater than or equal k. If $\ell = k$, then the subtree rooted at v will form a macronode. If $\ell > k$ then, consider the levels of the subtree rooted at v. We will put into the macronode the nodes of the levels, in ascending order, in such a way that the number of nodes in the macronode will be k. If the number of nodes in current level, plus the already inserted nodes, exceeds k, take any subset of the nodes of the current level in order to reach k. What is the diameter of a macronode? First of all note that the height of the tree rooted at v is at most k (otherwise, the subtree rooted in a child of v would have a number of nodes no less than k but smaller than the subtree rooted at v). Thus, the maximum distance between two nodes in the subtree is bounded by $2k$. □

We finally come to the proof of Lemma 3. The main thrust of the proof is to show that G_n^m contains a linear-size, low degree expander of type $G(N, M)$. Note that the existence of a graph of linear size that is "almost" of type $G(N, M)$ was already proven in [5], where "almost" means that a linear, albeit small, number of edges may be added and/or deleted from $G(N, M)$. These edges introduce complications, but what makes Lemma 3 is that the proof in [5] holds only if m is a very large constant, while we assume $m \geq 2$.

Proof (of Lemma 3). By theorem 3.2 of [4], w.h.p. there exists a subset W' of the nodes of $V(G_n^m)$ such that: (a) W' contains nodes added after time ϵn and before time $n/2$, (b) $|W'| \in \Theta(n)$, and (c) $V(G_n^m)$ projected on W' is connected.

Let us fix such a W'. Take any spanning tree of W' and partition it in macronodes of size $s = \lceil ((4mn)/|W'|)^2 \rceil \in O(1)$ as shown in lemma 7. (Recall that the last macronode may not have the required size. In that case, we remove its nodes from W'.) By the lemma, the diameter of each of the macronodes in $V(G_n^m)$ is at most $O(1)$.

We will show that the macronodes will be connected thanks to an Erdös-Renyi-like random graph $G(N, M)$ (a graph having N nodes and M edges chosen UAR between those with these properties) and some other edges. Also, $M \geq (1/2 + \epsilon)N$: by a theorem of [12] this implies that, this $G(N, M)$ will contain a giant component (i.e., a component of size $\Omega(n)$) of diameter $O(\log n)$. As the diameter of a macronode is $O(1)$, this will imply the main statement.

Consider nodes added between $\lceil n/2 \rceil$ and $3/4n - 1$. Each of those nodes will choose the first 2 of its m neighbours by selecting the outgoing edges of the nodes in W' with probability at least $(|W'|/(2mn))^2$. If such an event occurs we say that a "pseudo-edge" is added between the macronodes containing the two selected vertices. The macronodes, and the pseudo-edges, will compose the Erdös-Renyi-like random graph $G(N, M)$.

Consider the auxiliary graph in which macronodes are vertices and two of them are connected by an edge if there is a pseudo-edge connecting them. The number t of such macronodes in W' is $t \leq |W'|/s$. The number of pseudo-edges added between macronodes is, with high probability, at least $(|W'|/(2mn))^2 \cdot n/4 = |W'|^2/(16m^2 n)$. As the latter is greater than $(1/2 + \epsilon)|W'|/s$, it follows from the results of [12] that in the auxiliary graph there exists a giant component having diameter $O(\log n)$ and size $\Omega(n)$. The claim follows by choosing W as the set of nodes that make up the macronodes. □

7 Conclusion

We have shown how fast the PUSH-PULL strategy disseminates some information throughout the nodes of a PA graph, and how slow the PUSH, PULL strategies obtain the same result. In doing so, we have proved some lemmas that strengthen the connection between the PA random graphs and classical Erdös-Renyi random graphs.

We believe that our results might offer some insights on real rumor spreading among humans. Namely, it seems plausible that in a social network there exists a "core" of people that might not be VIPs, and yet collectively are able to reach a majority of their community in a few steps. Also, our proofs indicate how VIPs are only important for relaying the information and not as originators of rumours (that is, even if they never started a communication themselves, the information would still spread through the network — just by people asking and telling them what they know) .

Acknowledgements

We would like to thank Benjamin Doerr and D. Sivakumar for several useful discussions on this problem.

References

1. Barabási, A.L., Albert, R.: Emergence of Scaling in Random Networks. Science 286(5439), 509–512 (1999)
2. Berger, N., Borgs, C., Chayes, J.T., Saberi, A.: On the spread of viruses on the internet. In: SODA 2005, pp. 301–310 (2005)
3. Boyd, S.P., Ghosh, A., Prabhakar, B., Shah, D.: Gossip algorithms: design, analysis and applications. In: INFOCOM 2005, pp. 1653–1664 (2005)
4. Bollobás, B., Riordan, O.: Robustness and vulnerability of scale-free random graphs. Internet Mathematics 1, 1–35 (2003)
5. Bollobás, B., Riordan, O.: Coupling Scale-Free and Classical Random Graphs. Internet Mathematics 1(2) (2003)
6. Bollobás, B., Riordan, O.: The Diameter of a Scale-Free Random Graph. Combinatorica 24(1) (January 2004)
7. Bollobás, B., Riordan, O., Spencer, J., Tusnády, G.: The degree sequence of a scale-free random graph process. Random Structures & Algorithms 18(3) (May 2001)
8. Cooper, C., Frieze, A.M.: The cover time of the preferential attachment graph. J. Comb. Theory, Ser. B 97(2), 269–290 (2007)
9. Demers, A.J., Greene, D.H., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H.E., Swinehart, D.C., Terry, D.B.: Epidemic Algorithms for Replicated Database Maintenance. In: PODC 1987, pp. 1–12 (1987)
10. Dubhashi, D., Panconesi, A.: Concentration of Measure for the Analysis of Randomised Algorithms. Cambridge University Press, Cambridge (2009)
11. Feige, U., Peleg, D., Raghavan, P., Upfal, E.: Randomized Broadcast in Networks. Random Struct. Algorithms 1(4), 447–460 (1990)
12. Fernholz, D., Ramachandran, V.: The diameter of sparse random graphs. Random Struct. Algorithms 31(4), 482–516 (2007)
13. Flaxman, A., Frieze, A.M., Fenner, T.I.: High Degree Vertices and Eigenvalues in the Preferential Attachment Graph. Internet Mathematics 2(1) (2005)
14. Flaxman, A., Frieze, A.M., Vera, J.: Adversarial deletion in a scale free random graph process. In: SODA 2005, pp. 287–292 (2005)
15. Frieze, A., Grimmett, G.: The shortest-path problem for graphs with random arc-lengths. Discrete Applied Mathematics 10, 57–77 (1985)
16. Mihail, M., Papadimitriou, C.H., Saberi, A.: On Certain Connectivity Properties of the Internet Topology. In: FOCS 2003, pp. 28–35 (2003)
17. Mosk-Aoyama, D., Shah, D.: Fast Distributed Algorithms for Computing Separable Functions. IEEE Transactions on Information Theory 54(7)
18. Pittel, B.: On spreading a rumor. SIAM Journal on Applied Mathematics 47(1), 213–223 (1987)
19. Doerr, B., Friedrich, T., Sauerwald, T.: Quasirandom Broadcasting. In: Proceedings of SODA 2008, pp. 773–781 (2008)
20. Hromkovic, J., Klasing, R., Pelc, A., Ruzicka, P.: Dissemination of Information in Communication Networks: Broadcasting, Gossiping, Leader Election, and Fault-tolerance. Springer, Heidelberg (2005)

MANETS: High Mobility Can Make Up for Low Transmission Power[*]

Andrea E.F. Clementi[1], Francesco Pasquale[1,*], and Riccardo Silvestri[2]

[1] Dipartimento di Matematica, Università di Roma "Tor Vergata"
{clementi,pasquale}@mat.uniroma2.it
[2] Dipartimento di Informatica, Università di Roma "La Sapienza"
silvestri@di.uniroma1.it

Abstract. We consider *Mobile Ad-hoc NETworks (MANETs)* formed by n nodes that move independently at random over a finite square region of the plane. Nodes exchange data if they are at distance at most r within each other, where $r > 0$ is the *node transmission radius.* The *flooding time* is the number of time steps required to broadcast a message from a *source* node to every node of the network. Flooding time is an important measure of the speed of information spreading in dynamic networks.

We derive a nearly-tight upper bound on the *flooding time* which is a *decreasing* function of the maximal *velocity* of the nodes.

It turns out that, when the node velocity is "sufficiently" high, even if the node transmission radius r is far below the *connectivity threshold*, the flooding time does not asymptotically depend on r. So, flooding can be very fast even though every *snapshot* (i.e. the static random geometric graph at any fixed time) of the MANET is fully disconnected.

Our result is the first *analytical* evidence of the fact that high, random node mobility strongly speed-up information spreading and, at the same time, let *nodes save energy.*

1 Introduction

The impact of node mobility in data propagation is currently one of the major issues in Network Theory. The new trend is to consider node mobility as a *resource* for data forwarding rather than a *hurdle* [14,10]. This is well-captured by the model known as *opportunistic Mobile Ad-Hoc NETworks* (*opportunistic MANET*), an interesting recent evolution of MANET [18,11,10,22]. In opportunistic MANETS, mobile nodes are enabled to communicate even if a route connecting them never exists. Furthermore, nodes are not supposed to have or acquire any knowledge of the network topology (this one being highly-dynamic). Such two features make data communication in opportunistic networks a new challenging research topic from both foundational and practical view-points.

[*] Partially supported by the EU under the EU/IST Project 15964 AEOLUS.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 387–398, 2009.

Inspired by opportunistic MANET, we here consider the *Evolving Graph* yielded by a set of n nodes moving over a finite square. Two nodes can exchange data, at a given time step, if their relative distance at that time step is not larger than a fixed *transmission radius* $r > 0$.

The aim of this work is to investigate the speed of data propagation in the above evolving graph. It is not hard to see that this study must cope with technical problems which are far to be trivial. Just to get an insight of such difficulties, we observe that classic *static* concepts like global *connectivity* and network *diameter* are almost meaningless in this context. On one hand, we can easily construct a sequence of node configurations such that *every* corresponding *snapshot* (i.e. the communication graph at any fixed time step) of the network is not connected while the broadcast task can be performed in a logarithmic number of time steps. On the other hand, it is easy to construct another temporal sequence of node configurations where the snapshot diameter is always 3 while the broadcast task requires $\Theta(n)$ time. Previous experimental works in this topic show that data communication can benefit from node mobility even though *all* the *snapshots* of the MANETs are not connected [11,18,21,14,6]. However, to our knowledge, the only *analytical* evidence of this phenomenon is that proved in [11] concerning *network capacity*, an information-theoretic concept which is not known to be related to the *speed* of data propagation.

We thus believe that, in order to investigate data propagation on evolving graphs (such as MANETs), a different concept and/or performance measure should be adopted and investigated.

In our opinion, a fundamental role in this dynamic world is played by the *Flooding Time*: this is in fact the desired crucial concept/measure. The flooding mechanism is the simple broadcast protocol where every *informed* node sends the source message at every time step (a node is said to be *informed* if it knows the source message). The flooding time of an evolving graph is the first time step in which all nodes are informed. The flooding time is a natural lower bound for any broadcast protocol and it represents the maximal speed of data propagation: the same role of the diameter in static networks. Flooding time of some classes of *Markovian Evolving Graphs* [3] has been recently studied in [7,8].

Our work provides an analytical study of the flooding time of a natural class of MANETS: we prove a nearly-tight bound on the flooding time showing that high, random node mobility can *dramatically speed-up* data propagation with respect to the corresponding static model. This can be seen as a strong and somewhat final improvement of our previous work [8] where we (only) proved that low random node mobility does not *slow-down* data propagation.

Our MANET model: An informal definition. We consider a model of mobile networks called *geometric Markovian Evolving Graphs*, i.e., *geometric-MEG* [8]. It is the discrete version of the well-known *random-walk* model [5,9,12] and can also be viewed as the *walkers* model [9] on the square.

In a *geometric-MEG*, nodes (i.e. radio stations) move over a finite square region of the plane and each node performs, independently from the others, a sort of *Brownian* motion. In our model we make time and space discrete (see

Section 2 for details). The mobility parameter is the *move radius* ρ. At every time step, a node moves uniformly at random to any point which is within distance ρ from its current position; so ρ determines the *maximal node velocity.* At any time there is an edge (i.e. a bidirectional link) between two nodes if they are at distance at most r.

It turns out that a geometric-MEG is a temporal sequence of random disk graphs (*random geometric graphs* [19]). When r is over the *connectivity threshold*, such random graphs are with high probability[1] (in short, *w.h.p.*) connected and have diameter $D < n$ that depends on n and r. In our previous work [8], we proved that when $\rho < r$ (so we are in the case of *low node mobility*) and the latter is over the connectivity threshold, then the flooding time of geometric-MEG is w.h.p. asymptotically equivalent[2] to the diameter of the corresponding snapshots. Under the assumption $\rho < r$, it is not hard to show [8] this is *the best the flooding can achieve*: the result is in fact asymptotically tight. It thus follows that, in the *slow* case, random node mobility does not significantly affect the flooding time with respect to the static case.

Our results. In this work, we consider the case $\rho > r$. This can viewed as a model for opportunistic MANETs where message transmission is not so frequent due to critic environment conditions and/or poor node transmission power while node velocity is high and random. The resulting unit time (i.e. the time interval between two consecutive message transmissions) is enough large so that the move radius ρ is larger than the transmission radius r. We are thus motivated by the futuristic scenario of mobile wireless sensor environments composed of myriads of tiny nodes dispersed in the environment and subject to high, unpredictable mobility as a consequence of environmental dynamics such as wind, storms or water streams. To our knowledge, the impact of such a high node mobility on the speed of data propagation has never been considered, at least from a foundational perspective.

We provide a nearly-tight bound on the flooding time for such a case. Let G be a geometric-MEG of n nodes over a square of edge size[3] $\sqrt{n}$, transmission radius r and move radius ρ such that $r \geq r_0$ and $\rho \geq c\sqrt{\log n}$, where r_0 and c are sufficiently large positive constants. Then, w.h.p., flooding in G is completed within

$$\mathcal{O}\left(\frac{\sqrt{n}}{\rho} + \log n\right) \quad \text{time steps.}$$

It is not hard to show that, for $\rho \geqslant r$, the expected flooding time is $\Omega(\sqrt{n}/\rho)$ [8], so our upper bound is nearly-tight and becomes tight whenever the flooding time is $\Omega(\log n)$.

When the transmission radius r is over the connectivity threshold (i.e. $\Theta(\sqrt{\log n})$), our bound implies that if the move radius ρ is (asymptotically)

[1] As usual, we say that an event $\mathcal{E}$ occurs *with high probability* if $\mathbf{Pr}(\mathcal{E}) \geqslant 1 - 1/n^{\Theta(1)}$.

[2] Actually, our previous bound leaves an $O(\log\log n)$ gap in a small range of the network parameters.

[3] For clarity's sake, we choose here to keep node density constant as the number of nodes grows.

higher than r, the flooding time is (asymptotically) smaller than the diameter of the snapshots. When r is smaller than the connectivity threshold (say, it is a constant r_0) while $\rho \geqslant c\sqrt{\log n}$, then our bound implies that flooding can be efficiently completed despite the snapshot at every time step is formed by several connected components of small sizes [19,13].

In general, our upper bound says that, in this case, *the flooding time does not asymptotically depends on the transmission radius.*

This fact has important technological consequences in the futuristic scenario of large, high-mobile MANETS. The two major goals in MANETS are: i) guarantee good and fast data communication, and ii) minimize node energy-consumption (which is clearly an increasing function of r). It is well-known that in classic, (static or low-mobile) MANETS such two goals are in contrast with each other and, thus, a suitable trade-off must be determined (actually, optimizing this trade-off is currently a major research issue in ad-hoc networking [2,15,20]). In particular, we know that [19,13], in order to guarantee global connectivity (and thus data communication) in static random geometric graphs, the transmission radius must be $\Omega(\sqrt{\log n})$, so it must *increase with the network size.*

In this context, our bound is a strong mathematical evidence of the fact that, when node mobility is relatively high and random, the two above goals are not competing anymore. We can achieve fast data-forwarding by using small transmission radius (so, saving node energy). More importantly, the transmission radius can be an *absolute constant* and, so, it does not need to increase as the network size does. The technology of node transmitter devices can be thus *scalable.* Observe that node mobility in such opportunistic networks is due to the *host* mobility which is often fully independent from sensor devices: high sensor mobility does not (necessarily) imply high energy consumption [10,18].

Adopted Techniques. The bound in [8] on the flooding time for the *slow* case is achieved thanks to the expanding properties of the *connected* snapshots of the geometric-MEG which are, in turn, guaranteed by two facts: 1) The stationary node distribution at every time step is almost uniform; and 2) the transmission radius r is over the connectivity threshold. In particular, they imply that, starting from the second time step, the number of informed nodes is large enough to apply standard Chernoff-like bounds. This allowed us to evaluate the number of *new* informed nodes at any successive step. The role of node mobility is thus shown to have a negligible impact on the flooding process.

This scenario is no longer true when r is below the connectivity threshold (say constant). The snapshots of the geometric-MEG are very sparse and disconnected and, hence, their expanding properties are very scarce. This results into a relatively-long initial phase (called *Bootstrap*) of the flooding process where the number of informed nodes is not large enough to get useful concentration results from Chernoff-like bounds. When $\rho >> r$, the flooding process is mainly due to *node mobility* that, roughly speaking, brings the source information *outside* the small connected components of the sparse snapshots: We provide a clean analytical description of this phenomenon. A key-ingredient here is a probabilistic analysis of the Bootstrap. We present a set of probabilistic lemmas for *almost-increasing random processes* that allows us to evaluate, at every time step, the number of new informed nodes even when the latter has a small expected value. The rather general form of such lemmas might result useful in other similar situations where Chernoff's-like bounds are useless.

Due to lack of space, all the proofs are omitted. A full version of this paper can be found online at `http://arxiv.org/abs/0903.0520`.

2 The Node Mobility Model

We consider a model of dynamic graphs, introduced in [8], that is a discrete version of the *random walk mobility* model for radio networks [5]. In the latter model, nodes (i.e. radio stations) move on a bounded region of the plane (typically a square region) and each node performs, independently from the others, a sort of Brownian motion. At any time there is an edge (i.e. a bidirectional connection link) between two nodes if they are at distance at most r (r represents the *transmission radius*). In our model we discretize time and space. We choose to keep the density constant (i.e. the ratio between the number of nodes and the area) as the number n of nodes grows. Nodes move over a square of side $\sqrt{n}$ and so the density equals to 1. We remark that this choice is only for clarity's sake: all our results can be scaled to any density $\delta(n)$. The nodes can assume positions whose coordinates are integer multiple of a resolution coefficient $\epsilon > 0$. Formally, nodes move on the following set of points

$$L_{n,\epsilon} = \left\{ (i\epsilon, j\epsilon) \mid i,j \in \mathbb{N} \wedge i,j \leqslant \frac{\sqrt{n}}{\epsilon} \right\}$$

At any time step, a node can move to one of the positions of $L_{n,\epsilon}$ within distance ρ from the previous position. The positive real number ρ is a fixed parameter that we call *move radius*. It can be interpreted as the maximum velocity of a node[4]. Formally, we introduce the *move graph* $M_{n,\rho,\epsilon} = (L_{n,\epsilon}, E_{n,\rho,\epsilon})$, where

$$E_{n,\rho,\epsilon} = \{ (\mathbf{x}, \mathbf{y}) \mid \mathbf{x}, \mathbf{y} \in L_{n,\epsilon} \;\; d(\mathbf{x}, \mathbf{y}) \leqslant \rho \}$$

and $d(\cdot,\cdot)$ is the Euclidean distance. A node in position $\mathbf{x}$, in one time step, can move in any position in $\Gamma(\mathbf{x})$, where $\Gamma(\mathbf{x}) = \{\mathbf{y} \mid (\mathbf{x},\mathbf{y}) \in E_{n,\rho,\epsilon}\}$. The nodes are identified by the first n positive integers $[n]$. The time-evolution of the movement of a single node i is represented by a Markov chain $\{P_{i,t}\ ;\ t \in \mathbb{N}\}$ where $P_{i,t}$ are random variables (in short *r.v.*) whose state-space is $L_{n,\epsilon}$ and

$$\mathbf{Pr}\left(P_{i,t+1} = \mathbf{x}\right) \quad = \quad \begin{cases} \frac{1}{|\Gamma(P_{i,t})|} & \text{if } \mathbf{x} \in \Gamma(P_{i,t}) \\ 0 & \text{otherwise} \end{cases}$$

In other words, $P_{i,t}$ is the position of node i at time t. Thus, the time evolution of the movements of all the nodes is represented by a Markov chain $\mathcal{P}(n,\rho,\epsilon) = \{P_t\ :\ t \in \mathbb{N}\}$ whose state space is $L_{n,\epsilon} \times L_{n,\epsilon} \times \cdots \times L_{n,\epsilon}$ (n times) and

$$P_t = (P_{1,t}, P_{2,t}, \ldots, P_{n,t})$$

[4] Indeed, a node can run through a distance of at most ρ in a unit of time.

Let us fix a *transmission radius* $r > 0$. A *geometric-MEG* is a sequence of random variables $\mathcal{G}(n, \rho, r, \epsilon) = \{G_t \; : \; t \in \mathbb{N}\}$ such that $G_t = ([n], E_t)$ with

$$E_t = \{(i, j) \mid d(P_{i,t}, P_{j,t}) \leqslant r\}$$

As for the stationary case, we observe that the stationary distribution π_i of Markov chain $\{P_{i,t} \; ; \; t \in \mathbb{N}\}$ is (see [1])

$$\pi_i(\mathbf{x}) \quad = \quad \frac{|\Gamma(\mathbf{x})|}{\sum_{\mathbf{y} \in L_{n,\epsilon}} |\Gamma(\mathbf{y})|}$$

Moreover, the stationary distribution of $\mathcal{P}(n, \rho, \epsilon)$ is the product of the independent distributions π_i for all $i \in [n]$. We say that a geometric-MEG $\mathcal{G}(n, \rho, r, \epsilon) = \{G_t \; : \; t \in \mathbb{N}\}$ is a *stationary* geometric-MEG if the underlying P_0 is random with the stationary distribution of the Markov chain $\mathcal{P}(n, \rho, \epsilon) = \{P_t \; : \; t \in \mathbb{N}\}$. Notice that if $\mathcal{G}(n, \rho, r, \epsilon) = \{G_t \; : \; t \in \mathbb{N}\}$ is a stationary geometric-MEG then all G_ts are random with the same probability distribution that we call *stationary distribution* of $\mathcal{G}(n, \rho, r, \epsilon)$.

In the rest of the paper, we will always assume that the move radius ρ is not larger than $\sqrt{n}$.

3 Bounding the Flooding Time

In the flooding mechanism, every informed node sends the source message at every time step: so, all nodes that are within distance r from an informed node will be informed at the next time step. For the sake of simplicity, every time step is divided into two consecutive actions: i) the *move action*, where nodes make their random move, and ii) the *transmission action*, where the informed nodes send the source message. Clearly this assumption does not affect the asymptotical bound on the flooding time.

Our result can be formally stated as follows.

Theorem 1. *Let $\mathcal{G}(n, \rho, r, \epsilon)$ be a stationary geometric-MEG. If $r \geqslant r_0$ and $\rho \geqslant c\sqrt{\log n}$ for sufficiently large constants r_0 and c, then the flooding time is w.h.p.*

$$\mathcal{O}\left(\frac{\sqrt{n}}{\rho} + \log n\right).$$

3.1 Proof's Overview

The proof consists of a probabilistic analysis of the number of new informed nodes at every time step of the flooding process. In order to cope with this analysis, the temporal process is organized in three consecutive *phases*. Even though it is likely that, in the real process, these phases happen simultaneously rather than consecutively, our analysis yields the desired upper bound. The phases depend on the current number of informed nodes and on the "locality-degree" of the process. As for the latter, we need to partition the square into equal *supercells*, i.e., subsquares of side length $L = \Theta(\rho^2)$. This partition guarantees that any node v in a supercell S, after the move-action, can reach *any* position in *any* neighboring supercell with *almost-uniform* probability. Another crucial property yielded by the partition is that, for the first - say - $\mathcal{O}(n)$ time steps, every supercell will contain $\Theta(\rho^2)$ nodes, w.h.p.

The Bootstrap Phase. In this initial phase, we start our analysis focussing on what happens inside the neighborhood of the supercell S_0 containing the source, i.e., the supercell set $N(S_0)$ formed by S_0 and its adjacent supercells. We can say that, with positive-constant probability, S_0 contains $\Theta(r^2)$ informed nodes after the first time step. Observe that this is the crucial analysis point where we need to go from *positive-constant probability* to *high probability* and we cannot use Chernoff-like bounds. Indeed, in the successive time steps $t > 0$ of this phase, we consider the flooding-rate inside the supercell S_t' having the maximal number of informed nodes at time step t. We will then prove that, after $t = \mathcal{O}(\log n)$ time steps, there will be (at least) one supercell *quasi-informed* w.h.p., i.e., it will have $\Theta(\rho^2)$ informed nodes.

The Spreading Phase. After the Bootstrap, we can thus assume (w.h.p.) that there is (at least) one supercell quasi-informed. We can thus look at the flooding from a quasi-informed supercell to its adjacent ones. We show that, w.h.p., if a supercell is quasi-informed at a given time step, then all its adjacent supercells will be quasi-informed within the next time step. Since we prove that the *boundary* of any supercell set D has size at least $\Omega(\sqrt{|D|})$, it turns out that this flooding phase makes all the supercells quasi-informed within $\mathcal{O}(\sqrt{n}/\rho)$ time steps.

The Filling Phase. At the end of the previous phase, we thus have w.h.p. all supercells quasi-informed. The Filling phase consists of the sequence of time steps required to get all supercells informed. We prove that this final process can be completed in $O(\log n)$ time steps, , w.h.p.

3.2 Preliminaries

We need to introduce the following notions.

- The square is partitioned into squared *supercells* of side length L with
$$\frac{\rho}{3\sqrt{2}} \leqslant L \leqslant \frac{\rho}{2\sqrt{2}}$$
- Every supercell is partitioned into squared *cells* of side length ℓ with
$$\frac{r}{1+\sqrt{2}} \leqslant \ell \leqslant \frac{r}{\sqrt{2}}$$
- The neighborhood $N(S)$ of a supercell S is the set of supercells formed by S and all its adjacent supercells.

– A supercell is *quasi-informed* at time t if it contains $\gamma\rho^2$ informed nodes at that time, where γ is a suitable positive constant.

We say that the *density condition* holds at time t if, for every supercell S, the number of nodes in S at time t is at least $\eta\rho^2$, for a suitable constant $0 < \eta < 1$. Let $\mathcal{D}$ be the following event: the density condition holds for every time step $t = 0, 1, \ldots, n$.

The proof of the following lemma is an easy consequence of the almost uniformity of the stationary distribution of geometric-MEG.

Lemma 1. *Let $\mathcal{G}(n, \rho, r, \epsilon)$ be a stationary geometric-MEG. If $\rho \geqslant c\sqrt{\log n}$ for a sufficiently large constant c, then the probability of event $\mathcal{D}$ is at least $1 - 1/n^4$.*

In the rest of this section, we will tacitely assume that event $\mathcal{D}$ holds. Thanks to the previous lemma, since we are conditioning w.r.t. an event that holds w.h.p., the corresponding unconditional probabilities are affected by only a negligible factor.

For the sake of simplicity, we will use the following probability notations. For an event $\mathcal{E}$ and a random variable (*r.v.*) X, the notation

$$\mathbf{Pr}\left(\mathcal{E} \mid X\right) \leqslant p$$

means that, for every possible value x of X, it holds

$$\mathbf{Pr}\left(\mathcal{E} \mid X = x\right) \leqslant p.$$

3.3 The Bootstrap

We now provide an upper bound on the time required to get at least one supercell quasi-informed. We will prove the bound for $r = r_0$ where r_0 is a sufficiently large constant. Observe that, since the flooding time is a non-increasing function of the transmission radius r, the same upper bound holds for any $r \geqslant r_0$ as well.

The following lemma will be used to evaluate the number of new informed nodes after an initial sequence of consecutive time steps.

Lemma 2 (Almost-increasing random processes). *Let $\{X_t : t \in \mathbb{N}\}$ be a sequence of random variables with $X_0 = 1$. Assume that two real values $\alpha > 1$, $0 < \beta < 1$, a positive integer M, and a probability $p \in (0, 1)$ exist such that for every $t \in \mathbb{N}$ it holds that*

$$\mathbf{Pr}\left(X_{t+1} < \alpha X_t \mid X_t < M,\, X_{t-1}, \ldots, X_1\right) \leqslant p \tag{1}$$

$$\mathbf{Pr}\left(X_{t+1} \geqslant \beta X_t \mid X_t,\, X_{t-1}, \ldots, X_1\right) = 1 \tag{2}$$

If $p < \frac{\log \alpha}{e \log(\alpha/\beta)}$ then for any $t \geqslant \frac{\log M}{\log \alpha - e\, p \log(\alpha/\beta)}$ it holds that

$$\mathbf{Pr}\left(\bigcap_{i=1}^{t} \{X_i < M\}\right) \leqslant \exp\left(-pt\right)$$

Notice that the r.v. X_ts can be mutually dependent. Informally speaking, Lemma 2 states that, under Conditions (1) and (2), there is (at least) a time step $\bar{t} = \mathcal{O}(\log n)$ where the process value $X_{\bar{t}}$ reaches the "goal" M w.h.p.

The following lemmas allow us to apply the previous lemma to the flooding process.

For any supercell S, let $m_t(S)$ be the number of informed nodes in S at time step t. For any time step t let $Y_t = \max\{m_t(S) : S \text{ is a supercell}\}$.

Lemma 3. *For any time step t, it holds that*

$$\mathbf{Pr}\left(Y_{t+1} < 2Y_t \mid Y_t < \frac{r_0^2}{28}\right) \leqslant \exp\left(-\frac{r_0^2}{224}\right)$$

Lemma 4. *For any supercell S and for any time step t, it holds that*

$$\mathbf{Pr}\left(m_{t+1}(S) < 2m_t(S) \;\middle|\; \frac{r_0^2}{28} \leqslant m_t(S) < \gamma\rho^2\right) \leqslant 3\exp\left(-\frac{br_0^2}{56}\right)$$

The setting of constant b and the proof of the above lemma are based on the following bound on the number of *infected cells*. We say that a cell C is *infected* at time t if, immediately after the move action of time step t, C contains at least one informed node.

Lemma 5. *Positive constants a and b exist such that, for any time step t and for any supercell S, if at the beginning of time step t a supercell $S' \in N(S)$ contains m informed nodes, then*

$$\mathbf{Pr}\left(Z \leqslant am'\right) \leqslant \exp\left(-bm'\right)$$

where Z is the random variable counting the number of infected cells of S at time t and $m' = \min\{m, L^2/\ell^2\}$.

The proofs of Lemmas 4 and 5 allow us to fix the constant γ defining quasi-informed supercells.

Lemma 6 (Bootstrap Time). *Within $\mathcal{O}(\log n)$ time steps there is a quasi-informed supercell w.h.p.*

Proof. By using simple geometric arguments, if we choose $\beta = 1/121$, then it holds that $Y_{t+1} \geqslant \beta Y_t$ with probability one, thus satisfying Hypothesis (2) of Lemma 2. From Lemmas 3 and 4, it is easy to see that the r.v. Y_ts satisfy Hypothesis (1) of Lemma 2 with constants $\alpha = 2$, $M = \gamma\rho^2$ and $p = 3\exp\left(-b\frac{r_0^2}{224}\right)$. Hence, for a sufficiently large constant r_0, the thesis follows by applying Lemma 2. □

3.4 The Spreading Phase

Lemma 7 (Local-supercell spreading). *For any supercell S, if $m_t(S) \geqslant \gamma\rho^2$ then the event $m_{t+1}(S') \geqslant \gamma\rho^2$ holds for any supercell $S' \in N(S)$ with probability at least $1 - 1/n^4$.*

In order to prove a bound on the number of time steps that are sufficient to guarantee (with high probability) that one quasi-informed supercell spreads the information to all the supercells, we need two lemmas. The first one will provide a bound on the number of supercells that are adjacent to an arbitrary set of supercells.

Let Q be a $m \times m$ square grid, that is, Q is a square partitioned into $m \times m$ congruent sub-squares, called cells. For any subset B of the cells of Q, define the *boundary* ∂B of B as the set of all the cells that do not belong to B and that are adjacent to some cell in B:

$$\partial B = \{c \mid c \notin B \ \wedge \ \exists c' \in B : \ c' \text{ is adjacent to } c\}.$$

Lemma 8 (Boundary size). *Let Q be a $m \times m$ square grid and let B be any subset of the cells of Q. It holds that*

$$|\partial B| \ \geqslant \ \sqrt{\min\{|B|, m^2 - |B|\}}.$$

The second lemma will allow us to prove an upper bound on the number of steps to get all the supercells quasi-informed when, in one time step, the information propagates from all the quasi-informed supercells to their adjacent ones.

Lemma 9 (Spreading time I). *Let K be any integer with $K \geqslant 1$ and let $\{q_t \mid t \in \mathbb{N}\}$ be a sequence of integers such that $q_0 \geqslant 1$, for every $t \geqslant 0$, $q_t \leqslant K$ and $q_{t+1} \geqslant q_t + \sqrt{\min\{q_t, K - q_t\}}$. Then, it holds that, for every $t \geqslant 5\sqrt{K}$, $q_t = K$.*

By combining Lemmas 7, 8 and 9 we get the following bound.

Lemma 10 (Spreading time II). *If at time $t_1 \leqslant n/2$ there is at least one quasi-informed supercell then, with probability at least $1 - \frac{1}{n^2}$, at every time t with $t_1 + 22\frac{\sqrt{n}}{\rho} \leqslant t \leqslant n$, all the supercells are quasi-informed.*

3.5 The Filling Phase

We first prove that a node not yet informed and belonging to a quasi-informed supercell will get informed in one time step, with a positive-constant probability.

Lemma 11. *There exists a constant $\beta > 0$ such that, for any node u, if at the beginning of a time step t the supercell that contains u is quasi-informed and node u is not informed then, with probability at least β, node u gets informed by the end of time step t.*

From above lemma we can derive a logarithmic upper bound for the filling time.

Lemma 12 (Filling Time). *If a time step $t_2 \leqslant \frac{3n}{4}$ exists such that at every time step t with $t_2 \leqslant t \leqslant n$ all the supercells are quasi-informed, then by time $t_2 + \mathcal{O}(\log n)$ all the nodes are informed, w.h.p.*

Finally, Theorem 1 follows from Lemma 6, Lemma 10, and Lemma 12.

4 Conclusions

Some interesting issues concerning the flooding time of geometric-MEGs are still open. There is a logarithmic gap between our upper bound and the known lower bound [8] when the move radius ρ is very large. Closing this gap is an open problem.

A more challenging open issue is to extend our upper bound to the case where ρ and r are both very small (i.e. below $\sqrt{\log n}$).

Another open research work is to study the flooding time when the starting distribution is not the stationary one but is arbitrary (i.e. a worst-case analysis). We conjecture that the worst-case flooding time is not asymptotically larger than the stationary one for a large range of the network parameters.

Observe that our upper bound can be easily extended to the gossiping task (i.e. the all-to-all communication). It would be interesting to extend our analysis to other basic communication tasks such as data-gathering and routing.

Finally, a major challenge is to obtain similar results for more realistic mobility models [17].

References

1. Aldous, D., Fill, J.: Reversible Markov Chains and Random Walks on Graphs (2002), http://stat-www.berkeley.edu/users/aldous/RWG/book.html
2. Ambühl, C.: An optimal bound for the MST algorithm to compute energy efficient broadcast trees in wireless networks. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1139–1150. Springer, Heidelberg (2005)
3. Avin, C., Koucký, M., Lotker, Z.: How to explore a fast-changing world. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 121–132. Springer, Heidelberg (2008)
4. Azar, Y., Broder, A.Z., Karlin, A.R., Upfal, E.: Balanced allocations. SIAM Journal on Computing 29(1), 180–200 (1999)
5. Camp, T., Boleng, J., Davies, V.: A survey of mobility models for ad hoc network research. Wireless Communication and Mobile Computing 2(5), 483–502 (2002)
6. Chatzigiannakis, I., Kinalis, A., Nikoletseas, S.E., Rolim, J.D.P.: Fast and energy efficient sensor data collection by multiple mobile sinks. In: Proc. of MOBIWAC 2007, pp. 25–32 (2007)
7. Clementi, A., Macci, C., Monti, A., Pasquale, F., Silvestri, R.: Flooding time in edge-markovian dynamic graphs. In: Proc. of 27th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing (PODC 2008), pp. 213–222. ACM Press, New York (2008)
8. Clementi, A., Monti, A., Pasquale, F., Silvestri, R.: Information spreading in stationary markovian evolving graphs. In: Proc. of the 23rd IEEE International Parallel and Distributed Processing Symposium. IEEE Computer Society Press, Los Alamitos (2009)

9. Diaz, J., Mitsche, D., Perez-Gimenez, X.: On the connectivity of dynamic random geometric graphs. In: Proc. of 19th annual ACM-SIAM symposium on Discrete algorithms (SODA 2008), pp. 601–610 (2008)
10. Jain, S., et al.: Exploiting mobility for energy efficient data collection in wireless sensor networks. ACM/Kluwer Mobile Networks and Applications (MONET) 11(3) (2006)
11. Grossglauser, M., Tse, N.C.: Mobility increases the capacity of ad-hoc wireless networks. IEEE/ACM Trans. on Networking 10(4) (2002)
12. Guerin, R.A.: Channel occupancy time distribution in a cellular radio system. IEEE Trans. on Veichular Technology 36(3), 89–99 (1987)
13. Gupta, P., Kumar, P.R.: Critical power for asymptotic connectivity in wireless networks. In: Stochastic Analysis, Control, Optimization and Applications, pp. 547–566 (1998)
14. Kinalis, A., Nikoletseas, S.E.: Adaptive redundancy for data propagation exploiting dynamic sensory mobility. In: Proc. of ACM MSWIM 2008, pp. 149–156 (2008)
15. Kirousis, L.M., Kranakis, E., Krizanc, D., Pelc, A.: Power consumption in packet radio networks. Theoretical Computer Science 243, 289–305 (2000)
16. McDiarmid, C.: On the method of bounded differences. In: Siemons, J. (ed.) London Mathematical Society Lecture Note, vol. 141, pp. 148–188. Cambridge University Press, Cambridge (1989)
17. Mei, A., Stefa, J.: Swim: a simple model to generate small mobile worlds. In: Proc. of IEEE INFOCOM 2009 (2009)
18. Pelusi, L., Passarella, A., Conti, M.: Beyond manets: Dissertation on opportunistic networking. IIT-CNR Tech. Rep. (2006)
19. Penrose, M.: Random Geometric Graphs. Oxford University Press, Oxford (2003)
20. Santi, P., Blough, D.M.: The critical transmitting range for connectivity in sparse wireless ad hoc networks. IEEE Transactions on Mobile Computing 2(1), 25–39 (2003)
21. Zhang, Z.: Routing in intermittently connected mobile ad-hoc networks and delay tolerant networks: overview and challenges. IEEE Communication Surveys 8(1) (2006)
22. Zhao, W., Ammar, M., Zegura, E.: A message ferrying approach for data delivery in sparse mobile ad-hoc networks. In: Proc. of 5th ACM MobiHoc 2004 (2004)

Multiple Random Walks and Interacting Particle Systems

Colin Cooper[1,⋆], Alan Frieze[2,⋆⋆], and Tomasz Radzik[1,⋆⋆⋆]

[1] Department of Computer Science, King's College London, London WC2R 2LS, UK
{colin.cooper,tomasz.radzik}@kcl.ac.uk
[2] Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh, PA 15213, USA
alan@random.math.cmu.edu

Abstract. We study properties of multiple random walks on a graph under various assumptions of interaction between the particles. To give precise results, we make our analysis for random regular graphs. The cover time of a random walk on a random r-regular graph was studied in [6], where it was shown with high probability (**whp**), that for $r \geq 3$ the cover time is asymptotic to $\theta_r n \ln n$, where $\theta_r = (r-1)/(r-2)$. In this paper we prove the following (**whp**) results. For k independent walks on a random regular graph G, the cover time $C_G(k)$ is asymptotic to C_G/k, where C_G is the cover time of a single walk. For most starting positions, the expected number of steps before any of the walks meet is $\theta_r n/\binom{k}{2}$. If the walks can communicate when meeting at a vertex, we show that, for most starting positions, the expected time for k walks to broadcast a single piece of information to each other is asymptotic to $2\theta_r n(\ln k)/k$, as $k, n \to \infty$.

We also establish properties of walks where there are two types of particles, predator and prey, or where particles interact when they meet at a vertex by coalescing, or by annihilating each other. For example, the expected extinction time of k explosive particles (k even) tends to $(2\ln 2)\theta_r n$ as $k \to \infty$.

The case of n coalescing particles, where one particle is initially located at each vertex, corresponds to a voter model defined as follows: Initially each vertex has a distinct opinion, and at each step each vertex changes its opinion to that of a random neighbour. The expected time for a unique opinion to emerge is the expected time for all the particles to coalesce, which is asymptotic to $2\theta_r n$.

Combining results from the predator-prey and multiple random walk models allows us to compare expected detection time in the following cops and robbers scenarios: both the predator and the prey move randomly, the prey moves randomly and the predators stay fixed, the predators move randomly and the prey stays fixed. In all cases, with k predators and ℓ prey the expected detection time is $\theta_r H_\ell n/k$, where H_ℓ is the ℓ-th harmonic number.

[⋆] Research supported in part by Royal Society grant 2006/R2-IJP and EPSRC grant EP/D00232X/1.

[⋆⋆] Research supported in part by NSF grant CCF0502793.

[⋆⋆⋆] Research supported in part by Royal Society grant 2006/R2-IJP.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 399–410, 2009.

1 Introduction

A random walk is a simple process in which particles or messages move randomly from vertex to vertex in a graph. Random walks are an established method of graph exploration and connectivity testing with limited memory. If we consider the case where several random walks occur simultaneously, many questions and different types of application arise: In graph exploration, to what extent do the multiple random walks speed up the process? If the walks can interact how effective is communication, such as broadcasting, between the walks? If there are two different types of particles making walks, then we can model predator-prey processes (cops and robbers). In the case where each vertex of the graph initiates a random walk, there are applications in distributed data collection, gossiping and voting.

In this paper, we study properties of multiple random walks on a graph under various assumptions of interaction between the particles. To give detailed results for comparison purposes, we make the analysis for random regular graphs. The technique used is not specific to random graphs, nor to regular graphs. It can be applied to many graphs with at least reasonable edge expansion, and whose local edge structure around vertices has enough symmetry to be describable in a precise sense.

For brevity we restrict our proofs to random r-regular graphs, $r \geq 3$. Our results also apply to many non-random regular graphs e.g. Lubotsky-Phillips-Sarnak type expanders and, with minor alterations, to many regular graphs where $r \to \infty$ slowly, e.g. the hypercube on $n = 2^r$ vertices. In the case where $r \to \infty$, the parameter θ_r used throughout this paper becomes 1. To make our analysis, we reduce the multiple random walks to a single random walk on a suitably defined product graph, to which we apply the technique of [6]. The main difficulty is to analyze the structure of the product graph, in particular the pair-wise interaction of the walks. Once established, the reduction approach allows us to address a wide range of problems, some of which we now describe.

Suppose there are $k \geq 1$ particles, each making a simple random walk on a graph G. Essentially there are two possibilities, either the particles are oblivious of each other, or can interact on meeting. *Oblivious* particles act independently of each other, with no interaction on meeting. *Interactive* particles, can interact directly in some way on meeting. For example they may exchange information, coalesce, reproduce, destroy each other. We assume that interaction occurs *only when meeting at a vertex*, and that the random walks made by the particles are otherwise independent.

The paper gives precise results for the following topics on random regular graphs G:

1. **Multiple walks.** For k particles walking independently, we establish the cover time $C_G(k)$ of G.
2. **Talkative particles.** For k particles walking independently, which communicate on meeting, we give the expected time to broadcast a message.
3. **Predator-Prey.** For k predator and ℓ prey particles walking independently, we give the expected time to extinction of the prey particles, when predators eat prey particles on meeting.
4. **Annihilating particles.** For $k = 2\ell$ particles walking independently, which destroy each other (pairwise) on meeting, we give the expected time to extinction.

5. **Coalescing particles.** For k particles walking independently, which coalesce on meeting, we give the expected time to coalesce to a single particle. In the case where a walk starts at each vertex, we extend the analysis to a distributed model of voting, the **Voter model**.

The motivation for these models comes from many sources, and we give a brief introduction. A further discussion, with detailed references is given in the appropriate sections below. The formal definitions of the random variables above can be largely found in [2], Chapter 14.

Using random walks to test graph connectivity is an established approach, and it is natural to try to speed this up by parallel searching. Similarly, properties of communication between particles moving in a network, such as broadcasting and gossiping, are natural questions. In this context, the predator-prey model could represent interaction between server and client particles, where each client needs to attach to a server. Combining results from the predator-prey and multiple random walk models allows us to compare expected detection time for the following scenarios: both the predator and the prey move, the prey moves and the predators stay fixed, the predators move and the prey stays fixed. An application of this, is with the predators as cops and the prey as robbers.

Coalescing and annihilating particle systems are part of the classical theory of interacting particles; and our paper makes a new contribution to this area. A system of coalescing particles where initially one particle is located at each vertex, is dual to another classical problem, the voter model, which is defined as follows: Initially each vertex has a distinct opinion, and at each step each vertex changes its opinion to that of a random neighbour. It can be shown that the distribution of time taken for a unique opinion to emerge, is the same as the distribution of time for all the particles to coalesce. By establishing the expected coalescence time, we obtain the expected time to complete voting in the voter model.

Most known results for interacting particle systems are for the infinite d-dimensional grid Z^d (see e.g. Liggett [14]). As far as we know, the results presented here are the first which give precise answers for finite graphs, especially for the Voter model (Theorem 8). For an informative discussion on models of interacting particle systems see Chapter 14 of Aldous and Fill [2].

If one step of a random walk corresponds to a vertex forwarding a message to a random neighbour, and vertices combine messages they receive, the coalescing particle system gives the time taken to combine all messages. Another application is to calculate the average value of a vertex based function $f(v),\ v \in V$; for example temperature. To do this each vertex initiates a message, and the messages then perform a coalescing random walk. The voter model allows the distributed nomination of a central vertex, to e.g. relay messages. This can be used to implement the leader election problem in a distributed network.

Results: Oblivious particles

A standard measure of efficiency of graph exploration by a single random walk, is the cover time, which is defined as follows: Let $G = (V, E)$ be a connected graph, with

$|V| = n$ vertices and $|E| = m$ edges. For a given starting vertex $v \in V$ let C_v be the expected time taken for a simple random walk to visit every vertex of G. The *vertex cover time* C_G is defined as $C_G = \max_{v \in V} C_v$. The (vertex) cover time of connected graphs has been extensively studied. It is a classic result of Aleliunas, Karp, Lipton, Lovász and Rackoff [3] that $C_G \leq 2m(n-1)$. It was shown by Feige [10], [11], that for any connected graph G, the cover time satisfies $(1 - o(1))n \ln n \leq C_G \leq (1 + o(1))\frac{4}{27}n^3$.

For many classes of graphs the cover time can be found precisely. For random regular graphs, the following result was proved in [6].

Theorem 1. *Let $\mathcal{G}_r$ denote the space of r-regular graphs with vertex set $V = \{1, 2, \ldots, n\}$ and the uniform measure. Let $r \geq 3$ be constant, and let $\theta_r = \frac{r-1}{r-2}$. If G is chosen randomly from $\mathcal{G}_r$, then* **whp**

$$C_G \sim \theta_r n \ln n.$$

The results given are asymptotic in n, the size of the vertex set. Thus $A_n \sim B_n$ means that $\lim_{n \to \infty} A_n/B_n = 1$, and **whp** (with high probability) means with probability tending to 1 as $n \to \infty$.

Our first result concerns the speedup in cover time. Let $T(k, v_1, ..., v_k)$ be the time to cover all vertices for k independent walks starting at vertices $v_1, ..., v_k$. Define the k-particle cover time $C_k(G)$ in the natural way as $C_k(G) = \max_{v_1,\ldots,v_k} \mathbf{E}(T(k, v_1, ..., v_k))$ and define the speedup as $S_k = C(G)/C_k(G)$. That the speedup can vary considerably depending on the graph structure can be seen from the following results, which can be easily proved. For the complete graph K_n, the speedup is k; for P_n, the path of length n the speedup is $\Theta(\ln k)$.

Improving s-t connectivity testing by using k independent random walks was studied by Broder, Karlin, Raghavan and Upfal [5]. They proved that for k random walks starting from (positions sampled from) the stationary distribution, the cover time of an m edge graph is $O((m^2 \ln^3 n)/k^2)$. In the case of r-regular graphs, Aldous and Fill [2] (Chapter 6, Proposition 17) give an upper bound on the cover time of $C_k \leq (25 + o(1))n^2 \ln^2 n/k^2$. This bound holds for $k \geq 6 \ln n$.

More recently, the value of $C_k(G)$ was studied by Alon, Avin, Koucký, Kozma, Lotker and Tuttle [4] for general classes of graphs. The paper gives an example, the barbell graph, (two cliques joined by a long path) for which the speed-up is exponential in k provided $k \geq 20 \ln n$.

The paper [4] found that for expanders the speedup was $\Omega(k)$ for $k \leq n$ particles. The class of r-regular graphs we consider are expanders. For these graphs, comparing Theorem 2 with Theorem 1, we see that $C_G(k) \sim C_G/k$, i.e. the asymptotic speedup is *exactly linear*.

Theorem 2 Multiple particles walking independently
Let $r \geq 3$ be constant. Let G be chosen randomly from $\mathcal{G}_r$, then **whp** *and independently of the initial positions of the particles:*

(i) for $k = o(n/\ln^2 n)$ the k-particle cover time $C_G(k)$ satisfies

$$C_G(k) \sim \frac{\theta_r}{k} n \ln n,$$

(ii) for any k, $C_G(k) = O\left(\frac{n}{k}\ln n + \ln n\right)$.

Suppose we distinguish two types of particles, mobile, and fixed; and that mobile particles are predators and the fixed particles are prey (or vice versa). An application of the methods used in Theorem 2 give the following result. For comparison with the case where both predator and prey move, we have included the result of Theorem 5 below, for the predator-prey model. The moral of the story is that as long as at least one particle type moves, the expected detection time is the same.

Theorem 3 Comparison of search models
Let $k,\ \ell \leq n^{\epsilon}$ for a sufficiently small positive constant ϵ.

(i) Suppose there are k mobile predator particles walking randomly, and ℓ prey particles fixed at randomly chosen vertices of the graph. Let $\mathbf{E}(F_{k,\ell,i})$ be the expected detection time of all prey particles.
(ii) Suppose there are ℓ mobile prey particles walking randomly, and k predator particles fixed at randomly chosen vertices of the graph. Let $\mathbf{E}(F_{k,\ell,ii})$ be the expected detection time of all prey particles.

Let $\mathbf{E}(D_{k,\ell})$ be the expected extinction time of ℓ mobile prey using k mobile predators, as given by Theorem 5. Then **whp***, where H_ℓ is the ℓ-th harmonic number,*

$$\mathbf{E}(F_{k,\ell,i}) \sim \mathbf{E}(F_{k,\ell,ii}) \sim \mathbf{E}(D_{k,\ell}) \sim \frac{\theta_r H_\ell}{k} n.$$

Results: Interacting particles

Consider a pair of random walks, starting at vertices u and v. Let $M(u, v)$ be the number of steps before the walks first meet at a vertex. Clearly if $u = v$, then $M(u, v) = 0$. We say the walks are in *general position*, if the starting vertices of the walks are not too near. For our definition of general position $(v_1, v_2, ..., v_k)$, we choose a pairwise separation $d(v_i, v_j) \geq \omega = \omega(k, n)$ between particles, where

$$\omega(k, n) = \Omega(\ln \ln n + \ln k). \tag{1}$$

For the results given in this section, we assume that $r \geq 3$ is constant, that G is chosen randomly from $\mathcal{G}_r$, and that the results hold **whp** over our choice of G.

We first consider problems of passing information between particles. We assume that particles can only communicate when they meet at a vertex. We refer to such particles as *agents*, to distinguish them from non-communicating particles. If initially one agent has a message it wants to pass to all the others, we refer to this process as *broadcasting* (among the agents).

Theorem 4. Broadcast time
Let $k \leq n^{\epsilon}$ for a sufficiently small positive constant ϵ. Suppose k agents make random walks starting in general position. Let B_k be the time taken for a given agent to broadcast to all other agents. Then

$$\mathbf{E}(B_k) \sim \frac{2\theta_r}{k} H_{k-1} n,$$

where H_k is the k-th harmonic number. Thus when $k \to \infty$, $\mathbf{E}(B_k) \sim \frac{2\theta_r \ln k}{k} n$.

An alternative and less efficient way to pass on a message, is for the originating agent to tell it directly to all other agents (by meeting directly with all other agents). Compared to this, broadcasting improves the expected time for everybody to receive the message by a multiplicative factor of $k/2$, for large k. To see this, compare $\mathbf{E}(B_k)$ of Theorem 4, with $\mathbf{E}(D_{1,k-1})$ of Theorem 5 below. Meeting directly with all other agents corresponds to a predator-prey process with one predator (the broadcaster) and $k-1$ prey.

Our next results are for particles which interact in a far from benign manner. One variant of interacting particles is the predator-prey model, in which both types of particles make independent random walks. If a predator encounters prey on a vertex it eats them.

Theorem 5. Predator-prey
Let $k,\ \ell \leq n^\epsilon$ for a sufficiently small positive constant ϵ. Suppose k predator and ℓ prey particles make random walks, starting in general position. Let $D_{k,\ell}$ be the extinction time of the prey. Then

$$\mathbf{E}(D_{k,\ell}) \sim \frac{\theta_r H_\ell}{k} n.$$

A variant of predator-prey is interacting sticky particles, in which all particles are predatorial, and only one particle survives an encounter.

Theorem 6. Coalescence time: sticky particles
Let $k \leq n^\epsilon$ for a sufficiently small positive constant ϵ. Let S_k be the time to coalesce, when there are originally k sticky particles walking randomly, starting from general position. Then,

$$\mathbf{E}(S_k) \ \sim \ 2\theta_r n(k-1)/k,$$

so $\mathbf{E}(S_k) \sim 2\theta_r n,$ *if* $k \to \infty$.

As a twist on predator-prey, we consider "explosive" particles which destroy each other (pairwise) on meeting at a vertex (that is, if two meet, then both are destroyed, but if, say, five meet, then two pairs are destroyed and one particle survives).

Theorem 7. Extinction time: explosive particles.
Let $k \leq n^\epsilon$ for a sufficiently small positive constant ϵ. Suppose there are $k = 2\ell$ explosive particles walking randomly, starting in general position, and that particles destroy each other pairwise on meeting at a vertex. Let D_k be the time to extinction. Then

$$\mathbf{E}(D_k) \ \sim \ 2\theta_r n(H_{2\ell} - H_\ell),$$

so $\mathbf{E}(D_k) \sim 2\theta_r(\ln 2)n,$ *if* $k \to \infty$.

The proofs of Theorems 4-7 are given in Section 5.

Finally we consider the *voter model*. In this model, each vertex initially has a distinct opinion. At each time step, each vertex i contacts a random neighbour j, and changes its opinion to the opinion held by j. The number of opinions is non-increasing at each step. Let C_{vm} be the number of steps needed for a unique opinion to emerge in the voter model and let C_{crw} be the number of steps to complete a coalescing random walk when one particle starts at each vertex. By a duality argument these random variables have the same expected value.

Theorem 8. Voter model whp *for random r-regular graphs,*

$$\mathbf{E}C_{vm} = \mathbf{E}C_{crw} \sim 2\theta_r n.$$

Methodology. For oblivious particles, we use the techniques and results of [6] and [8] to establish the probability that a vertex is unvisited by any of the walks at a given time t. Let T be a suitably large mixing time. Provided the graph is typical (Section 2) and the technical conditions of Lemma 2 are met, then the probability that a vertex v is unvisited at steps $T, ..., t$ tends to $(1 - \pi_v/R_v)^t$. Here π is the stationary distribution and R_v is the number of returns to v during T by a walk starting at v. The value R_v is a property of the structure of the graph around vertex v. For most vertices of a typical graph $R_v \sim \theta_r$, which explains the origin of this quantity.

In [6] a technique, vertex contraction, was used to estimate the probability that the random walk had not visited a given set of vertices. For interacting particles, we use this technique to derive the probability that a walk on a suitably defined product graph H has not visited the diagonal (set of vertices $\boldsymbol{v} = (v_1, ..., v_k)$ with repeated vertex entries v_i) at a given time t. Basically we contract the diagonal to a single vertex, γ, and analyze the walk in the contracted graph Γ.

Proof of Theorems. Because of space restrictions, we only give results and ideas of proofs in this extended abstract. Full proofs of the theorems of this paper are in [9].

2 Typical r-Regular Graphs

We say an r-regular graph G is *typical* if it has the properties **P1-P4** listed below: Let $\epsilon_1 > 0$ be a sufficiently small constant. Let a cycle C be *small* if $|C| \leq L_1$, where

$$L_1 = \lfloor \epsilon_1 \log_r n \rfloor. \tag{2}$$

P1. G is connected, and not bipartite.
P2. The second eigenvalue of the adjacency matrix of G is at most $2\sqrt{r-1} + \epsilon$, where $\epsilon > 0$ is an arbitrarily small constant.
P3. There are at most $n^{2\epsilon_1}$ vertices on small cycles.
P4. No pair of cycles C_1, C_2 with $|C_1|, |C_2| \leq 100L_1$ are within distance $100L_1$ of each other.

The results of this paper are valid for any typical r-regular graph G, and indeed most r-regular graphs have this property.

Theorem 9. *Let $\mathcal{G}_r' \subseteq \mathcal{G}_r$ be the set of typical r-regular graphs. Then $|\mathcal{G}'| \sim |\mathcal{G}_r|$.*

P2 is a deep result of Friedman [13]. The other properties are easy to check. Note that P3 implies that most vertices of a typical r-regular graph are tree-like.

3 Estimating First Visit Probabilities

3.1 Convergence of the Random Walk

Let G be a connected graph with n vertices and m edges. For random walk $\mathcal{W}_u$ starting at a vertex u of G, let $\mathcal{W}_u(t)$ be the vertex reached at step t. Let $P = P(G)$ be the matrix of transition probabilities of the walk and let $P_u^{(t)}(v) = \mathbf{Pr}(\mathcal{W}_u(t) = v)$. Assuming G is not bipartite, the random walk $\mathcal{W}_u$ on G is ergodic with stationary distribution π. Here $\pi(v) = d(v)/(2m)$, where $d(v)$ the degree of vertex v. We often write $\pi(v)$ as π_v.

Let the eigenvalues of $P(G)$ be $\lambda_0 = 1 \geq \lambda_1 \geq \cdots \geq \lambda_{n-1} \geq -1$, and let $\lambda_{\max} = \max(\lambda_1, |\lambda_{n-1}|)$. The rate of convergence of the walk is given by

$$|P_u^{(t)}(x) - \pi_x| \leq (\pi_x/\pi_u)^{1/2} \lambda_{\max}^t. \tag{3}$$

For a proof of this, see for example, Lovasz [15].

In this paper we consider the joint convergence of k independent random walks on a graph $G = (V_G, E_G)$. It is convenient to use the following notation. Let $H_k = (V_H, E_H)$ have vertex set $V_H = V^k$ and edge set $E_H = E^k$. If $S \subseteq V_H$, then $\Gamma(S)$ is obtained from H by contracting S to a single vertex $\gamma(S)$. All edges, including loops are retained. Thus $d_\Gamma(\gamma) = d_H(S)$, where d_F denotes vertex degree in graph F. Moreover Γ and H have the same total degree $(nr)^k$, and the degree of any vertex of Γ, except γ, is r^k.

Let $k \geq 1$ be fixed, and let $H = H_k$. For $F = G, H, \Gamma$ let $\mathcal{W}_{u,F}$ be a random walk starting at $u \in V_F$. Thus $\mathcal{W}_{u,G}$ is a single random walk, and $\mathcal{W}_{u,H}$ corresponds to k independent walks in G.

Lemma 1. *Let G be typical. Let $F = G, H, \Gamma$. Let S be such that $d_H(S) \leq k^2 n^{k-1} r^k$. Let T_F be such that, for graph $F = (V_F, E_F)$, and $t \geq T_F$, the walk $\mathcal{W}_{u,F}$ satisfies*

$$\max_{x \in V_F} |P_u^{(t)}(x) - \pi_x| \leq \frac{1}{n^3},$$

for any $u \in V_F$. Then for $k \leq n$,

$$T_G = O(\ln n),\ T_H = O(\ln n) \textit{ and } T_\Gamma = O(k \ln n).$$

3.2 First Visit Time Lemma: Single Vertex v

Considering a walk $\mathcal{W}_v$, starting at v, let $r_t = \mathbf{Pr}(\mathcal{W}_v(t) = v)$ be the probability that this walk returns to v at step $t = 0, 1, \ldots$. Let

$$R_T(z) = \sum_{j=0}^{T-1} r_j z^j, \tag{4}$$

generate returns during steps $t = 0, 1, \ldots, T_1$. Our definition of return includes $r_0 = 1$.

The following lemma should be viewed in the context that G is an n vertex graph which is part of a sequence of graphs with n growing to infinity. For a proof see [8].

Lemma 2. *Let T be a mixing time such that*

$$\max_{u,x \in V} |P_u^{(t)}(x) - \pi_x| \leq n^{-3}.$$

Let $R_T(z)$ be given by (4), let $R_v = R_T(1)$, and let

$$p_v = \frac{\pi_v}{R_v(1 + O(T\pi_v))}. \tag{5}$$

Suppose the following conditions hold.

(a) *For some constant* $0 < \theta < 1$, *we have* $\min_{|z|\leq 1+\lambda} |R_T(z)| \geq \theta$, *where* $\lambda = \frac{1}{KT}$ *for some sufficiently large constant* K.

(b) $T^2\pi_v = o(1)$ *and* $T\pi_v = \Omega(n^{-2})$.

Let v *be a (possibly contracted) vertex, and for* $t \geq T$, *let* $\boldsymbol{A}_t(v)$ *be the event that* $\mathcal{W}_u$ *does not visit* v *during steps* $T, T+1, \ldots, t$. *Then*

$$\mathbf{Pr}(\boldsymbol{A}_t(v)) = \frac{(1+O(T\pi_v))}{(1+(1+O(T\pi_v))\pi_v/R_v)^t} + o(Te^{-t/KT}).$$

4 Interacting Particles: Applying the First Visit Time Lemma

Recall the definition of H_k consisting of k copies of G, and let $S = \{(v_1, ..., v_k)$: at least two v_i are the same$\}$. The particles making random walks are at the components of the vector corresponding to the vertex in question. Thus S is the set of particle positions in which at least two particles coincide at a given step. As before, let $\gamma(S)$ be the contraction of S to a single vertex, and let $\Gamma(S)$ be H_k with S contracted.

In order to usefully apply Lemma 2, and estimate the first visit probability of γ (and hence S), we need to establish three things.

(i) The value of R_γ, the expected number of returns to the diagonal S of H_k for k particles, and the value of $\pi(\gamma)$, the stationary distribution of γ in Γ.
(ii) The conditions of Lemma 2 hold with respect to the vertex γ of the graph Γ.
(iii) The probability that any particles meet during the mixing time T_Γ.

These points are formally summarized in Lemmas 3-4 below.

Lemma 3. *For typical graphs and* k *particles, the expected number of returns to* γ *in* T_Γ *steps is*

$$R_{\gamma(S)} = \theta_r + O\left(\frac{k^2}{n^{\Omega(1)}}\right). \tag{6}$$

If $k \leq n^\epsilon$ *for a small constant* ϵ, *then* $R_{\gamma(S)} \sim \theta_r$.

Lemma 4. *If* $k \leq n^\epsilon$ *then the conditions of Lemma 2 hold with respect to the vertex* γ *of a typical graph* Γ.

From (5) with $v = \gamma$, and Lemma 3 we have

$$p_\gamma = \frac{\pi_\gamma}{\theta_r(1+O(n^{-\Omega(1)}))}.$$

It follows from [9] that the value of π_γ corresponding to a meeting among k particles is $\pi_\gamma = (1+o(1))\binom{k}{2}/n$, and for a meeting between a given set of s particles and another set of k particles is $\pi_\gamma = (1+o(1))sk/n$. Applying this to Lemma 2 we have the following theorem.

Theorem 10. *Let $\boldsymbol{A}_k(t)$ be the event that a first meeting among the k particles after the mixing time T_Γ, occurs after step t. Let $p_k = \frac{\binom{k}{2}}{\theta_r n}(1 + O(n^{-\Omega(1)}))$. Then*

$$\mathbf{Pr}(\boldsymbol{A}_k(t)) = (1 + o(1))(1 - p_k)^t + O(T_\Gamma e^{-t/2KT_\Gamma}).$$

Let $\boldsymbol{B}_{s,k}(t)$ be the event that a first meeting between a given set of s particles and another set of k particles after the mixing time T_Γ, occurs after step t. Let $q_{sk} = \frac{sk}{\theta_r n}(1 + O(n^{-\Omega(1)}))$. Then

$$\mathbf{Pr}(\boldsymbol{B}_{s,k}(t)) = (1 + o(1))(1 - q_{sk})^t + O(T_\Gamma e^{-t/2KT_\Gamma}).$$

By an *occupied vertex*, we mean a vertex visited by at least one particle at that time step. The next lemma concerns what happens during the first mixing time, when the particles start from general position, and also the separation of the occupied vertices when a meeting occurs.

Lemma 5. *For typical graphs G and $k \leq n^\epsilon$ particles,*
(i) Suppose two (or more) particles meet at time $t > T_\Gamma$. Let p_L be the probability that the minimum separation between some pair of occupied vertices is less than L. Then $p_L = O(k^2 r^L/n)$.
(ii) Suppose the particles start walking on G with minimum separation at least $\alpha(\max\{\ln\ln n, \ln k\})$. Then, for a sufficiently large constant α,

$$\mathbf{Pr}(\textit{Some pair of particles meet during } T_\Gamma) = o(1).$$

From Lemma 5, we see that **whp** particles starting from general position do not meet during the mixing time T_Γ. When some set of particles do coincide after the mixing time, the remaining particles are in general position **whp**.

Corollary 1. *Let M_k (resp. $M_{s,k}$) be the time at which a first meeting of the particles occurs, then $\mathbf{E}(M_k) = (1 + o(1))/p_k$ (resp. $\mathbf{E}(M_{s,k}) = (1 + o(1))/q_{s,k}$).*

This follows from $\mathbf{E}(M_k) = \sum_{t \geq T} \mathbf{Pr}(\boldsymbol{A}_k(t))$ and $p_k T_\Gamma = o(1)$. □

5 Results for Interacting Particles

After an encounter, we allow the remaining particles time $T = T_G$ to re-mix . In any of Theorem 4-7 the total number of particle interactions k^2. Recall that $T_\Gamma = O(kT)$. From Lemma 5, the event that some particles meet during one of these kT_Γ mixing times has probability $O(k^3 T/n^{\Omega(1)}) = o(1)$ (by assumption).

The proof of Theorem 4-7 will now follow from Lemma 5 and Corollary 1.

5.1 Broadcasting, Predator-Prey: Theorems 4, 5

Recall that $D_{k,\ell}$ is the extinction time of the ℓ prey using k predators. Thus

$$\mathbf{E}(D_{k,\ell}) \;=\; O(k\ell T) + \sum_{s=1}^{\ell} \mathbf{E}(M_{s,k}) \;\sim\; n\theta_r \sum_{s=1}^{\ell} \frac{1}{sk} = \frac{n\theta_r}{k} H_\ell,$$

where H_ℓ is the ℓ-th harmonic number. Similarly, the time B_k, for a given agent to broadcast to all other agents is $\sum_{s=1}^{k-1} M_{s,k-s}$, and thus

$$\mathbf{E}(B_k) = O(k\ell T) + n\theta_r \sum_{s=1}^{k-1} \frac{(1+o(1))}{s(k-s)} \sim n\theta_r \sum_{s=1}^{k-1} \frac{1}{s(k-s)} = \frac{2n\theta_r}{k} H_{k-1}.$$

5.2 Expected Time to Coalescence: Theorem 6

Let S_k be the time for all the particles to coalesce, when there are originally k sticky particles walking in the graph. Then,

$$\mathbf{E}(S_k) = O(kT) + \sum_{s=1}^{k} \frac{(1+o(1))}{p_s} \sim n\theta_r \sum_{s=2}^{k} \frac{2}{s(s-1)} = 2\theta_r n \frac{k-1}{k}.$$

We see that for $k \to \infty$, $\mathbf{E}(S_k) \sim 2\theta_r n$.

5.3 Expected Time to Extinction: Explosive Particles: Theorem 7

Let D_k be the time to extinction, when there are originally $k = 2\ell$ explosive particles walking in the graph. Then

$$\mathbf{E}(D_k) = O(kT) + \sum_{s=1}^{\ell} \frac{(1+o(1))}{p_{2s}} \sim n\theta_r \sum_{s=1}^{\ell} \frac{2}{2s(2s-1)} = 2\theta_r n(H_{2\ell} - H_\ell).$$

Noting that $\lim_{\ell\to\infty}(H_{2\ell} - H_\ell) = \ln 2$, we have $\mathbf{E}(D_k) \sim 2\theta_r(\ln 2)\, n$, for $k \to \infty$.

References

1. Aldous, D.: Some inequalities for reversible Markov chains. J. London Math. Soc. 25(2), 564–576 (1982)
2. Aldous, D., Fill, J.: Reversible Markov Chains and Random Walks on Graphs. Monograph in preparation, http://stat-www.berkeley.edu/pub/users/aldous/RWG/book.html
3. Aleliunas, R., Karp, R.M., Lipton, R.J., Lovász, L., Rackoff, C.: Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems. In: Proceedings of the 20th Annual IEEE Symposium on Foundations of Computer Science, pp. 218–223 (1979)
4. Alon, N., Avin, C., Kouchý, M., Kozma, G., Lotker, Z., Tuttle, M.: Many random walks are faster then one. In: Proceedings of the 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures, pp. 119–128 (2008)
5. Broder, A., Karlin, A., Raghavan, A., Upfal, E.: Trading space for time in undirected s-t connectivity. In: Proceedings of the 21st Annual ACM Symposium on Theory of Computing, pp. 543–549 (1989)
6. Cooper, C., Frieze, A.M.: The cover time of random regular graphs. SIAM Journal on Discrete Mathematics 18, 728–740 (2005)

7. Cooper, C., Frieze, A.M.: The cover time of the preferential attachment graph. Journal of Combinatorial Theory Series B 97(2), 269–290 (2007)
8. Cooper, C., Frieze, A.M.: The cover time of the giant component of a random graph. Random Structures and Algorithms 32, 401–439 (2008)
9. Cooper, C., Frieze, A.M., Radzik, T.: Multiple random walks in random regular graphs (2008), http://www.math.cmu.edu/~af1p/Texfiles/Multiple.pdf
10. Feige, U.: A tight upper bound for the cover time of random walks on graphs. Random Structures and Algorithms 6, 51–54 (1995)
11. Feige, U.: A tight lower bound for the cover time of random walks on graphs. Random Structures and Algorithms 6, 433–438 (1995)
12. Feller, W.: An Introduction to Probability Theory, 2nd edn., vol. I. Wiley, Chichester (1960)
13. Friedman, J.: A proof of Alon's second eignevalue conjecture. Memoirs of the A.M.S. (to appear)
14. Liggett, T.M.: Interacting Particle Systems. Springer, Heidelberg (1985)
15. Lovász, L.: Random walks on graphs: a survey. In: Bolyai Society Mathematical Studies, Combinatorics, Paul Erdös is Eighty, Keszthely, Hungary, vol. 2, pp. 1–46 (1993)
16. Sinclair, A.: Improved bounds for mixing rates of Markov chains and multicommodity flow. Combinatorics, Probability and Computing 1, 351–370 (1992)

Derandomizing Random Walks in Undirected Graphs Using Locally Fair Exploration Strategies[⋆]

Colin Cooper[1], David Ilcinkas[2], Ralf Klasing[2], and Adrian Kosowski[2,3]

[1] Dept of Computer Science, King's College London
colin.cooper@kcl.ac.uk
[2] LaBRI, CNRS and Université de Bordeaux
{ilcinkas,klasing}@labri.fr
[3] Dept of Algorithms and System Modeling, Gdańsk University of Technology
adrian@kaims.pl

Abstract. We consider the problem of exploring an anonymous undirected graph using an oblivious robot. The studied exploration strategies are designed so that the next edge in the robot's walk is chosen using only local information, and so that some local equity (fairness) criterion is satisfied for the adjacent undirected edges. Such strategies can be seen as an attempt to derandomize random walks, and are natural undirected counterparts of the rotor-router model for symmetric directed graphs.

The first of the studied strategies, known as Oldest-First (OF), always chooses the neighboring edge for which the most time has elapsed since its last traversal. Unlike in the case of symmetric directed graphs, we show that such a strategy in some cases leads to exponential cover time. We then consider another strategy called Least-Used-First (LUF) which always uses adjacent edges which have been traversed the smallest number of times. We show that any Least-Used-First exploration covers a graph $G = (V, E)$ of diameter D within time $O(D\,|E|)$, and in the long run traverses all edges of G with the same frequency.

1 Introduction

A widely studied problem concerns the exploration of an anonymous graph $G = (V, E)$, with the goal of visiting all its vertices and regularly traversing its edges. At each discrete moment of time, the robot is located at a node of the graph, and is provided with only a local view of the adjacent edges of the graph. The exploration strategies studied in this paper fall into the line of research devoted to derandomizing random walks in graphs [4,7,19,20,22].

The *random walk* is an oblivious exploration strategy in which the edge used by the robot to exit its current location is chosen with equal probability from

[⋆] The research was partially funded by the KBN (Poland) Grant 4 T11C 047 25, by the ANR projects "ALADDIN" and "ALPAGE", by the INRIA project "CEPAGE", and by European COST Action Action 295 "Dynamic Communication Networks" (DYNAMO).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 411–422, 2009.

among all the edges adjacent to the current node; cf. e.g. [1,16] for an extensive introduction to the topic. Explorations achieved through random walks are on average good, in the sense that the following properties hold *in expectation*:

(1) Within polynomial time, the walk visits all of the vertices of the graph.
(2) Within polynomial time, the walk stabilizes to the steady state, and henceforth all edges are visited with the same frequency.

We focus on the problem of designing local exploration strategies which derandomize a random walk in a graph in an attempt to achieve the above stated properties in the deterministic sense of *worst-case performance*. The next vertex to be visited should depend only on the values of certain parameters associated with the edges adjacent to the current node. Such a problem naturally gives rise to the definition of *locally equitable strategies*, i.e. strategies, in which at each step the robot chooses from among the adjacent edges the edge which is in some sense the "poorest", in an effort to make the traversal fair. In this context, two natural notions of equity may be defined:

– An exploration is said to follow the *Oldest-First* (OF) strategy if it directs the robot to an unexplored neighboring edge, if one exists, and otherwise to the neighboring edge for which the most time has elapsed since its last traversal, i.e. the edge which has waited the longest.
– An exploration is said to follow the *Least-Used-First* (LUF) strategy if it directs the robot to a neighboring edge which has so far been visited by the robot the smallest number of times.

When the considered graph is *symmetric and directed*, and the above definitions are applied to directed edges, then the Oldest-First notion of equity is known to be strictly stronger than Least-Used-First, i.e. any exploration which follows the OF strategy also follows the LUF strategy [22]. Moreover, the Oldest-First strategy is in this context equivalent to a well-established efficient exploration model based on the rotor-router model (a.k.a. the "Propp machine", cf. e.g. [5] for an introduction of the model). In the directed case, both of the described locally fair exploration stratagies are known to preserve properties (1) and (2) of the random walk. More precisely, for a symmetric directed graph of diameter D, any exploration which follows such a strategy achieves a *cover time* of $O(D\,|E|)$ and stabilizes to a globally fair traversal of all the edges. Herein we look at the Oldest-First and Least-Used-First strategies when applied to the *undirected* edges of a graph. For this case, the results, and the used techniques, turn out to be surprisingly different.

Basic parameters. Two parameters of interest when discussing exploration strategies are the *cover time* of a graph and the *traversal frequency* of its edges. We introduce them first in the context of random walks.

Let $\mathbf{C}_s$ be the random variable describing the number of steps required for a random walk starting at vertex s, to visit every vertex of the graph. Then the *cover time* of the graph is the maximum, taken over all starting vertices s, of the expected values of variables $\mathbf{C}_s$, $\mathcal{C}(G) = \max_{s \in V} \mathbf{E}\, \mathbf{C}_s$. Let $\mathbf{c}_{s,e}(t)$ be the random variable describing the number of visits to edge e within time t, for a random walk starting at vertex s. We can define random variables describing the distribution of visits to edges for sufficiently large time, $\mathbf{f}_{s,e} = \liminf_{t \to \infty} \mathbf{c}_{s,e}(t)/t$ (where $\liminf$ is used instead of $\lim$ to guarantee correctness of the definition). The *traversal frequency* $f_e(G)$ of an edge e is defined as the minimum, taken over all starting vertices s, of the expected values of variables $\mathbf{f}_{s,e}$, $f_e(G) = \min_{s \in V} \mathbf{E}\, \mathbf{f}_{s,e}$.

Given any exploration algorithm $\mathcal{E}$ which is fully deterministic (or in other words, a specific exploration), the notions of *cover time for* $\mathcal{E}$ and *traversal frequency for* $\mathcal{E}$ can be defined analogously. The only difference is that then the variables $\mathbf{C}_s$ and $\mathbf{f}_{s,e}$ are deterministically defined, hence we need not speak of their expected values.

Related work. We confine ourselves to a short survey of works on random walks, and the rotor-router model and its variants. Many other approaches to the derandomization of random walks have been studied, most notably, through universal traversal sequences [2] (UTS) and universal exploration sequences [15] (UXS). UTS-s can be constructed in polylogarithimic space using pseudorandom generators, cf. e.g. [18], whereas UXS-s have been proved to be constructible in log-space [17].

Exploration with random walks. In expectation, random walks quickly "hit" all vertices, and the cover time $\mathcal{C}(G)$ of a connected graph satisfies the inequalities $\mathcal{C}(G) \geq |V| \log |V|$ and $\mathcal{C}(G) = O(|V|^3)$ [2]. With respect to the diameter, the cover time is upper bounded by $O(D\,|E| \log |V|)$. In fact, for many special graph classes, such as complete graphs, expanders, trees, or grids, tighter bounds on cover time can be obtained [1].

Random walks directly capture the property of equity in the sense that, for a random walk in the steady state, the expected frequency of visits to each edge is the same. More precisely, for a random walk on a connected undirected non-bipartite graph G, the stationary distribution of visits to edges is the uniform distribution with parameter $1/|E|$, thus for any e, $f_e(G) = 1/|E|$. Similarly, if we replace each edge $\{u, v\}$ with two symmetric directed edges (u, v), (v, u) then the stationary distribution of visits is again uniform with parameter $1/(2|E|)$, and so for any directed edge $\boldsymbol{e}$, $f_{\boldsymbol{e}}(G) = 1/(2|E|)$.

In expectation, the random walk stabilizes to such a fair traversal of the edges very quickly. Several notions have been introduced, informally corresponding to the expected moment at which (for a regular graph) all vertices have been visited a similar number of times, cf. [21]. One of the most studied is that of blanket time, which has been shown to be within a factor of $O(\log \log |V|)$ of the cover time, for all graphs [12].

Equitable exploration of directed graphs. For symmetric directed graphs, the Oldest-First exploration strategy corresponds to exploration in the rotor-router model, i.e. a set-up in which edges exiting each node have successive labels, and the next edge to be traversed is selected by a pointer. After this edge is traversed, the pointer moves on to the edge with the next label, in a cyclic way.

This approach was first studied in [4,19,20], and the cover time of Oldest-First for directed graphs was shown to be $O(|V||E|)$. Slightly later [22] obtained an improved bound on cover time of $O(D\,|E|)$, and also showed that after time at most $O(D\,|E|)$ the exploration stabilizes to a periodic traversal of some directed Eulerian cycle of the graph (containing each directed edge exactly once, i.e. of length $2|E|$). Consequently, Oldest-First explorations on symmetric directed graphs are fair, in the sense that all edges are visited with the same frequency $f_{\boldsymbol{e}}(G) = 1/(2|E|)$.

When considering symmetric directed graphs, an exploration achieved in accordance with the Oldest-First rule also satisfies the conditions of a Least-Used-First exploration. Whereas a Least-Used-First exploration need not in general stabilize to a traversal of a directed Eulerian cycle, it also retains the property that for any time moment, the number of visits to any two edges outgoing from the same vertex can differ by at most 1 [14, 13]. This property immediately implies that for symmetric directed graphs, any execution of Least-Used-First has a cover time of $O(D\,|E|)$, and also visits all directed edges with the same frequency.

In a slightly wider context, local exploration strategies have been considered for robots with bounded memory, cf. e.g. [8,9,17]. In some settings, the robot is additionally assisted by identifiers or markers placed on the nodes and/or edges of the explored graph, cf. e.g. [3,6,10].

Our results. Herein we establish certain properties of explorations which follow the Oldest-First or Least-Used-First strategies in undirected graphs.

*The Oldest-First (*OF*) strategy* in undirected graphs can be regarded as a natural analogue of the Oldest-First strategy (rotor-router model) for symmetric directed graphs. However, whereas the rotor-router model leads to explorations which traverse directed edges with equal frequency, and have a cover time bounded by $O(D\,|E|)$, this is not the case for Oldest-First explorations in undirected graphs. Indeed, in Section 2 we show the following theorems.

- In some classes of undirected graphs, any exploration which follows the Oldest-First strategy is unfair, with an exponentially large ratio of visits between the most often and least often visited edges (Theorem 1).
- There exist explorations following the Oldest-First strategy which have exponential cover time of $2^{\Omega(|V|)}$ in some graph classes (Theorem 2).

*The Least-Used-First (*LUF*) strategy* in undirected graphs is fundamentally better than the Oldest-First strategy, which is contrary to the situation in symmetric directed graphs. In fact, in Section 3 we show that, in undirected graphs, explorations which follow the LUF strategy are fair, efficient, and tolerant to perturbations of initial conditions, as expressed by the following theorems.

- Any exploration of an undirected graph which follows the Least-Used-First strategy is fair, achieving uniform distribution of visits to all edges (Theorem 5).

- Any exploration of an undirected graph which follows the Least-Used-First strategy achieves a cover time of $O(D\,|E|)$, where D denotes the diameter (Theorem 4). This bound is tight (Theorem 3). When the exploration starts from a state with non-zero (corrupted) initial values of traversal counts on edges, the cover time is bounded by $O((|V|+p)|E|)$, where p is the maximal value of a counter in the initial state (Theorem 6).

Notation. Unless otherwise stated, all considered graphs are assumed to be simple, undirected, and connected. The explored graph is denoted by $G=(V,E)$, with $|V|=n$ and $|E|=m$. The diameter of the graph is denoted by D and its maximum vertex degree by Δ. The set of neighbors of a vertex $v \in V$ is denoted by N_v. The set of non-negative integers is denoted by $\mathbb{N}$. A discrete interval $[a,b]$ is defined as the set of all integers k such that $a \le k \le b$ ($[a,b]=\emptyset$ when $a>b$).

2 The Oldest-First (OF) Strategy

In this section we show that any OF exploration is unfair (Theorem 1), and moreover that OF explorations may sometimes take exponential time to cover the whole graph (Theorem 2).

Theorem 1. *There exists a family of graphs $(G_n)_{n\ge1}$ of order $\Theta(n)$, such that for each graph G_n in this family, some two of its edges e and e' satisfy $\frac{f_e(G_n)}{f_{e'}(G_n)} = (\frac{3}{2})^n$ with $f_{e'}(G_n) \ne 0$, for any exploration following the* OF *strategy.*

Proof. Fix an arbitrary positive integer n. Let G_n be the graph defined as follows. The nodes are denoted $v_j^{(k)}$, for any $j \in [1,7]$ and any $k \in [1,n]$. Moreover, we have that $v_7^{(k)} = v_1^{(k+1)}$ for any $k \in [1,n-1]$. This means that G_n has $6n+1$ nodes. The $8n$ edges are the following: $e_1^{(k)} = \{v_1^{(k)}, v_2^{(k)}\}$, $e_2^{(k)} = \{v_2^{(k)}, v_3^{(k)}\}$, $e_3^{(k)} = \{v_2^{(k)}, v_4^{(k)}\}$, $e_4^{(k)} = \{v_3^{(k)}, v_5^{(k)}\}$, $e_5^{(k)} = \{v_4^{(k)}, v_5^{(k)}\}$, $e_6^{(k)} = \{v_2^{(k)}, v_6^{(k)}\}$, $e_7^{(k)} = \{v_5^{(k)}, v_6^{(k)}\}$, and $e_8^{(k)} = \{v_6^{(k)}, v_7^{(k)}\}$, for any $k \in [1,n]$. The graph G_n is depicted in Figure 1.

We assume that the exploration is starting from $v_1^{(1)}$. We will now focus on a block B of G_n, that is on the subgraph of G_n induced by the 7 nodes $\{v_1^{(k)}, \cdots, v_7^{(k)}\}$, for an arbitrary and fixed $k \in [1,n]$. To simplify the notation, we will remove the superscript ${}^{(k)}$ in the following, when there are no ambiguities.

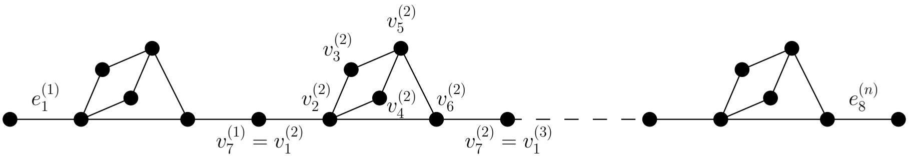


Fig. 1. The graph G_n

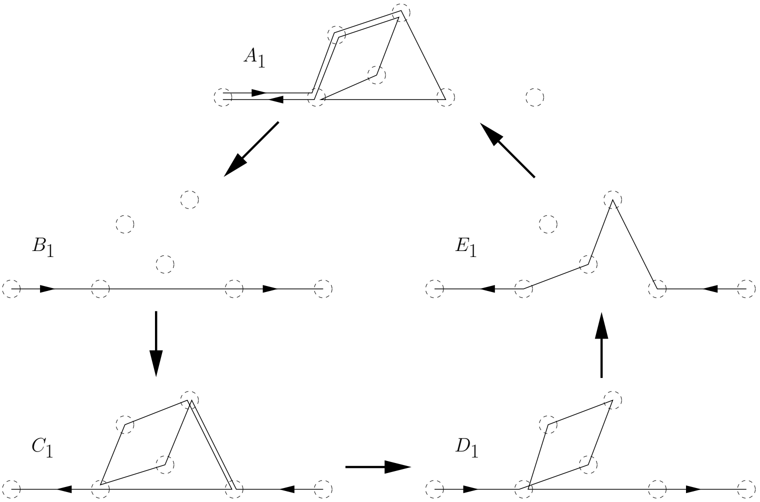


Fig. 2. The two possible cycles of traversals of a block B. Cycle $(A_1, B_1, C_1, D_1, E_1)$ is presented in the figure. Cycle $(A_2, B_2, C_2, D_2, E_2)$ is obtained as follows: $A_2 = \bar{A}_1$, $B_2 = \bar{E}_1$, $C_2 = \bar{D}_1$, $D_2 = \bar{C}_1$, $E_2 = \bar{B}_1$, where $\bar{X}$ denotes the reversal of the direction of the exploration route in X.

There may be several different explorations following the OF strategy from $v_1^{(1)}$. Indeed, when the exploration reaches a node with at least two edges that are not yet explored, the exploration may proceed along any of these unexplored edges.

By a tedious case-by-case analysis, we show that the behavior of the robot in successive traversals of a given block follows a cyclic pattern, as shown in Figure 2. In all the cases, in the time period during which the edge e_8 is traversed 4 times, the edge e_1 is traversed 6 times. We now notice that the exploration becomes eventually periodic. Indeed, only the local ordering of the last traversal times of the incident edges at each node influences the exploration. Therefore the number of different possible configurations of the graph and its ongoing exploration is bounded by some (large) function of n. Therefore, the exploration is eventually periodic and, for any edge e of the graph, the sequence $c_e(t)/t$ converges to the actual frequency of traversals $f_e(G_n)$ of the edge e. In particular, we have $\sum_{e \in E(G_n)} f_e(G_n) = 1$. Since we just proved that for any $k \in [1, n]$ we have $f_{e_1^{(k)}}(G_n) = \frac{3}{2} f_{e_8^{(k)}}(G_n)$, we have $f_{e_1^{(1)}}(G_n) = (\frac{3}{2})^n f_{e_8^{(n)}}(G_n)$, with $f_{e_8^{(n)}}(G_n) \neq 0$. This concludes the proof of the theorem. □

Theorem 2. *There exists a family of graphs $(G_n)_{n \geq 1}$ of order $\Theta(n)$, such that for each graph G_n in this family, some exploration following the* OF *strategy has a cover time of* $2^{\Omega(n)}$.

Proof. We consider the family of graphs described in Theorem 1. Given an arbitrary execution $\mathcal{E}$ of the OF strategy, there exist two edges e and e' satisfying

$\frac{f_e(G)}{f_{e'}(G)} = (\frac{3}{2})^n$ (with $f_{e'}(G) \neq 0$). Therefore, there exist two times t_1 and t_2, with $t_2 - t_1 \geq (\frac{3}{2})^n - 1$, such that the edge e' is not traversed between time t_1 and t_2. Let v be the current position of the traversal $\mathcal{E}$ at time t_1. Then, consider the exploration $\mathcal{E}'$ which starts at v and has the same execution from the beginning, as $\mathcal{E}$ from time t_1. It is clear that $\mathcal{E}'$ follows the OF strategy, and moreover it will not traverse e' before time $t_2 - t_1$. Thus, $\mathcal{E}'$ has a cover time of at least $(\frac{3}{2})^n - 1$. □

3 The Least-Used-First (LUF) Strategy

In Subsection 3.2 we will show that LUF strategies are fair and cover any graph in $O(m\,D)$ time. Before doing this, in Subsection 3.1 we construct a family of examples showing that such a bound on cover time is essentially tight.

3.1 A Worst Case Lower Bound on Cover Time

Theorem 3. *For sufficiently large n, $m \in [n-1, n(n-1)/2]$ and $D \leq n$, the worst-case cover time of the* LUF *strategy in the family of graphs of at most n nodes, at most m edges, and diameter at most D, is $\Omega(m\,D)$.*

Proof. Fix $n \geq 16$, $m \in [n-1, n(n-1)/2]$ and $D \in [8, n]$. Let G be the graph defined as follows, see Figure 3. Let $n_C = \lfloor D/8 \rfloor$. The graph G first consists of $3n_C + 1$ nodes organized in a chain of 4-node cycles. Let n_K be the largest even integer smaller than $n/2$ such that $n_K(n_K+1)/2 < m/2$. The graph G also consists of n_K additional nodes forming together with one extreme node of the chain a complete graph on n_K+1 vertices. To summarize, G has $n_K + 3n_C + 1 \leq n$ nodes, $4n_C + n_K(n_K+1)/2 \leq m$ edges and diameter $2n_C + 1 \leq D$.

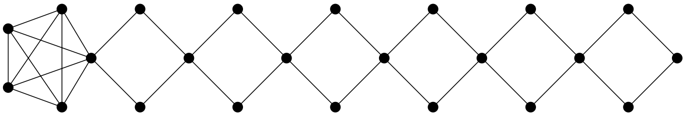


Fig. 3. The graph G with $n_C = 6$ and $n_K = 4$

It is easily shown that the worst-case cover time of G is at least $n_K(n_K + 1)/2 \cdot n_C$; we leave out the details of the analysis. Since $n_K(n_K+1)/2 \in \Omega(m)$ and $n_C \in \Omega(D)$, the theorem holds. □

3.2 An Upper Bound on Cover Time

We now proceed to prove the $O(m\,D)$ bound on cover time of any LUF exploration, through a sequence of technical lemmas.

Throughout the proofs we will use the following notation. When describing moments of time, the symbol t' is treated as a more compact notation for $t+1$, likewise t'' means $t+2$. The vertex occupied by the robot at time t is denoted by $r(t)$; the starting vertex of exploration is denoted by s, that is $s = r(0)$. With each edge e we associate a counter c_e called its *traversal count*, whose value at time t is denoted by $c_e(t)$; initially we assume $c_e(0) = 0$ for all $e \in E$. When traversing edge e in the time interval (t, t') we only increment the value of the counter associated with this edge, $c_e(t') = c_e(t) + 1$. For each node u we denote by $C_u(t)$ the set of traversal counts of the adjacent edges at time t: $C_u(t) = \{c_{\{u,v\}}(t) : v \in N_u\}$. The set of traversal counts of all edges of the graph is denoted by $C(t) = \{c_e(t) : e \in E\}$.

At any given time t, let parameter $k \in \mathbb{N} \cup \{-1\}$ be defined in such a way that $\max C(t) \in [2k+1, 2k+2]$, and let parameter $l \in \mathbb{N}$ be such that $\min C_r(t) \in [2l, 2l+1]$. Parameters k, l, and r used without an indication of time are assumed to refer to the moment of time denoted by t, while symbols k', l', r', and r'' should be treated as equivalent to $k(t')$, $l(t')$, $r(t')$, and $r(t'')$, respectively.

We start by making the following claim which is a simple extension of the following observation: for each vertex v different from both r and s, the total number of traversals of edges incident to v, performed when entering v, is the same as the total number of traversals of these edges performed when leaving v.

Lemma 1. *For a node $u \in V$, let $S_u(t) = \sum_{v \in N_u} c_{\{u,v\}}(t)$. If $S_u(t)$ is odd, then $r \neq s$ and either $u = r$, or $u = s$.*

Lemma 2. *If for some time moment t we have $k' = k+1$, then $r = s$ and $C_s(t) = \{2k+2\}$.*

Proof. If $k' = k+1$, then clearly $c_{\{r,r'\}}(t) = 2k+2 = \max C(t)$. This implies that during the time interval (t, t') the robot chooses an edge having the maximal traversal count. Clearly, this means that there is no edge with a smaller traversal count available at r, so $C_r(t) = \{2k+2\}$. Hence, in Lemma 1 the value of $S_r(t)$ is even, and we immediately obtain the claim, $r = s$. □

Lemma 3. *For any time moment t, $\max C_s(t) \geq 2k+1$.*

Proof. We can obviously assume that $k \geq 0$. Let $\tau < t$ be such a time moment that $k(\tau) = k-1$ and $k(\tau') = k$. Then by Lemma 2, $r(\tau) = s$ and $C_s(\tau) = \{2(k-1)+2\} = \{2k\}$. So, after traversing any edge adjacent to s, we obtain $\max C_s(\tau') = 2k+1$. Since $t \geq \tau'$ and $\max C_s(t) \geq \max C_s(\tau')$, the claim follows directly. □

Lemma 4. *If for some time moment t we have $C_r(t) = \{2p+2\}$, where p is some integer, then $r = s$ and $p = k$.*

Proof. When $C_r(t) = \{2p+2\}$, in Lemma 1 the value of $S_r(t)$ is even, and so $r = s$. Moreover, by Lemma 3 we cannot have $p < k$ since then $\max C_s(t) \leq 2k$. Thus $p = k$. □

Lemma 5. *For any time moment t, the following statements hold:*

- *there exists a subset $V_A = \{v_l, ..., v_{k-1}\}$ of vertices indexed by integers $a \in [l, k-1]$, such that $C_{v_a}(t) \subseteq [2a, 2a+3]$.*
- *for any other vertex $v \notin V_A$ we have $C_v(t) \subseteq [2b, 2b+2]$ for some integer value b.*

Proof. Initially, for $t = 0$ we have $C(0) = \{0\}$, $k = -1$, $l = 0$, and so the induction claim holds with $V_A(0) = \emptyset$.

Assuming that the induction assumption holds for time t and a corresponding set V_A is given, we will now prove that it also holds for time t' with an appropriately modified set V'_A. (Sometimes no modification will be necessary; for example, when G is a cycle, we have $V_A(\tau) = \emptyset$ for all $\tau \geq 0$.) We start by showing a small auxiliary claim.

Claim. $l' \in [l-1, l+1]$.

Proof: The traversal count, directly before traversal, of the edge used in time interval (t', t'') can be greater by at most one than that of the edge used in time interval (t, t'), so $c_{\{r', r''\}}(t') \leq c_{\{r, r'\}}(t) + 1 \leq 2l + 2$, and thus $l' \leq l + 1$. Suppose that $l' < l$; then we have $\min C_{r'}(t) < 2l$, and by the inductive assumption $r' \notin V_A(t)$. Thus $\max C_{r'}(t) - \min C_{r'}(t) \leq 2$, and we obtain $c_{\{r', r''\}}(t') = \min C_{r'}(t') \geq \min C_{r'}(t) \geq \max C_{r'}(t) - 2 \geq 2l - 2$, which means that always $l' \geq l - 1$, completing the proof of the claim.

Now, consider the following definition of set $V'_A = \{v'_a : a \in [l', k'-1]\}$ for time t': (1) For all $a \in [l+1, k-1]$, put $v'_a := v_a$; (2) If $l' \leq l$ and $l' < k'$, put $v'_l := v_l$; (3) If $l' = l - 1$ and $l' < k'$, put $v'_{l-1} := r'$. The above procedure clearly defines all elements v'_a for $a \in [l', k-1]$. We now observe that it does in fact define all elements v'_a for the whole of the required range, $a \in [l', k'-1]$. Indeed, if $k' = k + 1$, by Lemma 2 we have $c_{\{r, r'\}}(t) = 2k + 2$, so $l = k + 1 = k'$. Consequently, if $l' \geq k + 1$ in the proposed construction, then set V'_A is empty as required, and if $l' = l - 1 = k$, then the only element v'_{l-1} of V'_A is well defined.

We now verify the induction claim for the proposed definition of set V'_A by checking the imposed bounds on sets $C_v(t')$, for all vertices $v \in V$. Taking into account that for all vertices v other than r and r' we have $C_v(t) = C_v(t')$, by the construction of elements v'_a based on elements v_a, it is evident that it now suffices to check the bounds on $C_v(t')$ for $v \in \{r, r', v_l\}$; for all other vertices, the bounds follow directly from the induction assumption for time t. We therefore now successively consider vertices r, r', and v_l.

For vertex r we need to consider two possibilities: either $r \in V_A$, or $r \notin V_A$.

1. If $r \in V_A$, then $V_A \neq \emptyset$ and so $l \leq k - 1$. Since $\min C_r(t) \in [2l, 2l+1]$ by the definition of l, taking into account the inductive assumption concerning the bounds on $C_r(t)$ we must have $r = v_l$ (note that vertices v_a are only defined for indices $a \geq l$) and $C_r(t) \subseteq [2l, 2l+3]$. After traversing edge $\{r, r'\}$, we have $c_{\{r, r'\}}(t') = c_{\{r, r'\}}(t) + 1 = \min C_r(t) + 1 \in [2l+1, 2l+2]$, so we retain the property $C_r(t') \subseteq [2l, 2l+3]$. If $r = v'_l$, the bounds on set $C_r(t')$ are thus satisfied. We will now show that the other case, $r \neq v'_l$, is impossible. Indeed, when $r \neq v'_l$ we would have $l' = l + 1$ (otherwise, $l' \leq l$ would mean

that $l' \leq l < k \leq k'$, so $v'_l = v_l = r$). Therefore, $\min C_{r'}(t') \geq 2l+2$, so $c_{\{r,r'\}}(t') = 2l+2$ and $c_{\{r,r'\}}(t) = 2l+1$. Taking into account that $r' \neq r = v_l$, we have $r' \notin V_A$ (as $\min C_{r'}(t) \leq 2l+1$) and $C_{r'}(t) \subseteq [2l, 2l+2]$. As we have already observed that $\min C_{r'}(t') \geq 2l+2$ and $c_{\{r,r'\}}(t') = 2l+2$, we obtain $C_{r'}(t') = \{2l+2\}$. Applying Lemma 4 for time t' gives $r' = s$ and $k = l$, a contradiction with the assumption $l < k$.

2. If $r \notin V_A$, then since $c_{\{r,r'\}}(t) \in [2l, 2l+1]$, we must have $C_r(t) \subseteq [2l, 2l+2]$ (note that we must have $\min C_r(t) \geq 2l$). At time t', only the traversal count of edge $\{r, r'\}$ changes, $c_{\{r,r'\}}(t') = c_{\{r,r'\}}(t) + 1 = \min C_r(t) + 1 \in [2l+1, 2l+2]$, and we still have $C_r(t') \subseteq [2l, 2l+2]$. By the definition of set V'_A we have $r \notin V'_A$, so $C_r(t')$ fulfills the required bound with parameter $b = l$.

For vertex r' we likewise consider two possibilities: either $r' \in V_A$, or $r' \notin V_A$; in both cases, we obtain that the required bounds on $C_{r'}(t')$ are satisfied.

Finally, we consider vertex v_l (under the assumption that $l < k$, otherwise this case should be left out). Since the bounds for vertices r and r' have already been proven, we can restrict ourselves to the case of $v_l \neq r$ and $v_l \neq r'$. This means that the set of traversal counts adjacent to v_l does not change during the time interval (t, t'), i.e. $C_{v_l}(t) = C_{v_l}(t')$. Clearly, the only situation which needs some comment is when $v_l \notin V'_A$; we will show that such a case is not possible. Indeed, this would mean that $l' = l+1$ or $l' \geq k'$. If $l' = l+1$, then we would have $c_{\{r,r'\}}(t') = 2l+2$, so $c_{\{r,r'\}}(t) = 2l+1$, and since $r' \neq v_l$, we see from the inductive assumption that $r' \notin V_A$ and $C_{r'}(t) \subseteq [2l, 2l+2]$. Hence, noting that $l' = l+1$, we have $C_{r'}(t') = \{2l+2\}$, and by applying Lemma 4 for time t' we obtain $r' = s$ and $k = l$, a contradiction with the assumption $l < k$. Finally, we need to consider the case $l' \geq k'$. Then, since $k' \geq k$ and $l' \leq l+1$, we obtain $l' = k' = k = l+1$, which turns out to be a subcase of the previously considered case $l' = l+1$. □

Theorem 4. *For any graph, the cover time achieved by any* LUF *exploration is at most* $2m(D+1)$.

Proof. Consider any time moment t such that $l \geq k$. Then by Lemma 5 set V_A is empty, and for any vertex $v \in V$ we have $\max C_v(t) - \min C_v(t) \leq 2$. Let edge $\{v_a, v_b\}$ be such that $c_{\{v_a,v_b\}} \geq 2k+1$, and consider any other edge $\{u_a, u_b\}$ of the graph. Let us arbitrarily choose a shortest path $(w_1, w_2, \ldots, w_d)$, with $w_1 = u_a$ and $w_d = v_a$; obviously, $d \leq D+1$. The following relations hold: $c_{\{u_a,u_b\}}(t) \geq \min C_{w_1}(t) \geq \max C_{w_1}(t) - 2 \geq c_{\{w_1,w_2\}}(t) - 2 \geq \min C_{w_2}(t) - 2 \geq \max C_{w_2}(t) - 4 \geq \ldots \geq \max C_{w_d}(t) - 2d \geq c_{\{v_a,v_b\}} - 2d \geq 2k+1-2(D+1) = 2(k-D)-1$. So, at any time moment t such that $l \geq k > D$, each edge of the graph has been explored at least once. Notice that this is always true for the unique time moment t such that $\max C(t) = 2D+2$ and $\max C(t') = 2D+3$, and we will use this time moment t as an upper bound on cover time. Since at time $\tau = (2D+2)m+1$ we must have $\max C(\tau) > 2D+2$ by the pigeon-hole principle, we immediately obtain that $t < \tau$, and the claim follows. □

Taking into account that by Lemma 5, for any time moment t and for any vertex $v \in V$, we have $\max C_v(t) - \min C_v(t) \leq 3$, and using similar arguments as in the above proof, we obtain that at any moment of time t the following inequalities hold: $\max C(t) - \min C(t) \leq 3(D+1)$. We easily conclude that in the limit, all edges are explored with the same frequency.

Theorem 5. *For any graph, any exploration following the* LUF *strategy achieves uniform frequency on all edges, $f_e(G) = 1/m$.*

3.3 Cover Time of LUF with Modified Initial Conditions

It turns out that LUF explorations are resistant to minor perturbations, for example when the initial values of traversal count are not necessarily 0 for all edges e, but arbitrarily drawn from some range of values. We have the following theorems; details of the proofs are omitted.

Theorem 6. *For any graph, the cover time achieved by any exploration following the* LUF *strategy is $O(m(n+p))$, where p is the maximum value of edge traversal counters at time* 0.

Corollary 1. *For any graph, any exploration following the* LUF *strategy achieves uniform frequency on all edges, $f_e(G) = 1/m$, even when the initial values of edge traversal counts in the graph are non-zero.*

4 Final Remarks

We have shown that locally fair strategies in undirected graphs can closely imitate random walks, allowing us to obtain an exploration which is fair with respect to all edges, and efficient in terms of cover time. However, the fairness criterion has to be chosen much more carefully than for symmetric directed graphs: Least-Used-First works, but Oldest-First does not.

In future work it would be interesting to study modified notions of equity, which are inspired by random walks which select the next edge to be traversed with non-uniform probability. For example, it is possible to decrease the general-case bound on the cover time of a random walk to $O(|V|^2 \log |V|)$, by applying a probability distribution which reflects the degrees of the nearest neighbors of the current node [11]. It is an open question whether a similar bound can be obtained in the deterministic sense using a derandomized strategy.

References

1. Aldous, D., Fill, J.: Reversible Markov Chains and Random Walks on Graphs (2001), `http://stat-www.berkeley.edu/users/aldous/RWG/book.html`
2. Aleliunas, R., Karp, R.M., Lipton, R.J., Lovász, L., Rackoff, C.: Random walks, universal sequences and the complexity of maze problems. In: Proceedings of the 20th Annual IEEE Symposium on the Foundations of Computer Science (FOCS 1979), pp. 218–223 (1979)

3. Bender, M.A., Fernández, A., Ron, D., Sahai, A., Vadhan, S.P.: The power of a pebble: Exploring and mapping directed graphs. Information and Computation 176(1), 1–21 (2002)
4. Bhatt, S.N., Even, S., Greenberg, D.S., Tayar, R.: Traversing directed eulerian mazes. Journal of Graph Algorithms and Applications 6(2), 157–173 (2002)
5. Cooper, J., Doerr, B., Friedrich, T., Spencer, J.: Deterministic Random Walks on Regular Trees. In: Proceedings of 19th ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), pp. 766–772 (2008)
6. Deng, X., Papadimitriou, C.H.: Exploring an Unknown Graph. Journal of Graph Theory 32(3), 265–297 (1999)
7. Doerr, B., Friedrich, T.: Deterministic Random Walks on the Two-Dimensional Grid. Combinatorics, Probability and Computing (to appear, 2009), doi:10.1017/S0963548308009589
8. Fraigniaud, P., Ilcinkas, D., Peer, G., Pelc, A., Peleg, D.: Graph exploration by a finite automaton. Theoretical Computer Science 345(2-3), 331–344 (2005)
9. Gąsieniec, L., Pelc, A., Radzik, T., Zhang, X.: Tree exploration with logarithmic memory. In: Proceedings 19th ACM-SIAM Symposium on Discrete Algorithms (SODA 2007), pp. 585–594 (2007)
10. Hemmerling, A.: Labyrinth Problems: Labyrinth-Searching Abilities of Automata. Teubner-Texte zur Mathematik 114 (1989)
11. Ikeda, S., Kubo, I., Yamashita, M.: The hitting and cover times of random walks on finite graphs using local degree information. Theoretical Computer Science 410(1), 94–100 (2009)
12. Kahn, J., Kim, J.H., Lovász, L., Vu, V.H.: The cover time, the blanket time, and the Matthews bound. In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS 2000), pp. 467–475. IEEE, Los Alamitos (2000)
13. Koenig, S.: Complexity of Edge Counting. In: Goal-Directed Acting with Incomplete Information. Technical Report CMU-CS-97-199, Carnegie Mellon University (1997)
14. Koenig, S., Simmons, R.G.: Easy and Hard Testbeds for Real-Time Search Algorithms. In: Proceedings of the National Conference on Artificial Intelligence, pp. 279–285 (1996)
15. Koucký, M.: Universal traversal sequences with backtracking. Journal of Computer and System Sciences 65(4), 717–726 (2002)
16. Lovász, L.: Random walks on graphs: A survey. Bolyai Society Mathematical Studies 2, 353–397 (1996)
17. Reingold, O.: Undirected ST-connectivity in log-space. In: Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC 2005), pp. 376–385 (2005)
18. Saks, M.E.: Randomization and derandomization in space-bounded computation. In: Proceedings of the 11th Annual IEEE Conference on Computational Complexity, pp. 128–149 (1996)
19. Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Smell as a Computational Resource — a Lesson We Can Learn from the Ants. In: Proceedings of Fourth Israeli Symposium on Theory of Computing and Systems (ISTCS 1996), pp. 219–230 (1996)
20. Wagner, I.A., Lindenbaum, M., Bruckstein, A.M.: Distributed Covering by Ant-Robots Using Evaporating Traces. IEEE Transactions on Robotics and Automation 15(5), 918–933 (1999)
21. Winkler, P., Zuckerman, D.: Multiple cover time. Random Structures and Algorithms 9(4), 403–411 (1996)
22. Yanovski, V., Wagner, I.A., Bruckstein, A.M.: A Distributed Ant Algorithm for Efficiently Patrolling a Network. Algorithmica 37, 165–186 (2003)

On a Network Generalization of the Minmax Theorem

Constantinos Daskalakis[1] and Christos H. Papadimitriou[2,⋆]

[1] Microsoft Research, New England
[2] U.C. Berkeley

Abstract. We consider graphical games in which the edges are zero-sum games between the endpoints/players; the payoff of a player is the sum of the payoffs from each incident edge. Such games are arguably very broad and useful models of networked economic interactions. We give a simple reduction of such games to two-person zero-sum games; as a corollary, a mixed Nash equilibrium can be computed efficiently by solving a linear program and rounding off the results. Our results render polynomially efficient, and simplify considerably, the approach in [3].

1 Introduction

In 1928, von Neumann proved that every two-person zero-sum game has the minmax property [8], and thus a randomized equilibrium — which, we now know, is easily computable via linear programming. According to Aumann, two-person strictly competitive games — that is zero-sum games (see the discussion in the last section) — are "one of the few areas in game theory, and indeed in the social sciences, where a fairly sharp, unique prediction is made" [2]. In this paper, we present a sweeping generalization of this class to multi-player games played on a network.

Networked Interactions. In recent years, with the advent of the Internet and the many kinds of networks it enables, there has been increasing interest in games in which the players are nodes of a graph, and payoffs depend on the actions of a player's neighbors [6]. One interesting class of such games are the *graphical polymatrix games*, in which the edges are two-person games, and, once all players have chosen an action, the payoff of each player is the sum of the payoffs from each game played with each neighbor. For example, games of this sort with coordination games at the edges are useful for modeling the spread of ideas and technologies over social networks [7].

But what if the games at the edges are zero-sum — that is, we have a *network of competitors?* Do von Neumann's positive results carry over to this interesting case? Let us examine a few simple examples. If the network consists of isolated edges, then of course we have many independent zero-sum games and we are

[⋆] Supported by NSF grants CCF-0635319, CCF-0515259, a gift from Yahoo! Research, and a MICRO grant.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 423–434, 2009.

done. The next simplest case is the graph consisting of two adjacent edges. It turns out that in this case too von Neumann's ideas work: We could write the game, from the middle player's point of view, as a linear program seeking the mixed strategy x such that

$$\begin{aligned} &\max z_1 + z_2 \\ \text{subject to } &A_1 x \geq z_1 \\ &A_2 x \geq z_2, \end{aligned}$$

where A_1 and A_2 are the middle player's payoff matrices against the two other players. In other words, the middle player assumes that his two opponents will each punish her separately as much as they can, and seeks to minimize the total damage. In fact, a little thought shows that this idea can be generalized to any star network.

But what if the network is a triangle, for example? Now the situation becomes more complicated. For example, if player u plays matching pennies, say, with players v and w (take the stakes of the game with v to be higher than the stakes of the game with w), while v and w play between them, for much higher stakes, a game that rewards v for playing *heads*, then v cannot afford to pay attention to u, and u can steal a positive payoff along the edge (u, v), so that her total payoff is positive — despite the fact that she is playing two matching pennies games. Is there a general method for computing Nash equilibria in such three-player zero-sum polymatrix games? Or is this problem PPAD-complete?

Our main result (Theorem 2) is a reduction implying that *in any zero-sum graphical polymatrix game a Nash equilibrium can be computed in polynomial time,* by simply solving a two-player zero-sum game and rounding off the equilibrium. In other words, we show that there is a very broad and natural class of tractable network games to which von Neumann's method applies rather directly. The basic idea of the reduction is very simple: We create two players whose strategy set is equal to the *union* of the actions of all players, and have both of them "represent" all players. To make sure that the two players randomize evenly between the players they represent, we make them play, on the side, a high-stakes game of generalized rock-paper-scissors. It is not hard to see that any minmax strategy of this two-person zero-sum game can be made (by increasing the stakes of the side game) arbitrarily close to a Nash equilibrium of the original game.[1]

We prove our main result in Section 2. In Section 3 we show an interesting consequence: if the nodes of the network run any distributed iterative learning algorithm of the bounded regret variety known to perform well in many contexts, then the whole game converges to the Nash equilibrium (Theorem 3).

[1] Ilan Adler (private communication, April 2009) pointed out to us a proof of our main result by a direct reduction to linear programming: Formulate the two-player game (without the generalized rock-paper-scissors part) as a linear program, adding constraints which require that each of the two players assigns the same total probability mass to the strategies of each of the players it represents.

Related work. In a very interesting paper [3] (which we discovered after we had proved our results...), Bregman and Fokin present a general approach to solving what they call *separable zero-sum games:* multiplayer games that are zero-sum, and in which the payoff of a player is the sum of the payoffs of the player's interactions with each other player. Their approach is to formulate such games as a linear program with huge dimensions but low rank, and then solve it by a sequence of reductions to simpler and simpler linear programs that can be solved by the column generation version of the simplex method in a couple of special cases, one of which is our zero-sum polymatrix games. Even though their technique does not amount to a polynomial-time algorithm, we believe that it can be turned into one by a sophisticated application of the ellipsoid method and multiple layers of separating hyperplane generation algorithms. In contrast, our method is a very simple and direct reduction to two-player zero-sum games.

Definitions. An *n-player zero-sum graphical polymatrix game* is defined in terms of an undirected graph $G = (V, E)$, where $V := [n]$ is the set of players, and, for each edge $[u, v] \in E$, an $m_u \times m_v$ real matrix $A^{u,v}$ and another $A^{v,u} = -(A^{u,v})^{\mathrm{T}}$. That is, each player/node u has a set of actions, $[m_u]$, and each edge is a zero-sum game played between its two endpoints. Given any mapping f from V to the natural numbers such that $f(u) \in [m_u]$ for all $u \in V$ — that is, any choice of actions for the players, the payoff of player $u \in V$ is defined as

$$P_u[f] = \sum_{[u,v]\in E} A^{u,v}_{f(u),f(v)}.$$

In other words, the payoff of each player is the sum of all payoffs of the zero-sum games played with the player's neighbors.

In any game, a *(mixed) Nash equilibrium* is a distribution on actions for each player, such that, for each player, all actions with positive probabilities are best responses in expectation. In an *ϵ-Nash equilibrium*, all actions played by a player with positive probability give her expected utility which is within an additive ϵ from the expected utility given by the best response. A weaker but related notion of approximation is the notion of an *ϵ-approximate Nash equilibrium*, in which the mixed strategy of a player gives her expected utility that is within an additive ϵ from the expected utility of the best response. Clearly, an *ϵ-Nash equilibrium* is also an *ϵ-approximate Nash equilibrium*; but the opposite implication is not always true. Nevertheless, the two notions are computationally related as follows.

Proposition 1. *[4] Given an ϵ-approximate Nash equilibrium of an n-player game, we can compute in polynomial time a $\sqrt{\epsilon} \cdot (\sqrt{\epsilon} + 1 + 4(n-1)\alpha_{\max})$-Nash equilibrium, where $\alpha_{\max}$ is the magnitude of the maximum in absolute value possible utility of a player in the game.*

2 Main Result

Theorem 2. *There is polynomial-time reduction from any zero-sum graphical polymatrix game $\mathcal{GG}$ to a symmetric zero-sum bimatrix game $\mathcal{G}$, such that from* *any Nash equilibrium of $\mathcal{G}$ one can recover in polynomial time a Nash equilibrium of $\mathcal{GG}$.*

Proof of Theorem 2. In our construction we use a generalization of the well known rock-paper-scissors game, defined below.

Definition 1 (Generalized Rock-Paper-Scissors). *For an odd integer $n > 0$, the* n-strategy rock-paper-scissors game *is a symmetric zero-sum bimatrix game $(\Gamma, -\Gamma)$ with n strategies per player such that for all $u, v \in [n]$:*

$$\Gamma_{u,v} = \begin{cases} +1, & \textit{if} \quad v = u+1 \mod n \\ -1, & \textit{if} \quad v = u-1 \mod n \\ 0, & \textit{otherwise.} \end{cases}$$

It is not hard to see that, for every odd n, the unique Nash equilibrium of the n-strategy generalized rock-paper-scissors game is the uniform distribution over both players' strategies. Now let $\mathcal{GG} = \{A^{u,v}\}_{[u,v]\in E}$ be an n-player zero-sum graphical polymatrix game with edge set E, whose u-th player has m_u strategies. Assuming without loss of generality that n is odd, let us define the embedding $\mathcal{G}$ of $\mathcal{GG}$ into the n-strategy rock-paper-scissors game with scaling parameter $M > 0$ as follows: $\mathcal{G} = (R, C)$ is an $\sum_u m_u \times \sum_u m_u$ bimatrix game, whose rows and columns are indexed by pairs $(u : i)$, of players $u \in [n]$ and strategies $i \in [m_u]$, such that, for all $u, v \in [n]$, $i \in [m_u]$, $j \in [m_v]$,

$$R_{(u:i),(v:j)} = M \cdot \Gamma_{u,v} + A^{u,v}_{i,j}$$
$$C_{(u:i),(v:j)} = -M \cdot \Gamma_{u,v} + A^{v,u}_{j,i}.$$

In the above, we take $A^{u,v}$ and $A^{v,u}$ to be the all-zero matrices if $[u,v] \notin E$. Observe that $\mathcal{G}$ is zero-sum and also symmetric, since the generalized rock-paper-scissors game is symmetric.

Lemma 1. *Let $n > 0$ be an odd integer, $\mathcal{GG} = \{A^{u,v}\}_{[u,v]\in E}$ a zero-sum graphical polymatrix game whose largest in absolute value payoff entry has magnitude M/L, and $\mathcal{G} = (R, C)$ the embedding of $\mathcal{GG}$ into the n-strategy rock-paper-scissors game, with scaling parameter M. Then for all $u \in [n]$, in any Nash equilibrium (x, y) of $\mathcal{G}$, $x_u, y_u \in (\frac{1}{n} - \frac{n}{L}, \frac{1}{n} + \frac{n}{L})$, where $x_u = \sum_{i\in[m_u]} x_{u:i}$ and $y_u = \sum_{i\in[m_u]} y_{u:i}$ is the probability mass assigned by x and y to the block of strategies $(u : \cdot)$.*

Proof of Lemma 1. Observe first that, since $\mathcal{G}$ is a symmetric zero-sum game, the value of both players is 0 in every Nash equilibrium. We will use this to argue that $x_u \geq x_{(u+2 \mod n)} - \frac{1}{L}$, for all $u \in [n]$, and similarly for y. This is enough to conclude the proof of the lemma. For a contradiction, suppose that, in some Nash equilibrium (x, y), $x_u < x_{(u+2 \mod n)} - \frac{1}{L}$, for some u. Then the payoff to the column player for playing strategy $(u + 1 \mod n : j)$, for any $j \in [m_{u+1 \mod n}]$, is at least

$$Mx_{(u+2 \mod n)} - Mx_u - \frac{M}{L} > 0.$$

Since (x, y) is an equilibrium, the expected payoff to the column player from y must be at least as large as the expected payoff from $(u+1 \mod n : j)$, so in particular larger than 0. But this is a contradiction since we argued that in any Nash equilibrium of $\mathcal{G}$ the payoff of each player is 0. ■

We argue next that, given any Nash equilibrium (x, y) of $\mathcal{G}$, we can extract an approximate equilibrium of the game $\mathcal{GG}$ by assigning to each node u of $\mathcal{GG}$ the marginal distribution assigned by x to the block of strategies $(u : i)$, $i \in [m_u]$. For each node u, let us define the distribution $\hat{x}_u$ over $[m_u]$ as follows

$$\hat{x}_u(i) = \frac{x_{u:i}}{x_u}, \qquad \text{for all } i \in [m_u]. \tag{1}$$

Lemma 2. *In the setting of Lemma 1, if (x, y) is a Nash equilibrium of $\mathcal{G}$, then the collection of mixed strategies $\{\hat{x}_u\}_u$ is a $\frac{2M \cdot n^3}{L^2}$–Nash equilibrium of $\mathcal{GG}$.*

Proof of Lemma 2. Notice that, because $\mathcal{G}$ is a symmetric zero-sum game, if x is a minimax strategy of the row player, then x is also a minimax strategy of the column player. Hence, the pair of mixed strategies (x, x) is also a Nash equilibrium of $\mathcal{G}$. Now, for every node u of the polymatrix game, we are going to show that the collection $\{\hat{x}_u\}_u$ satisfies the equilibrium conditions at node u approximately. Indeed, because (x, x) is a Nash equilibrium of $\mathcal{G}$ it must be that, for all $i, j \in [m_u]$:

$$\mathcal{E}[\mathcal{P}_{u:i}] > \mathcal{E}[\mathcal{P}_{u:j}] \quad \Rightarrow \quad x_{u:j} = 0, \tag{2}$$

where

$$\mathcal{E}[\mathcal{P}_{u:i}] = \sum_v M \cdot \Gamma_{u,v} \cdot x_v + \sum_{[u,v]\in E} \sum_{\ell \in [m_v]} A^{u,v}_{i,\ell} \cdot x_{v:\ell}$$

is the expected payoff to the row player of $\mathcal{G}$ for playing strategy $(u : i)$. From Lemma 1 we have

$$\left| \sum_{[u,v]\in E} \sum_{\ell \in [m_v]} A^{u,v}_{i,\ell} \cdot x_{v:\ell} - \frac{1}{n} \sum_{[u,v]\in E} \sum_{\ell \in [m_v]} A^{u,v}_{i,\ell} \cdot \hat{x}_v(\ell) \right| \leq \frac{M \cdot n^2}{L^2}.$$

Hence, (2) implies

$$\frac{1}{n} \sum_{[u,v]\in E} \sum_{\ell \in [m_v]} A^{u,v}_{i,\ell} \cdot \hat{x}_v(\ell) > \frac{1}{n} \sum_{[u,v]\in E} \sum_{\ell \in [m_v]} A^{u,v}_{j,\ell} \cdot \hat{x}_v(\ell) + \frac{2M \cdot n^2}{L^2} \;\Rightarrow\; \hat{x}_u(j) = 0,$$

which is equivalent to

$$\mathcal{E}[P_{u:i}] > \mathcal{E}[P_{u:j}] + \frac{2M \cdot n^3}{L^2} \quad \Rightarrow \quad \hat{x}_u(j) = 0, \tag{3}$$

where $\mathcal{E}[P_{u:i}]$ is the expected payoff of node u in $\mathcal{GG}$ for playing pure strategy i, if the other players play according to the collection of mixed strategies $\{\hat{x}_v\}_{v \neq u}$.

Since (3) holds for all $u \in [n]$, $i, j \in [m_u]$, the collection $\{\hat{x}_u\}_u$ is a $\frac{2M \cdot n^3}{L^2}$-Nash equilibrium of $\mathcal{GG}$. ■

Choosing $M = 2^{q(|\mathcal{GG}|)} 2n^3 u_{\max}^2$ and $L = \frac{M}{u_{\max}}$, where $q(|\mathcal{GG}|)$ is some polynomial in the size of $\mathcal{GG}$, and $u_{\max}$ the magnitude of the maximum in absolute value entry in the payoff tables of $\mathcal{GG}$, the collection of mixed strategies $\{\hat{x}_u\}_u$ obtained from a Nash equilibrium (x, y) of $\mathcal{G}$, constitutes a $2^{-q(|\mathcal{GG}|)}$-Nash equilibrium of the game $\mathcal{GG}$. If $q(\cdot)$ is a sufficiently large polynomial, then a $2^{-q(|\mathcal{GG}|)}$-Nash equilibrium of $\mathcal{GG}$ can be transformed in polynomial time to an exact equilibrium. To see this, let us consider the following linear program with respect to the variables z and $\{\hat{y}_u\}_u$, where $\hat{y}_u$ is a distribution over $[m_u]$:

$$\min z$$

$$\text{s.t.} \quad \sum_{[u,v] \in E} \sum_{\ell \in [m_v]} A^{u,v}_{i,\ell} \cdot \hat{y}_v(\ell) \geq \sum_{[u,v] \in E} \sum_{\ell \in [m_v]} A^{u,v}_{j,\ell} \cdot \hat{y}_v(\ell) - z, \quad \begin{array}{c} \forall u \in [n], \\ i \in \text{supp}(\hat{x}_u), \\ j \in [m_u]. \end{array} \tag{4}$$

In LP (4), $\text{supp}(\hat{x}_u)$ denotes the support of the distribution $\hat{x}_u$. Observe in particular that $(2^{-q(|\mathcal{GG}|)}, \{\hat{x}_u\}_u)$ is a solution of LP (4) with objective value $2^{-q(|\mathcal{GG}|)}$. But, let us assume that $q(\cdot)$ has been chosen to be larger than the bit complexity of any optimal solution to LP (4) (for any possible set of supports $\{\text{supp}(\hat{x}_u)\}_u$). It follows then that the optimal solution to LP (4) has objective value $z = 0$, so that the corresponding collection $\{\hat{y}_u\}_u$ is an exact Nash equilibrium of $\mathcal{GG}$. ■

3 Distributed Learning

One of the more subtle advantages of two-person zero-sum games is that a large variety of learning algorithms converge to the Nash equilibrium. Hence in this section we study the behavior arising if every player in a zero-sum graphical polymatrix game runs a no-regret learning algorithm.

Definition 2 (No-Regret Behavior). *Let every node $u \in V$ of a graphical polymatrix game choose a mixed strategy x_u^t, at every time step $t = 1, 2, \ldots$. We say that the sequence of strategies $(x_u^t)_t$ chosen by u is a* no-regret sequence, *if for every mixed strategy x of player u and all times T*

$$\sum_{t=1}^{T} \left(\sum_{[u,v] \in E} (x_u^t)^{\mathrm{T}} \cdot A^{u,v} \cdot x_v^t \right) \geq \sum_{t=1}^{T} \left(\sum_{[u,v] \in E} x^{\mathrm{T}} \cdot A^{u,v} \cdot x_v^t \right) - o(T), \tag{5}$$

where the function $o(T)$ could depend on the number strategies available to player u, the number of neighbors of u and magnitude of the maximum in absolute value entry in the matrices $A^{u,v}$. The function $o(T)$ is called the regret of player u at time T.

Example 1 (Multiplicative Weights-Update Algorithm). In the multiplicative weights-update algorithm (see for example [5]) each player maintains a mixed strategy. At each period, each probability is multiplied by a factor exponential in the utility the corresponding strategy would yield against the opponent's mixed strategy (and the probabilities are renormalized). If every node in a zero-sum graphical polymatrix game runs such an algorithm, then the resulting regret is $O((\sqrt{T \cdot \log m_u} + \log m_u) \cdot d_u \cdot \alpha^u_{\max})$, where m_u is the number of strategies available to player u, d_u is the degree of u, and $\alpha^u_{\max}$ is the magnitude of the largest in absolute value entry in the payoff matrices $\{A^{u,v}\}_{[u,v]\in E}$.

Our main result is the following.

Theorem 3. *Suppose that every node $u \in V$ of a zero-sum graphical polymatrix game $\mathcal{GG}$ plays a no-regret sequence of strategies $(x^t_u)_{t=1,2,\ldots}$, with regret $g(T) = o(T)$. Then, for all T, the set of strategies $\bar{x}^T_u = \frac{1}{T}\sum_{t=1}^{T} x^t_u$, $u \in V$, is a $\left(2.3 \cdot n \cdot \frac{g(T)}{T} + \frac{2}{T}\right)$-approximate Nash equilibrium of the game.*

Proof. Our proof plan is the following: Using the no-regret strategy sequences of the players of $\mathcal{GG}$ we are going to define no-regret strategy sequences for the players of the symmetric zero-sum bimatrix game $\mathcal{G}$ defined in the proof of Theorem 2. We are going to show then that the time-averages of these sequences comprise an approximate equilibrium of the game $\mathcal{G}$, if M is sufficiently large. Going back to the game $\mathcal{GG}$ using the mapping (1), we will then deduce that the time-averages of the original sequences need to also comprise an approximate equilibrium of the game $\mathcal{GG}$.

To define the no-regret sequences of the players of the bimatrix game $\mathcal{G}$, it is tempting to take, at every time step, the (uniform) average of the strategies of the players of $\mathcal{GG}$. That is, for every time $t = 1, 2, \ldots$, assign to both players of $\mathcal{G}$ the strategy x^t, such that $x^t_{u:i} = \frac{1}{n}x^t_u(i)$, for all u and i. This, however, may result in large regrets for the players of $\mathcal{G}$ (essentially because the payoffs of the side game are eliminated in this accounting, and the two players will tend to skew their distributions towards the most "lucrative" of the players that they represent). We define instead a non-uniform averaging with weights selected by solving yet another related game, in a manner that depends on the payoffs of the nodes of $\mathcal{GG}$ under the no-regret sequences.

Let us denote the average payoff of player u over the period $t = 1, \ldots, T$ as

$$\bar{P}^T_u := \frac{1}{T}\sum_{t=1}^{T}\left(\sum_{[u,v]\in E}(x^t_u)^{\mathrm{T}} \cdot A^{u,v} \cdot x^t_v\right).$$

Let also $\alpha_{\max}$ be the magnitude of the largest in absolute value entry in the payoff tables of the game $\mathcal{GG}$. We show the following lemma.

Lemma 3. *For any $Z > n^2$ and $M > 2nZ \cdot \alpha_{\max}$, there exist $c > 0$, and positive weights $\{k_u > 0 : u \in V\}$, such that for all u:*

$$M \cdot \left(\frac{1}{k_{(u+1 \bmod n)}} - \frac{1}{k_{(u-1 \bmod n)}}\right) = -\frac{1}{n}\bar{P}^T_u + c; \qquad (6)$$

$$\frac{1}{k_u} \in \left[\frac{1}{n} - \frac{n}{Z}, \frac{1}{n} + \frac{n}{Z}\right] \quad \textit{and} \quad \sum_u \frac{1}{k_u} = 1. \tag{7}$$

Recall that we have identified the vertices of $\mathcal{GG}$ with the integers in $[n]$, and without loss of generality let us assume that n is odd. Also, for conciseness, in the remaining of the proof of Theorem 3 we are going to omit "mod n." We proceed with the proof of Lemma 3.

Proof of Lemma 3. We define a symmetric bimatrix game $\mathcal{G}'$, with n strategies per player corresponding to the different nodes of the game $\mathcal{GG}$. The payoff matrices (R, C) of the row and column players of $\mathcal{G}'$ are defined as follows. For all $u, v \in V$:

$$R_{u,v} = M \cdot \mathit{\Gamma}_{u,v} + \frac{1}{n}\bar{P}_u^T; \quad C_{u,v} = -M \cdot \mathit{\Gamma}_{u,v} + \frac{1}{n}\bar{P}_v^T;$$

where $\mathit{\Gamma}_{u,v}$ is the payoff matrix defined in the proof of Theorem 2. Since the game $\mathcal{G}'$ is symmetric, there exists a symmetric equilibrium (x, x), where $x = (x_u)_{u \in V}$. We will argue first that $x_u \geq \frac{1}{n} - \frac{1}{Z}$, for all $u \in V$; from this we can easily deduce that $x_u \in \left[\frac{1}{n} - \frac{n}{Z}, \frac{1}{n} + \frac{n}{Z}\right]$, for all u. Take $u \in \arg\min_u\{x_u\}$ and suppose that $x_u < \frac{1}{n} - \frac{1}{Z}$. Then there exists a pair of nodes v and $(v+2)$ such that $x_{v+2} - x_v > \frac{1}{nZ}$. Indeed, if $x_{v+2} - x_v \leq \frac{1}{nZ}$ for all v, it would be easy to deduce (because n is odd) that $x_v \leq x_u + \frac{1}{Z} < \frac{1}{n}$, for all v, which is clearly impossible. Now, given that $x_{v+2} - x_v > \frac{1}{nZ}$, the utility of the players of the game $\mathcal{G}'$ for playing pure strategy $v+1$ is

$$M \cdot (x_{v+2} - x_v) + \frac{1}{n}\bar{P}_{v+1}^T > \frac{M}{nZ} - \alpha_{\max} > \alpha_{\max},$$

since $\alpha_{\max}$ is a bound on the absolute value of every entry in the payoff matrices of the game $\mathcal{GG}$, every node has at most n neighbors, and $\frac{M}{nZ} > 2\alpha_{\max}$. On the other hand, since $u \in \arg\min_u\{x_u\}$ it follows that the payoff of the players of the game $\mathcal{G}'$ for playing pure strategy $u-1$ is

$$M \cdot (x_u - x_{u-2}) + \frac{1}{n}\bar{P}_{u-1}^T \leq \alpha_{\max}.$$

Since (x, x) is an equilibrium, it must be that $x_{u-1} = 0$. It follows that there must exist some w such that $x_w = 0$ and $x_{w-1} \neq 0$. But, the utility of the players of the game $\mathcal{G}'$ for playing pure strategy $w-1$ is

$$M \cdot (x_w - x_{w-2}) + \frac{1}{n}\bar{P}_{w-1}^T \leq \alpha_{\max}.$$

And, using again the fact that the utility for playing $v+1$ is larger than $\alpha_{\max}$, it follows that $x_{w-1} = 0$ (a contradiction). This finishes the proof of Assertion (7) taking $k_u := x_u^{-1}$, for all u.

Now, we need to justify (6). Since $x_u > 0$ for all u, it follows that the expected payoff for playing every u is the same. So, there exists c such that, for all u, $M \cdot (x_{u+1} - x_{u-1}) + \frac{1}{n}\bar{P}_u^T = c$. Assertion (6) then follows. ■

Now let us choose $Z = n^2 T \mathit{\Lambda}$ (where $\mathit{\Lambda} > 1$ will be decided later), $M > 2nZ \cdot \alpha_{\max}$, and let us define strategies for the players of $\mathcal{G}$ by averaging the strategies of the nodes of $\mathcal{GG}$ with the weights $\{1/k_u\}_u$ given by Lemma 3. That is, for all t, we define the strategy x^t for each player of $\mathcal{G}$ as follows:

$$x^t_{u:i} = \frac{1}{k_u} x^t_u(i), \text{ for all } u \in [n], i \in [m_u].$$

We show that if both players of $\mathcal{G}$ adopt the sequence of strategies $x^t, t = 1, 2, \ldots$, defined above then the regret of each at time T is at most $\left(\frac{g(T)}{n} + \frac{2\alpha_{\max}}{\Lambda}\right)$.

Lemma 4. *For all mixed strategies z:*

$$\sum_{t=1}^{T} (x^t)^{\mathrm{T}} \cdot R \cdot x^t \geq z^{\mathrm{T}} \cdot R \cdot \left(\sum_{t=1}^{T} x^t\right) - \frac{g(T)}{n} - \frac{2\alpha_{\max}}{\Lambda}, \tag{8}$$

$$\sum_{t=1}^{T} (x^t)^{\mathrm{T}} \cdot C \cdot x^t \geq \left(\sum_{t=1}^{T} x^t\right)^{\mathrm{T}} \cdot C \cdot z - \frac{g(T)}{n} - \frac{2\alpha_{\max}}{\Lambda}. \tag{9}$$

Proof of Lemma 4. Since $\mathcal{G}$ is symmetric it is enough to justify (8). Indeed, for the left hand side we have:

$$\begin{aligned}
\sum_{t=1}^{T} (x^t)^{\mathrm{T}} \cdot R \cdot x^t &= \sum_{t=1}^{T} \sum_{u\in[n]} \frac{1}{k_u} \left(M \cdot \left(\frac{1}{k_{u+1}} - \frac{1}{k_{u-1}}\right) + \sum_{[u,v]\in E} \frac{1}{k_v} (x^t_u)^{\mathrm{T}} A^{u,v} x^t_v \right) \\
&= \sum_{t=1}^{T} \sum_{u\in[n]} \frac{1}{k_u} \left(-\frac{1}{n}\bar{P}^T_u + c + \sum_{[u,v]\in E} \left(\frac{1}{n} \pm \frac{n}{Z}\right) (x^t_u)^{\mathrm{T}} A^{u,v} x^t_v \right) \\
&\geq \sum_{t=1}^{T} \sum_{u\in[n]} \frac{1}{k_u} \left(-\frac{1}{n}\bar{P}^T_u + c + \frac{1}{n} \sum_{[u,v]\in E} (x^t_u)^{\mathrm{T}} A^{u,v} x^t_v - \frac{n}{Z} n\alpha_{\max} \right) \\
&= Tc - \frac{T}{n} \sum_{u\in[n]} \frac{\bar{P}^T_u}{k_u} + \frac{1}{n} \sum_{u\in[n]} \frac{1}{k_u} \sum_{t=1}^{T} \left(\sum_{[u,v]\in E} (x^t_u)^{\mathrm{T}} A^{u,v} x^t_v \right) - \frac{n^2 T}{Z}\alpha_{\max} \\
&= Tc - \frac{T}{n} \sum_{u\in[n]} \frac{\bar{P}^T_u}{k_u} + \frac{1}{n} \sum_{u\in[n]} \frac{1}{k_u} T\bar{P}^T_u - \frac{n^2 T}{Z}\alpha_{\max} \\
&= Tc - \frac{n^2 T}{Z}\alpha_{\max} = Tc - \frac{\alpha_{\max}}{\Lambda}.
\end{aligned}$$

Let us now consider a mixed strategy z such that $z_{u:i} = 1$, for some $u \in [n]$ and $i \in [m_u]$. If we establish (8) for this z, it is easy to see that (8) holds for any z.

$$\begin{aligned}
z^{\mathrm{T}} \cdot R \cdot \left(\sum_{t=1}^{T} x^t\right) &= \sum_{t=1}^{T} \left(M \cdot \left(\frac{1}{k_{u+1}} - \frac{1}{k_{u-1}}\right) + \sum_{[u,v]\in E} \frac{1}{k_v} e^{\mathrm{T}}_{u:i} A^{u,v} x^t_v \right) \\
&= \sum_{t=1}^{T} \left(-\frac{1}{n}\bar{P}^T_u + c + \sum_{[u,v]\in E} \left(\frac{1}{n} \pm \frac{n}{Z}\right) e^{\mathrm{T}}_{u:i} A^{u,v} x^t_v \right)
\end{aligned}$$

$$\leq \sum_{t=1}^{T}\left(-\frac{1}{n}\bar{P}_u^T + c + \frac{1}{n}\sum_{[u,v]\in E} e_{u:i}^{\mathrm{T}} A^{u,v} x_v^t + \frac{n}{Z} n\alpha_{\max}\right)$$
$$= Tc - \frac{T}{n}\bar{P}_u^T + \frac{1}{n}\sum_{t=1}^{T}\left(\sum_{[u,v]\in E} e_{u:i}^{\mathrm{T}} A^{u,v} x_v^t\right) + \frac{n^2}{Z}T\alpha_{\max}$$
$$= Tc - \frac{T}{n}\bar{P}_u^T + \frac{1}{n}\sum_{[u,v]\in E} e_{u:i}^{\mathrm{T}} A^{u,v}\left(\sum_{t=1}^{T} x_v^t\right) + \frac{n^2 T\alpha_{\max}}{Z} \leq Tc + \frac{g(T)}{n} + \frac{\alpha_{\max}}{\Lambda},$$

where for the last derivation we used that the strategy sequence of the node u of $\mathcal{GG}$ has regret at most $g(T)$. Combining the above bounds we get (8). ■

We argue next the following

Lemma 5. *The pair of strategies* $(\frac{1}{T}\sum_{t=1}^{T} x^t, \frac{1}{T}\sum_{t=1}^{T} x^t)$ *is a* $\frac{2}{T}\left(\frac{g(T)}{n} + \frac{2\alpha_{\max}}{\Lambda}\right)$*-approximate Nash equilibrium of the game* $\mathcal{G}$*.*

Proof of Lemma 5. Let $\Phi := \frac{g(T)}{n} + \frac{2\alpha_{\max}}{\Lambda}$, and let us fix a pure strategy z^* for the row player. (8) implies

$$\sum_{t=1}^{T}(x^t)^{\mathrm{T}} \cdot R \cdot x^t \geq z^{*\mathrm{T}} \cdot R \cdot \left(\sum_{t=1}^{T} x^t\right) - \Phi. \tag{10}$$

Recalling that $C = -R$ and setting $z = x^t$ we get from (9) that for all t:

$$-\sum_{t=1}^{T}(x^t)^{\mathrm{T}} \cdot R \cdot x^t \geq -\left(\sum_{t=1}^{T} x^t\right)^{\mathrm{T}} \cdot R \cdot x^t - \Phi. \tag{11}$$

Combining (10) and (11), we get $\left(\sum_{t=1}^{T} x^t\right)^{\mathrm{T}} \cdot R \cdot x^t + \Phi \geq z^{*\mathrm{T}} \cdot R \cdot \left(\sum_{t=1}^{T} x^t\right) - \Phi$, for all t. Summing this for $t = 1, \ldots, T$ we get

$$\left(\sum_{t=1}^{T} x^t\right)^{\mathrm{T}} \cdot R \cdot \left(\sum_{t=1}^{T} x^t\right) \geq T \cdot z^{*\mathrm{T}} \cdot R \cdot \left(\sum_{t=1}^{T} x^t\right) - 2\Phi T.$$

Dividing by T^2 and recalling that that the above holds for all z^* completes the proof. ■

We conclude the proof of Theorem 3 by arguing that the set of strategies $\{\bar{x}_u^T\}_u$, where $\bar{x}_u^T = \frac{1}{T}\sum_{t=1}^{T} x_u^t$, comprise an approximate equilibrium of the game $\mathcal{GG}$. Denote by $\Gamma := \frac{2}{T}\left(\frac{g(T)}{n} + \frac{2\alpha_{\max}}{\Lambda}\right)$ and take $\Xi = \frac{n^2}{1-\frac{n^2}{Z}}\Gamma + \frac{n^3}{Z}2\alpha_{\max} + \frac{1}{T}$.

Lemma 6. *For all* $u \in [n]$*, and for all mixed strategies* z_u *of node* u*:*

$$\sum_{[u,v]\in E}(\bar{x}_u^T)^{\mathrm{T}} \cdot A^{u,v} \cdot \bar{x}_v^T \geq \sum_{[u,v]\in E} z_u^{\mathrm{T}} \cdot A^{u,v} \cdot \bar{x}_v^T - \Xi. \tag{12}$$

Proof of lemma 6. Suppose that (12) is violated for some pair u, z_u. We are going to contradict the assertion of Lemma 5. Indeed, let us define the following strategy q for the row player of $\mathcal{G}$:

$$q_{v:i} = \begin{cases} \frac{1}{k_v}\bar{x}_v^T(i), & \text{if } v \in [n] \setminus \{u\}, i \in [m_v]; \\ \frac{1}{k_u} z_u(i), & \text{if } v = u, i \in [m_u]; \end{cases}$$

and let us consider the change in the row player's payoff if she replaces strategy $\bar{x}^T := \frac{1}{T}\sum_{t=1}^{T} x^t$ by q. Clearly, the payoff from $\bar{x}^T$ is

$$\sum_{v \in [n]} \frac{1}{k_v} (\bar{x}_v^T)^{\mathrm{T}} R_v \bar{x}^T, \tag{13}$$

where we denote by R_v the matrix v restricted to the rows $(v : i)$, $i \in [m_v]$. Similarly, the payoff that the row player gets from q is

$$\frac{1}{k_u}(z_u)^{\mathrm{T}} R_u \bar{x}^T + \sum_{v \in [n]\setminus\{u\}} \frac{1}{k_v} (\bar{x}_v^T)^{\mathrm{T}} R_v \bar{x}^T. \tag{14}$$

Subtracting the two payoffs we get that the difference between the payoff from q and the payoff from $\bar{x}^T$ is

$$\begin{aligned} \frac{1}{k_u}\left(z_u^{\mathrm{T}} R_u \bar{x}^T - (\bar{x}_u^T)^{\mathrm{T}} R_u \bar{x}^T\right) &= \frac{1}{k_u}\left(\sum_{[u,v] \in E} \frac{1}{k_v}(z_u - \bar{x}_u^T)^{\mathrm{T}} \cdot A^{u,v} \cdot \bar{x}_v^T\right) \\ &= \frac{1}{k_u}\left(\sum_{[u,v] \in E} \left(\frac{1}{n} \pm \frac{n}{Z}\right)(z_u - \bar{x}_u^T)^{\mathrm{T}} \cdot A^{u,v} \cdot \bar{x}_v^T\right) \\ &\geq \frac{1}{k_u}\sum_{[u,v] \in E}\left(\frac{1}{n}(z_u - \bar{x}_u^T)^{\mathrm{T}} \cdot A^{u,v} \cdot \bar{x}_v^T - \frac{n}{Z} 2\alpha_{\max}\right) \\ &\geq \frac{1}{k_u} \cdot \frac{1}{n} \sum_{[u,v] \in E} (z_u - \bar{x}_u^T)^{\mathrm{T}} \cdot A^{u,v} \cdot \bar{x}_v^T - \frac{1}{k_u}\frac{n^2}{Z} 2\alpha_{\max} \\ &\geq \frac{1}{k_u} \cdot \frac{1}{n} \Xi - \frac{1}{k_u}\frac{n^2}{Z} 2\alpha_{\max} > \Gamma, \end{aligned}$$

and this contradicts the assertion of Lemma 5 that $(\bar{x}^T, \bar{x}^T)$ is a Γ-approximate Nash equilibrium. ■

From Lemma 6 it follows that the strategies $\{\bar{x}_u^T\}_u$ comprise a Ξ-approximate Nash equilibrium of the game $\mathcal{GG}$. Choosing $\Lambda > \max\{8n^2, 5n^2\alpha_{\max}\}$ it follows that $\Xi < 2.3 \cdot n \cdot \frac{g(T)}{T} + \frac{3}{T}$.

4 Discussion

We believe that graphical polymatrix games are useful models of important social phenomena, such as trading or other interaction in social networks. In this paper we focused on the zero-sum variety of such games. Without restrictions on the games played on the edges, it is easy to see that the problem of computing a Nash equilibrium becomes intractable even for two strategies per player [4]. It is interesting to understand what other classes of polymatrix games have nice computational properties. For example, can the Nash equilibrium of such games, with a small number of strategies at each player, be approximated well?

Our main result raises a number of other important questions. Consider a directed graph such that along every edge (u, v) nodes u and v play the same zero-sum game $(A, -A)$. It is easy to see (assuming, for example, that the overall game is non-degenerate) that each node can be assigned a *value*, characterizing its expected payoff at equilibrium. This brings up an interesting question: which structural properties of the graph and of the position of a node in it — as well as the nature of the game A — determines these values? Such investigation could result in important insights into networked economic activity.

Finally, in an earlier version of our paper we had included an extension of our main result to the (ostensibly) more general case in which the games played at the edges are *strictly competitive,* games that share with zero-sum games this property: if both opponents change their mixed strategy, then their utilities either both stay the same, or one increases while the other decreases. In subsequent joint work with Ilan Adler [1], however, we proved that the only examples of such games are zero-sum games (or their trivial affine variants, resulting from a zero-sum game by adding a constant to all payoffs of one player, or multiplying them all by the same positive constant). In other words, this well known, and much discussed in the literature, generalization of zero-sum games is, rather astonishingly, *void!*

References

1. Adler, I., Daskalakis, C., Papadimitriou, C.H. (manuscript, 2009)
2. Aumann, R.J.: Game Theory. In: Eatwell, J., Milgate, M., Newman, P. (eds.) The New Palgrave: A Dictionary of Economics, Macmillan & Co., London (1987)
3. Bregman, L.M., Fokin, I.N.: On Separable Non-Cooperative Zero-Sum Games. Optimization 44(1), 69–84 (1998)
4. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The Complexity of Computing a Nash Equilibrium. In: STOC (2006); SIAM Journal on Computing, special issue for STOC 2006 (to appear)
5. Freund, Y., Schapire, R.E.: Adaptive Game Playing Using Multiplicative Weights. Games and Economic Behavior 29, 79–103 (1999)
6. Kearns, M.J., Littman, M.L., Singh, S.P.: Graphical Models for Game Theory. In: UAI (2001)
7. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the Spread of Influence through a Social Network. In: SIGKDD (2003)
8. von Neumann, J.: Zur Theorie der Gesellschaftsspiele. Math. Annalen 100, 295–320 (1928)

Rate-Based Transition Systems for Stochastic Process Calculi*

Rocco De Nicola[1], Diego Latella[2], Michele Loreti[1], and Mieke Massink[2]

[1] Dipartimento di Sistemi e Informatica - Università di Firenze
[2] Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo"- CNR

Abstract. A variant of Rate Transition Systems (RTS), proposed by Klin and Sassone, is introduced and used as the basic model for defining stochastic behaviour of processes. The transition relation used in our variant associates to each process, for each action, the set of possible futures paired with a measure indicating their rates. We show how RTS can be used for providing the operational semantics of stochastic extensions of classical formalisms, namely CSP and CCS. We also show that our semantics for stochastic CCS guarantees associativity of parallel composition. Similarly, in contrast with the original definition by Priami, we argue that a semantics for stochastic π-calculus can be provided that guarantees associativity of parallel composition.

1 Introduction

Performance and dependability issues are of utmost importance for "network-aware" computing, due to the enormous size of systems—networks typically consist of thousands or even millions of nodes—and their strong dependence on mobility and interaction. Spontaneous computer crashes may easily lead to failure of remote execution or process movement, while spurious network failures may cause loss of code fragments or unpredictable delays. The enormous magnitude of computing devices involved in global computing yields failure rates that no longer can be ignored. The presence of such random phenomena implies that correctness of global computing software and their safety guarantees are no longer rigid notions.

A number of stochastic process algebras have been proposed in the last two decades with the aim of combining two very successful approaches to concurrent systems specification and analysis, namely Labeled Transition Systems (LTS) and Continuous Time Markov Chains (CTMC). Indeed, LTS have proved to be a very convenient framework for providing compositional semantics of languages for specifying large complex system and for the analysis of their *qualitative properties* of systems. CMTC have, instead, been used mainly in performance evaluation, and thus for the analysis of *quantitative properties* taking into account

* Research partially funded by EU IP SENSORIA (contract n. 016004), EU project RESIST/FAERUS (IST-2006-026764), the CNR-RSTL project XXL, the Italian national projects FIRB-MUR TOCAI.IT and PRIN PACO.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 435–446, 2009.

also aspects related to both time and probability. Examples of stochastic process algebras include TIPP [8], PEPA [14], EMPA [2], stochastic π-calculus [22] and StoKlaim [5]. Semantics of these calculi have been provided by resorting to variants of the Structured Operational Semantics (SOS) approach but, as noticed in [18], they are not based on any general framework for operational semantics descriptions of stochastic processes, and indeed differ substantially from one another. Moreover, due to the different underlying models, it is rather difficult to appreciate differences and similarities of such semantics.

The common feature of all the above mentioned approaches is that the actions used to label transitions are enriched with rates of exponentially distributed random variables (r.v.) characterising their mean duration. On the other hand, they differ for the way synchronization rates are determined, the actions performed by processes are counted, etc.. Moreover, although the same class of r.v. is assumed, i.e. exponentially distributed ones, we have that the underlying models and notions are significantly different, ranging, e.g. from multi relations for PEPA, to *proved transition systems* for stochastic π-calculus, to *unique rate names* for StoKlaim.

In [18], a variant of Labelled Transition Systems is introduced, namely Rate Transition Systems (RTS), which is used for defining the stochastic semantics of process calculi. The main feature of RTS is that the transition relation is actually a function ρ associating a rate value in $\mathbb{R}_{\geq 0}$ to each *state-action-state* triple: $\rho(P, \alpha, Q) = \lambda > 0$ if and only if P evolves via action α to Q with rate λ. Stochastic semantics of process calculi are defined by relying on the general framework of SGSOS. Moreover, in [18] conditions are put forward for guaranteeing associativity of the parallel composition operator in the SGSOS framework. It is then proved that one cannot guarantee associativity of parallel composition operator up to stochastic bisimilarity when the synchronisation paradigm of CCS is used in combination with the synchronisation rate computation based on *apparent rates* [14]. This implies for instance that parallel composition of Stochastic π is not associative. And, it has to be said that associativity of parallel composition is a higly desirable property in particular for networks and distributed systems, especially in presence of dynamic process creation.

In the present paper, we introduce a variant of RTS where the transition relation $\longrightarrow$ associates to a given process P and a given transition label α a function, denoted by $\mathscr{P}$, $\mathscr{Q}$,..., mapping each term into a non-negative real number. The reduction $P \xrightarrow{\alpha} \mathscr{P}$ has the following meaning: if $\mathscr{P}(Q) = v,\ (with\ v \neq 0)$, then Q is reachable from P by executing α, the duration of such execution being exponentially distributed with rate v; if $\mathscr{P}(Q) = 0$, then Q is not reachable from P via α. We have then that if $P \xrightarrow{\alpha} \mathscr{P}$ then $\oplus\mathscr{P} \overset{def}{=} \sum_Q \mathscr{P}(Q)$ represents the total rate of α in P. Moreover, we adapt the *apparent rate* approach to calculi like CCS and, consequently, π-calculus. This adaptation guarantees associativity and commutativity properties of parallel composition. The approach is somewhat reminiscent of that of Deng et al. [7] where probabilistic process algebra terms are associated to a discrete probability distribution over such terms.

In the rest of the paper, after introducing Rate Transition Systems, we show how they can be used for providing the stochastic operational semantics of the two classical formalisms CSP and CCS. We prove that our characterizations of the stochastic variants of the above mentioned process calculi either are in full agreement with the originally proposed ones or show the differences. Furthermore, we show that in our approach associativity of the parallel composition operator can be guaranteed also in the stochastic extensions of calculi based on a two party synchronisation pattern, like CCS and π-calculus. We also introduce a natural notion of bisimulation over RTS that is finer than Markovian bisimulation and use it to establish the associativity results. Due to space limitation, all proofs are omitted. For the same reason, the treatment of stochastic π-calculus is omitted; the complete RTS semantics, related results and proofs can be found in detail in [6].

2 Rate Transitions Systems

The semantics of process algebras is classically described by means of Labelled Transitions Systems (LTS). The semantics of stochastic process algebras [11,15] are classically defined by means of Continuous Time Markov Chains (CTMC). Here we assume the reader is familiar with basic notions concerning CTMC and exponentially distributed r.v. [9]; we only recall our working definition of CTMC:

Definition 1. *A Continuous-Time Markov Chain (CTMC) is a tuple* $(S, \mathbf{R})$ *where* S *is a countable set of states and* $\mathbf{R}$ *a* rate matrix *assigning non-negative values to pairs of states, such that for all* $s \in S$, $\sum_{s' \in S} \mathbf{R}[s, s']$ *converges*[1].

Intuitively, $(S, \mathbf{R})$ models a stochastic process where, for any state $s \in S$, whenever $\sum_{s' \in S} \mathbf{R}[s, s'] > 0$, the probability to take an outgoing transition from s by (continuous) time t is $1 - e^{-\sum_{s' \in S} \mathbf{R}[s,s'] \cdot t}$, i.e. the s-residence time is exponentially distributed with rate $\sum_{s' \in S} \mathbf{R}[s, s']$, and the probability to take a transition from state s to state s', given that s is left, is $\frac{\mathbf{R}(s,s')}{\sum_{s'' \in S} \mathbf{R}[s,s'']}$. If $\sum_{s' \in S} \mathbf{R}[s, s'] = 0$, then s is said to be *absorbing*, i.e. if the process enters state s, it remains in s forever. In what follows, the rate matrix function $\mathbf{R}$ of any CTMC $(S, \mathbf{R})$ is lifted to sets of states $C \subseteq S$ in the natural way: $\mathbf{R}[s, C] \overset{def}{=} \sum_{s' \in C} \mathbf{R}[s, s']$.

2.1 Rate Transition Systems and Markov Chains

We now present RTS, a generalisation of LTS, specifically designed for describing stochastic behaviours of process algebras and instrumental to generate CMTC to be associated to given systems. RTS have been introduced in [18], however, in that work, a rate is associated to each transition, while in our approach the transition relation associates to each state and to each action a function mapping each state to a non negative real number. Formally:

[1] Notice that this definition allows self loops in CTMC, i.e. $\mathbf{R}[s, s] > 0$ is allowed. We refer the reader to [1] for details.

Definition 2 (Rate Transition Systems). *A rate transition systems is a triple* $(S, A, \longrightarrow)$ *where* S *is a set of states,* A *a set of transition labels,* $\longrightarrow$ *a subset of* $S \times A \times \Sigma_S$ *and* Σ_S *is the set* $[S \to \mathbb{R}_{\geq 0}]$ *of total functions from* S *to* $\mathbb{R}_{\geq 0}$*.*

In the sequel RTS will be denoted by $\mathcal{R}$,$\mathcal{R}_1$,$\mathcal{R}'$,..., while $\mathscr{P}, \mathscr{Q}, \mathscr{R}, \ldots$ will range over the elements of Σ_S. Intuitively, $s_1 \xrightarrow{\alpha} \mathscr{P}$ and $\mathscr{P}(s_2) = v \in \mathbb{R}_{>0}$ means that s_2 is reachable from s_1 via the execution of α with rate v. On the other hand, $\mathscr{P}(s_2) = 0$ means that s_2 is not reachable from s_1 via α. Notice that the above definition, differently from the original one in [18], includes also *nondeterministic* systems where from a certain state the same actions can lead to different rate functions.

Notation 1. *In the sequel, we will use* $\emptyset$ *to denote the constant function* 0*, while* $[s_1 \mapsto v_1, \ldots, s_n \mapsto v_n]$ *will denote a function associating* v_i *to* s_i *and* 0 *to all the other states. Moreover, for* $X \subseteq S$ *and* $\mathscr{P} \in \Sigma_S$ $\mathscr{P}(X) = \sum_{s \in X} \mathscr{P}(s)$ *and* $\oplus\mathscr{P}$ *denotes* $\mathscr{P}(S)$*.*

Definition 3. *Let* $\mathcal{R} = (S, A, \longrightarrow)$ *be an RTS, then:*

- $\mathcal{R}$ *is* well defined *if and only if for each* $s \in S$*,* $\alpha \in A$ *and* $\mathscr{P} \in \Sigma_S$ *such that* $s \xrightarrow{\alpha} \mathscr{P}$ *we have:* $\exists x : \oplus\mathscr{P} \leq x$
- $\mathcal{R}$ *is* image finite *if and only if for each* $s \in S$*,* $\alpha \in A$ *and* $\mathscr{P}$ *such that* $s \xrightarrow{\alpha} \mathscr{P}$ *either* $\mathscr{P} = \emptyset$ *or* $\mathscr{P} = [s_1 \mapsto \lambda_1, \ldots, s_n \mapsto \lambda_n]$
- $\mathcal{R}$ *is* fully stochastic *if and only if for each* $s \in S$*,* $\alpha \in A$*,* $\mathscr{P}$ *and* $\mathscr{Q}$ *we have:* $s \xrightarrow{\alpha} \mathscr{P}, s \xrightarrow{\alpha} \mathscr{Q} \Longrightarrow \mathscr{P} = \mathscr{Q}$

In the following we will only consider *well defined* RTS.

In general, given RTS $(S, A, \longrightarrow)$ we will be interested in the CTMC composed by the states reachable from a subset C of S only via the actions in $A' \subseteq A$. To that purpose we use the following two definitions:

Definition 4. *For sets* $C \subseteq S$ *and* $A' \subseteq A$*, the set of derivatives of* C through A'*, denoted* $Der(C, A')$*, is the smallest set such that:*

- $C \subseteq Der(C, A')$*,*
- *if* $s \in Der(C, A')$ *and there exists* $\alpha \in A'$ *and* $\mathscr{Q} \in \Sigma_S$ *such that* $s \xrightarrow{\alpha} \mathscr{Q}$ *then* $\{s' \mid \mathscr{Q}(s') > 0\} \subseteq Der(C, A')$

Definition 5. *Let* $\mathcal{R} = (S, A, \longrightarrow)$ *be a* fully stochastic *RTS, for* $C \subseteq S$*, the CTMC of* C*, when one considers only actions in a finite set* $A' \subseteq A$ *is defined as* $CTMC[C, A'] \overset{def}{=} (Der(C, A'), \mathbf{R})$ *where for all* $s_1, s_2 \in Der(C, A')$*:* $\mathbf{R}[s_1, s_2] \overset{def}{=} \sum_{\alpha \in A'} \mathscr{P}^{\alpha}(s_2)$ *with* $s_1 \xrightarrow{\alpha} \mathscr{P}^{\alpha}$*.*

Notice that RTS are naturally mapped to Continuous Time Markov Decision Processes [23,12]. Moreover, it turns out that general, non-fully stochastic, RTS are a convenient framework for automatic time bounded reachability probability analysis of Interactive Markov Chains [10,12], where nondeterminism and time are treated in an orthogonal way.

2.2 Rate Aware Bisimulation

Two key concepts in the theory of process algebras are the notions of *behavioural equivalence and congruence.* In the literature, many behavioural equivalences have been proposed which differ in what they consider essential aspects of *observable* behaviour. More recently, such behavioural equivalences have been extended to *Markovian* process algebras.

In this paper, we focus on *Strong Markovian Bisimulation Equivalence* [4,14], which has a direct correspondence with the notion of *lumpability*—a successful minimisation technique—of CTMCs [14,17], and for which efficient algorithms have been devised for computing the best possible lumping [13].

Definition 6 (Strong Markovian bisimilarity [4]). *Given CTMC* $(S, \mathbf{R})$

- *An equivalence relation $\mathcal{E}$ on S is a Markovian bisimulation on S if and only if for all $(s_1, s_2) \in \mathcal{E}$ and for all equivalence classes $C \in S_{/\mathcal{E}}$ the following condition holds: $\mathbf{R}[s_1, C] \leq \mathbf{R}[s_2, C]$.*
- *Two states $s_1, s_2 \in S$ are strong Markovian bisimilar, written $s_1 \sim_M s_2$, if and only if there exists a Markovian bisimulation $\mathcal{E}$ on S with $(s_1, s_2) \in \mathcal{E}$.*

We introduce *Rate Aware Bisimulation Equivalence* as the natural equivalence induced by the next state function and show that it implies Strong Markovian Bisimulation Equivalence. We point out that our semantic approach makes the definition of the Rate Aware Bisimulation Equivalence very natural.

Definition 7 (Rate Aware Bisimilarity). *Given RTS* $(S, A, \longrightarrow)$

- *An equivalence relation $\mathcal{E} \subseteq S \times S$ is a* rate aware *bisimulation if and only if, for all $(s_1, s_2) \in \mathcal{E}$, for all α and $\mathscr{P}$:*

$$s_1 \xrightarrow{\alpha} \mathscr{P} \Longrightarrow \exists \mathscr{Q} : \ s_2 \xrightarrow{\alpha} \mathscr{Q} \wedge \forall C \in \mathcal{C}_{/\mathcal{E}} \mathscr{P}(C) = \mathscr{Q}(C)$$

- *Two states $s_1, s_2 \in S$ are* rate aware bisimilar *($s_1 \sim s_2$) if there exists a rate aware bisimulation $\mathcal{E}$ such that $(s_1, s_2) \in \mathcal{E}$.*

For instance, if we consider the RTS with set of states $\{s_i | 1 \leq i \leq 7\}$, where $s_1 \xrightarrow{\alpha} [s_3 \mapsto \lambda_1, s_2 \mapsto \lambda_2]$, $s_4 \xrightarrow{\alpha} [s_5 \mapsto \lambda_3, s_6 \mapsto \lambda_4]$ and $s_7 \xrightarrow{\alpha} [s_8 \mapsto \lambda_5]$, states s_1, s_4 and s_7 are rate aware bisimilar whenever $\lambda_1 + \lambda_2 = \lambda_3 + \lambda_4 = \lambda_5$.

Notice that *rate aware bisimilarity* and *strong bisimilarity* [19] coincide when one does not take rates into account, i.e. when the range of rate functions is $\{0, 1\}$. The following proposition guarantees that if two processes are *rate aware* equivalent, then the corresponding states in the generated CTMC are *strong Markovian equivalent.*

Proposition 1. *Let $\mathcal{R} = (S, A, \longrightarrow)$ be a* fully stochastic *RTS, for each $A' \subseteq A$ and for each $s_1, s_2 \in S$ and $CTMC[\{s_1, s_2\}, A']$: $s_1 \sim s_2 \Longrightarrow s_1 \sim_M s_2$*

Notice that the reverse is not true. For example, if one considers RTS with states $\{s_i | 1 \leq i \leq 6\}$ where $s_1 \xrightarrow{\alpha} [s_3 \mapsto \lambda_1]$, $s_1 \xrightarrow{\gamma} [s_2 \mapsto \lambda_2]$, $s_4 \xrightarrow{\beta} [s_5 \mapsto \lambda_5]$, and $s_4 \xrightarrow{\alpha} [s_5 \mapsto \lambda_1]$, , states s_1 and s_4 are Markovian equivalent in the $CTMC[\{s_1, s_4\}, \{\alpha\}]$, which does not contain states s_2 and s_5, but $s_1 \not\sim s_4$.

3 PEPA: A Process Algebra for Performance Evaluation

The first process algebra we take into account is the Performance Evaluation Process Algebra (PEPA) developed by Hillston [14]. This algebra enriches CSP [16] with combinators useful for modeling performance related features.

Like in CSP, in PEPA systems are described as interactions of *components* that may engage in *activities*. Components reflect the behaviour of relevant parts of the system, while activities capture the actions that the components perform. The specification of a PEPA activity consists of a pair (α, λ) in which *action* α symbolically denotes the performed action, while *rate* λ characterises the negative *exponential* distribution of its duration.

If $\mathcal{A}$ is a set of *actions*, ranged over by $\alpha, \alpha', \alpha_1, \ldots$, then $\mathcal{P}_{PEPA}$ is the set of process terms $P, P', P_1, \ldots$ defined according to the following grammar

$$P ::= (\alpha, \lambda).P \mid P + P \mid P \bowtie_L P \mid P/L \mid A$$

where λ is a positive real number, L is a subset of $\mathcal{A}$ and A is a *constant* which is assumed defined by an appropriate equation $A \overset{\triangle}{=} P$ for some process term P, where *constants* occur only guarded in P, i.e. under the scope of a action prefix.

Component $(\alpha, \lambda).P$ models a process that perform action α and then behaves like P. The action duration is determined by a random variable exponentially distributed with rate λ.

Component $P + Q$ models a system that may behave either as P or as Q, representing a *race condition* between components. The cooperation operator $P \bowtie_L Q$ defines the set of action types L on which components P and Q must synchronise (or *cooperate*); both components proceed independently with any activity not occurring in L. The expected duration of a cooperation of activities $\alpha \in L$ is a function of the expected durations of the corresponding activities in the components. Roughly speaking, it corresponds to the longest one (the actual definition can be found in [14], where the interested reader can find all formal details of PEPA). Components P/L behaves as P except that activities in L are hidden and appearing as τ transitions. The behaviour of process variable A is that of P, provided that a definition $A \overset{\triangle}{=} P$ is available for A.

We now provide the stochastic semantics of PEPA in terms of RTS. To this aim, we consider the RTS $\mathcal{R}_{PEPA} = (\mathcal{P}_{PEPA}, \mathcal{A}, \longrightarrow)$ where $\longrightarrow$ is formally defined in Fig. 1. These rules permit deriving with a single proof *all* possible configurations reachable from a process with a given transition label.

Rule (Act) states that $(\alpha, \lambda).P$ evolves with α to $[P \mapsto \lambda]$ (see Notation 1). Rule ($\emptyset$-Act) states that no process is reachable from $(\alpha, \lambda).P$ by performing activity $\beta \neq \alpha$.

Rule (Sum) permits modeling stochastic behaviors of non deterministic choice. This rule states that the states reachable from $P+Q$ via α are all those that can be reached either by P or by Q. Moreover, transition rates are determined by summing local rates of transitions occurring either in P or in Q. Indeed, $\mathscr{P} + \mathscr{Q}$ denotes the next state function $\mathscr{R}$ such that: $\mathscr{R}(R) = \mathscr{P}(R) + \mathscr{Q}(R)$.

$$\frac{}{(\alpha,\lambda).P \xrightarrow{\alpha} [P \mapsto \lambda]}\ (\textsc{Act}) \qquad \frac{\alpha \neq \beta}{(\alpha,\lambda).P \xrightarrow{\beta} \emptyset}\ (\emptyset\text{-}\textsc{Act})$$

$$\frac{P \xrightarrow{\alpha} \mathscr{P} \quad Q \xrightarrow{\alpha} \mathscr{Q}}{P + Q \xrightarrow{\alpha} \mathscr{P} + \mathscr{Q}}\ (\textsc{Sum}) \qquad \frac{P \xrightarrow{\alpha} \mathscr{P} \quad Q \xrightarrow{\alpha} \mathscr{Q} \quad \alpha \notin L}{P \bowtie_L Q \xrightarrow{\alpha} \mathscr{P} \bowtie_L Q + P \bowtie_L \mathscr{Q}}\ (\textsc{Int})$$

$$\frac{P \xrightarrow{\alpha} \mathscr{P} \quad Q \xrightarrow{\alpha} \mathscr{Q} \quad \alpha \in L}{P \bowtie_L Q \xrightarrow{\alpha} \mathscr{P} \bowtie_L \mathscr{Q} \cdot \frac{\min\{\oplus\mathscr{P}, \oplus\mathscr{Q}\}}{\oplus\mathscr{P} \cdot \oplus\mathscr{Q}}}\ (\textsc{Coop})$$

$$\frac{P \xrightarrow{\alpha} \mathscr{P} \quad \alpha \notin L}{P/L \xrightarrow{\alpha} \mathscr{P}/L}\ (\text{P-}\textsc{Hide}) \qquad \frac{\alpha \in L}{P/L \xrightarrow{\alpha} \emptyset}\ (\emptyset\text{-}\textsc{Hide})$$

$$\frac{P \xrightarrow{\tau} \mathscr{P}_\tau \quad \forall \alpha \in L. P \xrightarrow{\alpha} \mathscr{P}_\alpha}{P/L \xrightarrow{\tau} \mathscr{P}_\tau/L + \sum_{\alpha \in L} \mathscr{P}_\alpha/L}\ (\textsc{Hide}) \qquad \frac{P \xrightarrow{\alpha} \mathscr{P} \quad A \stackrel{\triangle}{=} P}{A \xrightarrow{\alpha} \mathscr{P}}\ (\textsc{Call})$$

Fig. 1. PEPA Operational Semantics Rules

Rules (Int) and (Coop) govern cooperation. Rule (Int) states that if $\alpha \notin L$ computations of $P \bowtie_L Q$ are obtained by considering the interleaving of the transitions of P and Q. Hence, if we let $\mathscr{P}$ and $\mathscr{Q}$ be the next state functions of P and Q after α ($\alpha \notin L$), the next state function of $P \bowtie_L Q$ after α is obtained by combining $\mathscr{P} \bowtie_L Q$ and $P \bowtie_L \mathscr{Q}$, i.e. the next state function of P, composed with Q, and the next state function of Q, composed with P, respectively, as defined below.

Notation 2. *For next state function $\mathscr{P}$, process algebra operator op and process Q we let $\mathscr{P}$ op Q (resp. Q op $\mathscr{P}$, op $\mathscr{P}$) be the function $\mathscr{R}$ such that $\mathscr{R}(R)$ is $\mathscr{P}(P)$ if $R = P$ op Q (resp. Q op P, op P) and* 0 *otherwise.*

Rule (Coop) is used for computing the next state function when a synchronization between P and Q occurs. In that case, the next state function of $P \bowtie_L Q$ is determined as $\mathscr{P} \bowtie_L \mathscr{Q}$, as defined below.

Notation 3. *For next state functions $\mathscr{P}$, $\mathscr{Q}$ and set $L \subseteq \mathcal{A}$, $\mathscr{P} \bowtie_L \mathscr{Q}$ is the function such that $\mathscr{P} \bowtie_L \mathscr{Q}(R)$ is $\mathscr{P}(P) \cdot \mathscr{Q}(Q)$ if $R = P \bowtie_L Q$,* 0 *otherwise.*

As described in [14], actual rates in $\mathscr{P} \bowtie_L \mathscr{Q}$ are multiplied by the minimum of the apparent rate of α in P and Q and divided by their product.

Notation 4. *For next state functions $\mathscr{P}$, and $x, y \in \mathbb{R}_{\geq 0}$ $\mathscr{P} \cdot \frac{x}{y}$ is the function $\mathscr{R}$ such that $\mathscr{R}(R) = \mathscr{P}(R) \cdot \frac{x}{y}$ if $y \neq 0$, $\emptyset$ otherwise.*

The apparent rates of α in a process P is defined as the total capacity of P to carry out activities of type α. In [14], the apparent rate of α in a process P is computed by using an auxiliary function $r_\alpha(P)$. By using our RTS approach, if $P \xrightarrow{\alpha} \mathscr{P}$, then the apparent rate of α in P is determined as: $\oplus\mathscr{P} = \sum_Q \mathscr{P}(Q)$.

Rule (P-Hide) states that the set of processes reachable from P/L with α is determined by the set of processes reachable from P with α. Rule ($\emptyset$-Hide) states that no process is reachable from P/L with $\alpha \in L$. Rule (Hide) states that the set of processes reachable from P/L with a τ is determined by the set of processes reachable from P with τ and by considering, for each α in L, the set of processes reachable from P with α.

Notice that $\forall \alpha \in L.P \xrightarrow{\alpha} \mathscr{P}_\alpha$ in the premises of rule (Hide) denotes that to prove a transition one has to prove a transition for each $\alpha \in L$. Theorem 1 below guarantees the finiteness of the proposed semantics.

Theorem 1. *$\mathcal{R}_{PEPA}$ is* fully stochastic *and* image finite.

In the sequel by $\xrightarrow{}_{PEPA}$ we mean the transition relation defined in [14].

Theorem 2. *For all $P, Q \in \mathcal{P}_{PEPA}$ and $\alpha \in \mathcal{A}$ the following holds: $P \xrightarrow{\alpha} \mathscr{P} \wedge \mathscr{P}(Q) = \lambda > 0$ if and only if $P \xrightarrow{\alpha,\lambda}_{PEPA} Q$.*

The RTS associated to PEPA processes can be used for associating to each process P a CTMC. This is obtained by considering $CTMC[\{P\}, \mathcal{A}]$ where $\mathcal{A}$ is the set of all activities that process P can perform.

4 Stochastic CCS

The second stochastic process algebra we consider in this paper is a stochastic extension of the Calculus of Communicating System (CCS) [19]. Differently from CSP, where processes composed in parallel *cooperate* in a *multi-party* synchronization, in CCS parallel processes interact with each other by means of a *two-party* synchronisation.

In Stochastic CCS (StoCCS), *output actions* are equipped with a parameter (a *rate*, $\lambda \in \mathbb{R}^+$) characterising a random variable with a negative exponential distribution, modeling the duration of the action. *Input actions* are annotated with a *weight* ($\omega \in \mathbb{N}^+$): a positive integer that will be used for determining the probability that the specific input is selected when a complementary output is executed. This approach is inspired by the *passive actions* presented in [14].

Let $\mathcal{C}$ be a set of channels ranged over by $a, b, c, \ldots$, $\overline{\mathcal{C}}$ denotes the *co-names* of $\mathcal{C}$. Elements in $\overline{\mathcal{C}}$ are ranged over by $\overline{a}, \overline{b}, \overline{c}, \ldots$. A synchronization between processes P and Q occurs when P *sends* a signal over channel (action $\overline{a}$) while Q *receives* a signal over the same channel (action a). The result of a synchronization is an *internal*, or *silent*, transition that is labeled τ. In StoCCS a synchronization over channel a is rendered by the label $\overleftrightarrow{a}$. The reasons for this choice will be clarified later. We let $\overleftrightarrow{\mathcal{C}}$ be $\{\overleftrightarrow{a} | a \in \mathcal{C}\}$. The set of labels $\mathcal{L}$ is then $\mathcal{C} \cup \overline{\mathcal{C}} \cup \{\tau\} \cup \overleftrightarrow{\mathcal{C}}$, while its elements are ranged over by $\ell, \ell', \ell_1, \ldots$.

$\mathcal{P}_{CCS}$ is the set of *Stochastic CCS* process terms $P, P', P_1, Q, Q', Q_1 \ldots$ defined according to the following grammar:

$$P, Q \ ::= \ \mathbf{0} \mid G \mid P|Q \mid P[f] \mid P\backslash L \mid A \qquad G \ ::= \ a^{\omega}.P \mid \overline{a}^{\lambda}.P \mid G + G$$

where $L \subseteq \mathcal{C}$ while f is a *renaming function*, i.e. a function in $\mathcal{L} \rightarrow \mathcal{L}$ such that $f(\overline{a}) = \overline{f(a)}$, $f(\overleftrightarrow{a}) = \overleftrightarrow{f(a)}$ and $f(\tau) = \tau$. A is a *constant* which is assumed being

$$\frac{}{a^{\omega}.P \xrightarrow{a} [P \mapsto \omega]}\ (\textsc{In}) \qquad \frac{\ell \neq a}{a^{\omega}.P \xrightarrow{\ell} \emptyset}\ (\emptyset\text{-}\textsc{In})$$

$$\frac{}{\overline{a}^{\lambda}.P \xrightarrow{\overline{a}} [P \mapsto \lambda]}\ (\textsc{Out}) \qquad \frac{\ell \neq \overline{a}}{\overline{a}^{\lambda}.P \xrightarrow{\ell} \emptyset}\ (\emptyset\text{-}\textsc{Out})$$

$$\frac{P \xrightarrow{\ell} \mathscr{P} \quad Q \xrightarrow{\ell} \mathscr{Q}}{P + Q \xrightarrow{\ell} \mathscr{P} + \mathscr{Q}}\ (\textsc{Sum}) \qquad \frac{P \xrightarrow{\ell} \mathscr{P} \quad Q \xrightarrow{\ell} \mathscr{Q} \quad \ell \neq \overleftrightarrow{a}}{P|Q \xrightarrow{\ell} \mathscr{P}|Q + P|\mathscr{Q}}\ (\textsc{Int})$$

$$\frac{P \xrightarrow{\overleftrightarrow{a}} \mathscr{P} \quad P \xrightarrow{a} \mathscr{P}_i \quad P \xrightarrow{\overline{a}} \mathscr{P}_o \quad Q \xrightarrow{\overleftrightarrow{a}} \mathscr{Q} \quad Q \xrightarrow{a} \mathscr{Q}_i \quad Q \xrightarrow{\overline{a}} \mathscr{Q}_o}{P|Q \xrightarrow{\overleftrightarrow{a}} \mathscr{P}|Q + P|\mathscr{Q} + \frac{\mathscr{P}_i \cdot \mathscr{Q}_o}{\oplus \mathscr{P}_i} + \frac{\mathscr{P}_o \cdot \mathscr{Q}_i}{\oplus \mathscr{Q}_i}}\ (\textsc{Sync})$$

$$\frac{}{\mathbf{0} \xrightarrow{\ell} \emptyset}\ (\textsc{Nil}) \qquad \frac{\ell \in L}{P\backslash L \xrightarrow{\ell} \emptyset}\ (\emptyset\text{-}\textsc{Res})$$

$$\frac{P \xrightarrow{\ell} \mathscr{P} \quad \ell \notin L}{P\backslash L \xrightarrow{\ell} \mathscr{P}\backslash L}\ (\text{P-}\textsc{Res}) \qquad \frac{P \xrightarrow{\tau} \mathscr{P}_{\tau} \quad \forall \ell \in L.P \xrightarrow{\overleftrightarrow{\ell}} \mathscr{P}_{\overleftrightarrow{\ell}}}{P\backslash L \xrightarrow{\tau} \mathscr{P}_{\tau}\backslash L + \sum_{\ell \in L} \mathscr{P}_{\overleftrightarrow{\ell}}\backslash L}\ (\textsc{Res})$$

$$\frac{\forall \ell : P \xrightarrow{\ell} \mathscr{P}_{\ell}}{P \xrightarrow{\beta} \sum_{\ell : f(\ell) = \beta} \mathscr{P}_{\ell}[f]}\ (\textsc{Ren}) \qquad \frac{A \stackrel{\triangle}{=} P \quad P \xrightarrow{\ell} \mathscr{P}}{A \xrightarrow{\ell} \mathscr{P}}\ (\textsc{Call})$$

Fig. 2. StoCCS Operational Semantics

defined by a proper defining equation $A \stackrel{\triangle}{=} P$ for some process term P, where each constant can occur only guarded in P. For the sake of notational simplicity, we assume that each process G never contains at the same time an input and an output action on the same channel. In other words, processes of the form $\overline{a}.P + a.Q$ are forbidden. This does not introduce a significant restriction because such processes do not have an obvious meaning in the context of stochastic process algebras.

Action prefixing and non-deterministic choice have the same meaning as in PEPA. Process $P|Q$ models a system where P and Q proceed in parallel and interact with each other using the two-parties synchronisation described above. Restriction ($P\backslash L$) and renaming ($P[f]$) are respectively used for inhibiting interactions of P over channels in L and for renaming channels in P according to function f.

Following a similar approach as the one used for PEPA, we now define the stochastic semantics of StoCCS in term of RTS. We let $\mathcal{R}_{StoCCS} = (\mathcal{P}_{CCS}, \mathcal{L}, \longrightarrow)$, where $\longrightarrow$ is formally defined in Fig. 2.

The proposed semantics follows the same approach used by Priami in [22] for the stochastic π-calculus and makes use of the PEPA notions of active and passive actions. All the rules have the expected meaning and are similar to those defined for PEPA and simply render the CCS semantics in a context where all the possible next processes are computed in a single derivation.

More attention has to be paid to rule (Sync) that is used for deriving synchronisations of parallel processes. In PEPA we have *multi-party* synchronisations.

Hence, the next states of $P \bowtie_L Q$ after $\ell \in L$ can be simply obtained by combining the possible next states of P and Q after ℓ. In CCS we have two-party synchronisations, thus the next states of $P|Q$ after $\overleftrightarrow{a}$, i.e. after a synchronisation over channel a, are: (1) the next states of P alone after $\overleftrightarrow{a}$, in parallel with Q; (2) the next states of Q alone after $\overleftrightarrow{a}$, in parallel with P; (3) the next states of P after $\overline{a}$ in parallel with the next states of Q after a; (4) the next states of P after a in parallel with the next states of Q after $\overline{a}$. Moreover, synchronisation rates between inputs in P and outputs in Q (and vice-versa) are obtained by multiplying the input weights of P, i.e. $\mathscr{P}_i$, by the output rates of Q, i.e. $\mathscr{Q}_o$, over the total weight of all the inputs in P, i.e. $\oplus\mathscr{P}_i$ (and vice-versa). As an example, consider $P \triangleq \overline{a}^2.P_1$ and $Q \triangleq a^4.Q_1|a^2.Q_2$, then we have that $\overleftrightarrow{a}$ leads process $P|Q$ to $P_1|(Q_1|a^2.Q_2)$ with rate $\frac{4}{3}$ and to $P_1|(a^4.Q_1|Q_2)$ with rate $\frac{2}{3}$.

Theorem 3. *$\mathcal{R}_{StoCCS}$ is* fully stochastic *and* image finite*.*

It is easy to prove that the stochastic semantics of Fig. 2 coincides with the one proposed in [18]. Unfortunately, the proposed semantics, like in [18], does not respect a standard and expected property of the CCS parallel composition. Indeed, using the above semantics, this operator is not associative. For instance $\overline{a}^\lambda.P|(a^{\omega_1}.Q_1|a^{\omega_2}.Q_2)$ and $(\overline{a}^\lambda.P|a^{\omega_1}.Q_1)|a^{\omega_2}.Q_2$ exhibit different stochastic behaviours. The former, after $\overleftrightarrow{a}$, reaches $P|(Q_1|a^{\omega_2}.Q_2)$ with rate $\frac{\lambda\cdot\omega_1}{\omega_1+\omega_2}$ and $P|(a^{\omega_1}.Q_1|Q_2)$ with rate $\frac{\lambda\cdot\omega_2}{\omega_1+\omega_2}$. The latter reaches both $(P|Q_1)|a^{\omega_2}.Q_2$ and $(P|a^{\omega_1}.Q_1)|Q_2$ with rate λ. From the results in [18] it follows that it is impossible to define an SGSOS semantics that guarantees the associativity of CCS parallel composition. It is moreover worth pointing out that the definition of the semantics for the stochastic π-calculus, as in [22], suffers of the same problem [18]. In the sequel, we show that this problem can be overcome by using our approach. To that purpose we modify rule (Sync) in such way that: the rates of the synchronisations occurring in P and Q are updated in order to take into account the inputs available in both P and Q; the rates of the synchronisations between outputs in P and inputs in Q (and vice-versa) have to be divided by the *total* rate of *input* in both P and Q. Rule (Sync) can be reformulated as follows:

$$\frac{P \xrightarrow{\overleftrightarrow{a}} \mathscr{P} \quad P \xrightarrow{a} \mathscr{P}_i \quad P \xrightarrow{\overline{a}} \mathscr{P}_o \quad Q \xrightarrow{\overleftrightarrow{a}} \mathscr{Q} \quad Q \xrightarrow{a} \mathscr{Q}_i \quad Q \xrightarrow{\overline{a}} \mathscr{Q}_o}{P|Q \xrightarrow{\overleftrightarrow{a}} \frac{\mathscr{P}|Q\cdot\oplus\mathscr{P}_i}{\oplus\mathscr{P}_i+\oplus\mathscr{Q}_i} + \frac{P|\mathscr{Q}\cdot\oplus\mathscr{Q}_i}{\oplus\mathscr{P}_i+\oplus\mathscr{Q}_i} + \frac{\mathscr{P}_i\cdot\mathscr{Q}_o}{\oplus\mathscr{P}_i+\oplus\mathscr{Q}_i} + \frac{\mathscr{P}_o\cdot\mathscr{Q}_i}{\oplus\mathscr{P}_i+\oplus\mathscr{Q}_i}}$$

Using this rule, the associativity of parallel composition, up to *rate aware bisimulation*, is guaranteed. E.g., in the case of $\overline{a}^\lambda.P|(a^{\omega_1}.Q_1|a^{\omega_2}.Q_2)$ and $(\overline{a}^\lambda.P|a^{\omega_1}.Q_1)|a^{\omega_2}.Q_2$, after $\overleftrightarrow{a}$, the following rate functions are reachable:

$$\left[P|(Q_1|a^{\omega_2}.Q_2) \mapsto \frac{\lambda\cdot\omega_1}{\omega_1+\omega_2}\ ,\ P|(a^{\omega_1}.Q_1|Q_2) \mapsto \frac{\lambda\cdot\omega_2}{\omega_1+\omega_2}\right]$$

$$\left[(P|Q_1)|a^{\omega_2}.Q_2 \mapsto \frac{\lambda\cdot\omega_1}{\omega_1+\omega_2}\ ,\ (P|a^{\omega_1}.Q_1)|Q_2 \mapsto \frac{\lambda\cdot\omega_2}{\omega_1+\omega_2}\right]$$

Theorem 4. *In StoCCS parallel composition is associative up to rate aware bisimilarity, i.e. for each P, Q and R, $P|(Q|R) \sim (P|Q)|R$*

Notice that this result is not in contradiction with the one presented in [18] where it is proved that associativity of parallel composition does not hold if one uses PEPA-like synchronisation rates for CCS. Indeed, our result is obtained thanks to the use of a specific explicit label for synchronisation transitions ($\overleftrightarrow{a}$) that in [18] are labelled by τ. Our choice permits updating synchronisation rates while taking into account possible new inputs popping up along the derivation. Notice finally that, it is easy to prove that $\sim$ is a congruence for each operator of StoCCS. The CTMC associated to a StoCCS process P is obtained by considering $CTMC[\{P\}, \overleftrightarrow{\mathcal{C}} \cup \{\tau\}]$.

5 Conclusions

We have introduced a variant of Rate Transition Systems and used them to define the semantics of stochastic extensions of several process algebras among which CSP, CCS and π-calculus [6]. An original feature of this variant is that the transition relation associates to each process, for each action, the set of possible futures paired with a measure indicating their rates. This feature leads to a compact, uniform and elegant definition of the operational semantics. In one case this has also lead to the proposal of an alternative semantics for stochastic CCS that enjoys associativity of the parallel composition operator. We have also introduced a natural notion of bisimulation over RTS that is finer than Markovian bisimulation and useful for reasoning about stochastic behaviours.

Even if in the present paper we have considered a synchronisation mechanism implicitly based on *active* and *passive* actions, other synchronisation patterns proposed in the literature can be easily dealt with as well. For instance, one could associate proper rates both to output and input actions and define the synchronisation rate as a suitable function of such rates. Finally, we have applied our framework also to the definition of stochastic process calculi for service oriented computing [21,3]. Interesting future work includes the further study of the format of the RTS rules aiming at reaching similar general results on bisimulation congruence as in [18].

References

1. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.-P.: Model-Checking Algorithms for Continuous-Time Markov Chains. IEEE Transactions on Software Engineering 29(6), 524–541 (2003)
2. Bernardo, M., Gorrieri, R.: A tutorial on EMPA: A theory of concurrent processes with nondeterminism, priorities, probabilities and time. Theoret. Comput. Sci. 202(1-2), 1–54 (1998)
3. Bravetti, M., Latella, D., Loreti, M., Massink, M., Zavattaro, G.: Combining timed coordination primitives and probabilistic tuple spaces. In: TGC 2008. LNCS, vol. 5474, pp. 52–68. Springer, Heidelberg (2009)

4. Brinksma, E., Hermanns, H.: Process Algebra and Markov Chains. In: Brinksma, E., Hermanns, H., Katoen, J.-P. (eds.) EEF School 2000 and FMPA 2000. LNCS, vol. 2090, pp. 183–231. Springer, Heidelberg (2001)
5. De Nicola, R., Katoen, J.-P., Latella, D., Loreti, M., Massink, M.: Model checking mobile stochastic logic. Theoretical Computer Science 382(1), 42–70 (2007)
6. De Nicola, R., Latella, D., Loreti, M., Massink, M.: Rate-based Transition Systems for Stochastic Process Calculi, http://www.dsi.unifi.it/~loreti/
7. Deng, Y., van Glabbeek, R., Hennessy, M., Morgan, C., Zhang, C.: Characterising testing preorders for finite probabilistic processes. In: Proceedings of LICS 2007, pp. 313–325. IEEE Computer Society, Los Alamitos (2007)
8. Glotz, N., Herzog, U., Rettelbach, M.: Multiprocessor and distributed systems design: The integration of functional specification and performance analysis using stochastic process algebras. In: Donatiello, L., Nelson, R. (eds.) PECCS 1993. LNCS, vol. 729. Springer, Heidelberg (1993)
9. Haverkort, B.R.: Markovian Models for Performance and Dependability Evaluation. In: Brinksma, E., Hermanns, H., Katoen, J.-P. (eds.) EEF School 2000 and FMPA 2000. LNCS, vol. 2090, pp. 38–83. Springer, Heidelberg (2001)
10. Hermanns, H.: Interactive Markov Chains. In: Hermanns, H. (ed.) Interactive Markov Chains. LNCS, vol. 2428, pp. 57–88. Springer, Heidelberg (2002)
11. Hermanns, H., Herzog, U., Katoen, J.-P.: Process algebra for performance evaluation. Theoret. Comput. Sci. 274(1-2), 43–87 (2002)
12. Hermanns, H., Johr, S.: Uniformity by Construction in the Analysis of Nondeterministic Stochastic Systems. In: DSN 2007, pp. 718–728. IEEE Computer Society Press, Los Alamitos (2007)
13. Hermanns, H., Siegle, M.: Bisimulation algorithms for stochastic process algebras and their BDD-based implementation. In: Katoen, J.-P. (ed.) AMAST-ARTS 1999, ARTS 1999, and AMAST-WS 1999. LNCS, vol. 1601, pp. 244–264. Springer, Heidelberg (1999)
14. Hillston, J.: A compositional approach to performance modelling. Distinguished Dissertation in Computer Science, Cambridge University Press, Cambridge (1996)
15. Hillston, J.: Process algebras for quantitative analysis. In: IEEE Symposium on Logic in Computer Science, pp. 239–248. IEEE Computer Society Press, Los Alamitos (2005)
16. Hoare, C.: Communicating Sequential Processes. Prentice-Hall, Englewood Cliffs (1985)
17. Kemeny, J., Snell, J.: Finite Markov Chains. Springer, Heidelberg (1976)
18. Klin, B., Sassone, V.: Structural operational semantics for stochastic process calculi. In: Lipp, P., Sadeghi, A.-R., Koch, K.-M. (eds.) Trust 2008. LNCS, vol. 4968. Springer, Heidelberg (2008)
19. Milner, R.: Communication and Concurrency. Prentice-Hall, Englewood Cliffs (1989)
20. Milner, R., Parrow, J., Walker, J.: A Calculus of Mobile Processes, I and II. Information and Computation 100(1), 1–77 (1992)
21. Nicola, R.D., Latella, D., Loreti, M., Massink, M.: MarCaSPiS: a markovian extension of a calculus for services. In: Proc. of SOS 2008. ENTCS, Elsevier, Amsterdam (2008)
22. Priami, C.: Stochastic π-Calculus. The Computer Journal 38(7), 578–589 (1995)
23. Puterman, M.: Markov Decision Processes (1994)

Improved Algorithms for Latency Minimization in Wireless Networks[*]

Alexander Fanghänel, Thomas Keßelheim, and Berthold Vöcking

Department of Computer Science, RWTH Aachen University, Germany

Abstract. In the *interference scheduling problem*, one is given a set of n communication requests described by source-destination pairs of nodes from a metric space. The nodes correspond to devices in a wireless network. Each pair must be assigned a power level and a color such that the pairs in each color class can communicate simultaneously at the specified power levels. The feasibility of simultaneous communication within a color class is defined in terms of the Signal to Interference plus Noise Ratio (SINR) that compares the strength of a signal at a receiver to the sum of the strengths of other signals. The objective is to minimize the number of colors as this corresponds to the time needed to schedule all requests.

We introduce an instance-based measure of interference, denoted by I, that enables us to improve on previous results for the interference scheduling problem. We prove upper and lower bounds in terms of I on the number of steps needed for scheduling a set of requests. For general power assignments, we prove a lower bound of $\Omega(I/(\log \Delta \log n))$ steps, where Δ denotes the aspect ratio of the metric. When restricting to the two-dimensional Euclidean space (as previous work) the bound improves to $\Omega(I/\log \Delta)$. Alternatively, when restricting to linear power assignments, the lower bound improves even to $\Omega(I)$. The lower bounds are complemented by an efficient algorithm computing a schedule for linear power assignments using only $\mathcal{O}(I \log n)$ steps. A more sophisticated algorithm computes a schedule using even only $\mathcal{O}(I + \log^2 n)$ steps. For dense instances in the two-dimensional Euclidean space, this gives a constant factor approximation for scheduling under linear power assignments, which shows that the price for using linear (and, hence, energy-efficient) power assignments is bounded by a factor of $\mathcal{O}(\log \Delta)$.

In addition, we extend these results for single-hop scheduling to multi-hop scheduling and combined scheduling and routing problems, where our analysis generalizes previous results towards general metrics and improves on the previous approximation factors.

1 Introduction

The media access control (MAC) layer of wireless networks is responsible for scheduling signals taking into account interference caused by concurrent transmissions. Early algorithmic studies of this task were based on graph theoretical

[*] This work has been supported by the UMIC Research Centre, RWTH Aachen University.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 447–458, 2009.

vicinity models (see, e.g., [10,17,8]). In more recent literature, these studies have been critized to not model interference appropriately as they assume that the interference caused by signals ends abruptly at some boudary (see, e.g.,[14,15,4,5]).

Like the other recent studies mentioned above, we describe interference using the so-called *physical model* in which it is assumed that the strength of a signal fades with the distance from the sender. This fading is described by a *path loss exponent* $\alpha \geq 1$.[1] The strength of a signal sent with some power p received by a node (transceiver) v at distance d from the source of the signal is assumed to be p/d^α. The node v can successfully receive the signal if its strength is sufficiently large in comparison to the sum of other signals that are sent simultaneously plus ambient noise, that is, if the *signal to interference plus noise ratio (SINR)* is above some threshold $\beta > 1$, the so-called *gain*.

The *interference scheduling problem* is formally defined as follows. Let V be a set of nodes from a metric space. Let $d(u, v)$ denote the distance between two nodes u and v. One is given a set $\mathcal{R}$ of n requests consisting of pairs $(u_i, v_i) \in V^2$, where u_i is the source and v_i the destination of the signal from the i-th request. For every $i \in [n] := \{1, \dots, n\}$, one needs to specify a power level $p_i > 0$ and a color $c_i \in [k] := \{1, \dots, k\}$ such that the *latency*, i. e., the number of colors, k, is minimized and the pairs in each color class satisfy the SINR constraints for all signals: For every $i \in [n]$, it must hold that

$$\frac{p_i}{d(u_i, v_i)^\alpha} \geq \beta \left(\sum_{\substack{j \in [n] \setminus \{i\} \\ c_j = c_i}} \frac{p_j}{d(u_j, v_i)^\alpha} + \nu \right) ,$$

where $\nu \geq 0$ expresses ambient noise. The so-called *scheduling complexity* of $\mathcal{R}$, as introduced by Moscribroda and Wattenhofer [14], is the minimal number of colors (steps) needed to schedule the requests in $\mathcal{R}$.

In this work, we mostly focus on *linear power assignments*, i.e., for a request pair (u_i, v_i) the power is proportional to $d(u_i, v_i)^\alpha$ and, hence, linear in fading. Linear power schemes also have been considered in [2,19,4]. Our analysis will show, that one loses only a factor of order $\log \Delta$ due to restricting to this power scheme (where the aspect ratio Δ denotes the ratio between the longest and shortest distance between any two nodes). Let us remark that the dependence on the aspect ratio Δ cannot be avoided using the linear power assignment which, without taking into account other parameters than n, cannot achieve an approximation ratio better than $\Omega(n)$ [14,6]. Besides leading to good performance results, linear power assignments have the advantage being energy-efficient as the minimal transmission power required to transmit along a distance d is $\Theta(d^\alpha)$.

1.1 Our Contribution

We introduce an instance-based measure of interference that enables us to estimate the scheduling complexity of any set of requests within small factors.

[1] It is usually assumed, that α satisfies $2 < \alpha < 5$. Our analysis holds for any constant $\alpha \geq 1$, unless stated otherwise.

Definition 1 (Measure of Interference). *Let $\mathcal{R} \subseteq V \times V$ be a set of requests. For $w \in V$ define*

$$I_w(\mathcal{R}) = \sum_{(u,v)\in\mathcal{R}} \min\left\{1, \frac{d(u,v)^\alpha}{d(u,w)^\alpha}\right\} \ .$$

Using this we define the measure of interference induced by the requests $\mathcal{R}$:

$$I = I(\mathcal{R}) = \max_{w\in V} I_w(\mathcal{R}) \ .$$

We prove upper and lower bounds on the number of steps needed for scheduling $\mathcal{R}$ in terms of I. For general power assignments and general metrics, we prove a lower bound of $\Omega(I/\log\Delta\log n)$ steps. When restricting to the two-dimensional Euclidean space and assuming $\alpha > 2$ the bound improves to $\Omega(I/\log\Delta)$. Alternatively, when restricting to linear power assignments and assuming general metrics, this bound improves even to $\Omega(I)$. The lower bounds are complemented by an efficient algorithm computing a schedule for linear power assignments using only $\mathcal{O}(I\log n)$ steps. A more sophisticated algorithm computes a schedule using even only $\mathcal{O}(I + \log^2 n)$ steps. This gives a constant factor approximation of the optimal schedule under linear power assignments for *dense* instances, i.e., if $I \geq \log^2 n$. Combining this upper bound for linear power assignments with the lower bound for general power assignments and the two-dimensional Euclidean space shows that the price for using linear, in other words, energy-efficient power assignments is of order $\mathcal{O}(\log\Delta)$.

We further extend our results towards multi-hop scheduling and routing. In the multi-hop scheduling problem, a request is defined by a sequence of pairs, so-called *paths*, rather than a single pair of nodes. Along each of these paths, one should forward a signal from the first to the last node on the path. Let D denote the maximum number of hops on each of these paths, the so-called *dilation*. Generalizing, the lower bounds from the single-hop to the multi-hop problem, shows that one needs at least $\Omega(I/\log\Delta\log n + D)$ steps, for general power assignments, $\Omega(I/\log\Delta + D)$ for the Euclidean space, and $\Omega(I + D)$ steps, for linear power assignments. We show how to extend our second algorithm for the single-hop scheduling to the multi-hop case, where it produces a schedule of at most $\mathcal{O}(I + D\cdot\log^2 n)$ steps.

Our results for multi-hop scheduling reminds of the $\mathcal{O}(\text{congestion} + \text{dilation})$-type results that have been shown previously for routing in wired networks, see, e.g. [11,12,1,18]. In fact, this previous work was the inspiration to search for an instance-based density measure that allows to derive lower bounds for the scheduling complexity in wireless networks like the congestion in wired networks. At this point, let us remark that, unlike the congestion, our interference measure I does not trivially give a lower bound on the number of steps needed for scheduling a set of requests but it requires a careful analysis as also the upper bound does.

Finally, we extend our result to combined multi-hop routing and scheduling. Now requests are again defined by pairs of nodes. The problem is to find source-destination paths for all requests and to compute a power assignment and a schedule delivering all packets using as few steps as possible. Combining our multi-hop scheduling algorithm with a linear programming approach for computing paths that minimize the term $\max\{I, D\}$ gives an $\mathcal{O}(\log \Delta \log^3 n)$-approximation for the combined routing and scheduling problem in general metrics. In the two-dimensional Euclidean space the approximation factor is $\mathcal{O}(\log \Delta \log^2 n)$. This generalizes the results from Chafekar et al. [4] (cf. Section 1.2) towards general metrics and improves on their approximation factors.

1.2 Related Work

The first theoretical studies about interference scheduling in the physical model focus on topologies generated by placing nodes randomly in two-dimensional Euclidean space, see, e.g., [7,3,9].

The study of interference scheduling with respect to arbitrary topologies has been initiated by Moscibroda and Wattenhofer [14]. They present the first analysis of the interference scheduling problem. However, they do not handle general request sets but only specific kinds of sets.

This result has been extended by Moscibroda et al. [15] to arbitrary demands. Their result is an $\mathcal{O}(\log^2 n \cdot I_{\text{in}})$ algorithm, where I_{in} is a certain interference measure. This result enables them to improve the bound for strong connectivity from $\mathcal{O}(\log^4 n)$ to $\mathcal{O}(\log^3 n)$. Unfortunately, I_{in} is no lower bound for the optimal schedule length. Thus, it does not give any approximation guarantee for general request sets since there is no comparison between I_{in} and the optimal schedule length.

In [13], another measure of interference χ_ρ called disturbance is introduced where $\rho > 0$ is a parameter. The algorithm described achieves a schedule length of $\mathcal{O}(\chi_\rho \rho^2 \log n \cdot (\log n + \rho))$. Unfortunately, also this result does not yield a comparison to the optimal schedule length.

Fanghänel et al. [6] deal with directed and undirected request sets. For the directed case they extend the results of Moscibroda and Wattenhofer by showing that any power assignment that is *oblivous*, i. e., the transmission power is based only on the distance between the sender and the receiver, cannot be bounded in an useful manner without taking into account metric properties like the aspect ratio Δ. For the undirected case they prove the *square-root power assignment* to be an $\mathcal{O}(\log^{3.5+\alpha} n)$-approximation. However, neither is this power scheme energy efficient, nor can their constructive results be generalized towards the multi-hop case with standard techniques, as there is no measure of interference given that is a lower bound for the optimal schedule.

Chafekar et al. [4] study the combined routing and multi-hop version of the interference scheduling problem. It is crucial for their analysis to deal with two-dimensional Euclidean instances and $\alpha > 2$. This allows to use graph coloring in a similar way to the approaches used in the graph-theoretical vicinity models. Our approach instead works in general metrics taking the non-locality of the SINR constraint into account. In their analysis the considered power assignment is restricted, that is, it is assumed that power levels must be chosen from a specified interval $[p_{\min}, p_{\max}]$. It yields a schedule using $\mathcal{O}(\text{opt}' \cdot \log^2 n \log \Delta \log^2 \Gamma)$ time slots where opt$'$ denotes the minimal number of time slots needed for a schedule with slightly smaller power range $[p_{\min}, (1-\epsilon)p_{\max}]$ and Γ denotes the ratio between $p_{\max}$ and $p_{\min}$.

2 Introducing a Measure of Interference

In this section we justify the choice of our measure of interference, as it yields lower bounds for the optimal schedule length under both arbitrary and linear power assignments. In Section 2.1 we show, that the length T of an optimal schedule using a linear power assignment is lower bounded by $\Omega(I)$. In Section 2.2 we show, that a lower bound for the length of an optimal schedule under an arbitrary power assignment is $\Omega(I/\log \Delta \cdot \log n)$ in general metrics and $\Omega(I/\log \Delta)$ in the two-dimensional Euclidean space for $\alpha > 2$.

2.1 A Comparison to the Optimal Schedule Using Linear Power Assignments

Theorem 1. *Let T be the minimum schedule length for a set of requests $\mathcal{R}$ in a linear power assignment. Then we have $T = \Omega(I)$.*

Proof. Let there be a schedule of length T when using a linear power assignment. Then there exist sets of requests $\mathcal{R}_1, \ldots, \mathcal{R}_T$ each of which satisfies the SINR constraint for the linear power assignment. I is subadditive, i. e., we have $I\left(\bigcup_{t=1}^{T} \mathcal{R}_t\right) \leq \sum_{t=1}^{T} I\left(\mathcal{R}_t\right)$. Thus it suffices to show that $I(\mathcal{R}_t) = \mathcal{O}(1)$ for such a set.

Let $\mathcal{R}_t = \{(u_1, v_1), \ldots, (u_{\bar{n}}, v_{\bar{n}})\}$. Let furthermore be $w \in V$. The node w does not necessarily act as a receiver v_i in this request set $\mathcal{R}_t$. This is why we define v_j as the closest (active) receiver from w, i. e. $j \in \arg\min_{i \in [\bar{n}]} d(v_i, w)$. This node might also be w itself.

To bound the measure of interference, we distinguish between two kinds of requests. We define a set U of indices of requests whose senders u_i lie within a distance of at most $\frac{1}{2}d(v_j, w)$ from w, i. e. $U = \{i \in [\bar{n}] \mid d(u_i, w) \leq \frac{1}{2}d(v_j, w)\}$. Using the triangle inequality we can conclude for all $i \in U$:

$$d(u_i, v_j) \leq d(u_i, w) + d(w, v_j) \leq \frac{3}{2}d(v_j, w) \ . \tag{1}$$

In addition, we have

$$\begin{aligned} d(v_j, w) &\leq d(v_i, w) && \text{since } v_j \text{ is the closest receiver} \\ &\leq d(v_i, u_i) + d(u_i, w) && \text{by triangle inequality} \\ &\leq d(v_i, u_i) + \frac{1}{2}d(v_j, w) && \text{by definition of } U \ . \end{aligned}$$

This implies

$$d(v_j, w) \leq 2d(u_i, v_i) \ . \tag{2}$$

Combining Equation 1 and Equation 2 we get $d(u_i, v_j) \leq 3d(u_i, v_i)$. Thus it holds

$$|U \setminus \{j\}| = \sum_{\substack{i \in U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{d(u_i, v_i)^\alpha} \leq \sum_{\substack{i \in U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{\frac{1}{3^\alpha} d(u_i, v_j)^\alpha} \leq \frac{3^\alpha}{\beta} \ .$$

For all $i \in [\bar{n}] \setminus U$ it holds that

$$\begin{aligned} d(u_i, v_j) &\leq d(u_i, w) + d(w, v_j) && \text{by triangle inequality} \\ &\leq d(u_i, w) + 2d(u_i, w) && \text{by definition of } U \\ &= 3d(u_i, w) \ . \end{aligned}$$

Now, we can sum up all $i \in [\bar{n}] \setminus U$:

$$\sum_{\substack{i \in [\bar{n}] \setminus U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{d(u_i, w)^\alpha} \leq \sum_{\substack{i \in [\bar{n}] \setminus U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{\frac{1}{3^\alpha} d(u_i, v_j)^\alpha} \leq \frac{3^\alpha}{\beta} \ .$$

Summing up all $i \in [\bar{n}]$ gives

$$I_w(\mathcal{R}_t) \leq |U \setminus \{j\}| + \sum_{\substack{i \in [\bar{n}] \setminus U \\ i \neq j}} \frac{d(u_i, v_i)^\alpha}{d(u_i, w)^\alpha} + 1 \leq \frac{2 \cdot 3^\alpha}{\beta} + 1 = \mathcal{O}(1) \ .$$

□

2.2 A Comparison to the Optimal Schedule

Using similar arguments, we can also prove bounds on the optimal schedule length using arbitrary power assignments. Due to space limitations the proofs are omitted and can be found in the full version.

Theorem 2. *Let T denote the optimal schedule length using any power assignment. Then we have $T = \Omega\left({}^{I}/_{\log \Delta \cdot \log n}\right)$.*

In previous work, the instances often are restricted to the Euclidean plane and α is required to be strictly greater than 2. Under these assumptions we can use geometric arguments to get an even better bound of $\Omega({}^{I}/_{\log \Delta})$ on the optimal schedule length, as the following theorem shows.

Theorem 3. *Let the instance be located in the Euclidean plane and let $\alpha > 2$. Then we have $T = \Omega\left({}^{I}/_{\log \Delta}\right)$, where T denotes the optimal schedule length using any power assignment.*

In total we found several bounds on the measure of interference that allow comparisons to the scheduling complexity. To complete these results, we will present a single-hop algorithm that generates a schedule of length $\mathcal{O}(I + \log^2 n)$ whp in the next section and extend this to multi-hop scheduling afterwards.

3 Single-Hop Scheduling

The measure of interference enables us to design randomized algorithms using linear power assignments, i. e., the power for the transmission from u to v is $c \cdot d(u,v)^\alpha$ for some fixed $c \geq \beta\nu$. As a key fact, we can simplify the SINR constraint in this setting as follows. If $\mathcal{R}$ is a set of requests that can be scheduled in one time slot, we have for all nodes v' with $(u',v') \in \mathcal{R}$

$$\sum_{\substack{(u,v)\in\mathcal{R}\\(u,v)\neq(u',v')}} \frac{c \cdot d(u,v)^\alpha}{d(u,v')^\alpha} \leq \frac{c}{\beta} - \nu \enspace .$$

Since $\beta > 1$ we can write equivalently

$$\sum_{\substack{(u,v)\in\mathcal{R}\\(u,v)\neq(u',v')}} \min\left\{1, \frac{d(u,v)^\alpha}{d(u,v')^\alpha}\right\} \leq \frac{1}{\beta} - \frac{\nu}{c} \enspace . \tag{3}$$

For simplicity of notation we replace $\frac{1}{\beta} - \frac{\nu}{c}$ by $\frac{1}{\beta'}$ in the following.

3.1 A Basic Algorithm

The idea of our basic algorithm (Algorithm 1) is that each sender decides randomly in each time slot if it tries to transmit until it is successful. The probability of transmission is set to $\frac{1}{2\beta' I}$ and is not changed throughout the process.

```
while packet has not been successfully transmitted do
    try transmitting with probability $\frac{1}{2\beta' I}$
end
```

Algorithm 1. A simple single-hop algorithm

Theorem 4. *Algorithm 1 generates a schedule of length at most $\mathcal{O}(I \log n)$ whp.*

Proof. Let us first consider the probability of success for a fixed request (u_k, v_k) in a single step of the algorithm. Let X_i, $i \in [n]$, be the 0/1 random variable indicating if sender u_i tries to transmit in this step. Assume a sender u_k tries to transmit in this step, i. e. $X_k = 1$. To make this attempt successful, the SINR constraint (Equation 3) has to be satisfied. We can express this event as $Z \leq {}^{1}\!/_{\beta'}$ where Z is defined by

$$Z = \sum_{\substack{i\in[n]\\i\neq k}} \min\left\{1, \frac{d(u_i,v_i)^\alpha}{d(u_i,v_k)^\alpha}\right\} X_i \enspace .$$

We have $\mathbf{E}[Z] \leq {}^1\!/_{2\beta'}$ and thus we can use Markov's inequality to bound the probability that this packet cannot be transmitted successfully by

$$\mathbf{Pr}\left[Z \geq \frac{1}{\beta'}\right] \leq \mathbf{Pr}\left[Z \geq 2\mathbf{E}[Z]\right] \leq \frac{1}{2} \ .$$

To make the transmission successful the two events $X_k = 1$ and $Z \leq {}^1\!/_{\beta'}$ have to occur. Since they are independent it holds that

$$\mathbf{Pr}\left[X_k = 1, Z \leq \frac{1}{\beta'}\right] = \mathbf{Pr}\left[X_k = 1\right] \cdot \mathbf{Pr}\left[Z \leq \frac{1}{\beta'}\right] \geq \frac{1}{2\beta' I}\left(1 - \frac{1}{2}\right) = \frac{1}{4\beta' I} \ .$$

The probability for packet k not to be successfully transmitted in $(k_0+1)4\beta' I \ln n$ independent repeats of such a step is therefore at most

$$\left(1 - \frac{1}{4\beta' I}\right)^{(k_0+1)4\beta' I \ln n} \leq e^{-(k_0+1)\ln n} = n^{-(k_0+1)} \ .$$

Applying a union bound we get an overall bound on the probability that one of n packets is not successfully transmitted in these independent repeats by n^{-k_0}. This means all senders are successful within $\mathcal{O}(I \log n)$ steps whp. □

3.2 A More Sophisticated Algorithm

An obvious disadvantage of the basic algorithm is that the probability of transmission stays the same throughout the process. To improve it, one idea could be to increase the probability of transmission after some transmissions have successfully taken place. Applying this idea, the new algorithm assigns random delays to all packets. The maximum delay is decreased depending on I^{curr}, which denotes the measure of interference that is induced by the requests that have not been scheduled at this point.

The algorithm works as follows: During one iteraton of the outer *while* loop by repeatedly assigning random delays to the packets the measure of interference is reduced to a half of its initial value. This is repeated until we have $I^{\text{curr}} < \log n$ and the basic algorithm is applied.

Theorem 5. *Algorithm 2 generates a schedule of length at most $\mathcal{O}(I + \log^2 n)$ steps whp.*

The proof of this theorem can be found in the full version.

In sufficiently dense instances, i. e., $I \geq \log^2 n$, the second algorithm yields a constant-factor approximation for the optimal schedule compared to the linear power assignment with high probability. Compared to the optimal power assignment the approximation factor is $\mathcal{O}(\log \Delta \cdot \log n)$ whp for general metrics resp. $\mathcal{O}(\log \Delta)$ for the two-dimensional Euclidean plane.

Algorithm 1 can be implemented in a distributed way losing an additional factor $\log n$ in the following way. In contrast to the centralized problem, the nodes

```
while $I^{curr} \geq \log n$ do
    $J := I^{\mathrm{curr}}$
    while $I^{curr} \geq \frac{J}{2}$ do
        if packet $i$ has not been successfully transmitted then
            assign a delay $1 \leq \delta_i \leq 16e\beta' J$ i. u. r.
            try transmission after waiting the delay
        end
    end
end
execute algorithm Algorithm 1
```

Algorithm 2. An $\mathcal{O}(I + \log^2 n)$ whp algorithm

do not know the correct value of I, thus, they do not know their transmission probability. Now in the distributed setting the algorithm processes in each *while* iteration $\log n$ steps, where in each of these steps the transmission probability is halfed, that is, starting by $^1/_{2\beta'}$ down to $^1/_{2\beta' n}$.

Algorithm 2 can be modified analogeously, leading to a schedule of length $O(\log n \cdot (I + \log^2 n))$ whp.

4 Extensions for Multi-hop Scheduling and Routing

The multi-hop variant of the interference scheduling problem was first stated by Chafekar et al. [4] as *Cross-Layer Latency Minimization* (CLM). Given m source destination pairs (s_i, t_i), the objective is to find paths from s_i to t_i to send the packets along, powers for each transmission and a schedule assigning the hops to time slots. In this section we will present how the measure of interference introduced in Section 2 and the single-hop algorithms from Section 3 can be extended to multi-hop scheduling.

4.1 Multi-hop Scheduling with Fixed Paths

Let us first consider the paths to be fixed. In this case the task is to schedule a set of requests $\mathcal{R}$ consisting of n pairs of nodes that lie on paths, respecting dependencies such that one request may not be served before the ones lying earlier on the path have been served. Obviously, the bounds on the measure of interference proven in Section 2 still hold. We additionally express these dependencies in the dilation D, which is the maximum path length. Of course, any schedule using an arbitrary power assignment has length at least D.

In a naive approach to solve this problem we could regard the multi-hop problem as a concatenation of D single-hop problems and schedule each of them separately. This schedule has a length of $\mathcal{O}((I + \log^2 n)D)$ steps whp. Algorithm 3 extends this idea by assigning a random delay to each packet. This technique has also been applied for scheduling in wired networks, e.g., by Leighton et al. [12].

By this shift, a number of time frames is created and to each of them a set of requests $\mathcal{R}_i$ is assigned. Due to the random delay the measure of interference $I(\mathcal{R}_i)$ is sufficiently balanced between those time frames. As different hops that lie on the same path are assigned to different time frames, our single-hop algorithm can be used to generate a schedule for each time frame.

```
forall $i \in [m]$ do
    assign a delay $1 \leq \delta_i \leq \frac{2eI}{\log^2 n}$ i. u. r.
end
forall $1 \leq t \leq \frac{2eI}{\log^2 n} + D$ do
    execute Algorithm 2 on all hops $(i, j)$ with $\delta_i + j = t$
end
```

Algorithm 3. Fixed path multi-hop scheduling

Theorem 6. *The schedule generated by Algorithm 3 has length $\mathcal{O}(I + D \log^2 n)$ whp.*

Due to space limitations, this proof can be found in the full version.

4.2 Finding Optimal Paths (Routing)

To find optimal paths an approach first used by Srinivasan and Teo for wired networks [18], solving an *Integer Linear Program* (ILP) approximately by using relaxation and randomized rounding, can be adapted. Chafekar et al. [4] also use it as a part of their CLM algorithm.

First, let us formalize the problem of finding paths such that $\max\{I, D\}$ is minimal as ILP. We introduce a set of edges $E \subseteq V \times V$ which describes the set of links that may be used. Let furthermore $N_{\text{in}}(v)$ resp. $N_{\text{out}}(v)$ denote the incoming resp. outgoing edges from v.

Minimize w subject to:

$$\forall i \in [m] \qquad \sum_{e \in N_{\text{out}}(s_i)} y(i,e) - \sum_{e \in N_{\text{in}}(s_i)} y(i,e) = 1 \tag{4a}$$

$$\forall i \in [m], v \in V \setminus \{s_i, t_i\} \qquad \sum_{e \in N_{\text{out}}(v)} y(i,e) - \sum_{e \in N_{\text{in}}(v)} y(i,e) = 0 \tag{4b}$$

$$\forall i \in [m] \qquad \sum_{e \in E} y(i,e) \leq w \tag{4c}$$

$$\forall i \in [m], v \in V \qquad \sum_{e'=(u',v')} y(i,e') \min\left\{1, \frac{d(u',v')^\alpha}{d(u',v)^\alpha}\right\} \leq w \tag{4d}$$

$$\forall i \in [m], e \in E \qquad y(i,e) \in \{0,1\} \tag{4e}$$

This ILP is designed to minimize $w = \max\{I, D\}$ as follows. Condition 4d ensures that $I \leq w$ whereas Condition 4c ensures $D \leq w$. By leaving out Condition 4e, this ILP can be relaxed to an LP which then describes a multicommodity flow problem.

This LP can be solved in polynomial time. Afterwards we can use the LP result to approximate a solution of the ILP, by selecting paths of length at most $2w$ and applying the technique of randomized rounding [16]. In a simple analysis we find out the following. If I^* and D^* are the values such that $\max\{I, D\}$ is minimal – which is the optimal solution for the ILP – we calculate paths such that $I = \mathcal{O}(I^* \log n)$ whp and $D \leq 2D^*$ this way.

4.3 Consequences for the CLM Problem

Let us combine our results to get an approximation algorithm for the CLM problem as stated by Chafekar et al. [4]. Assume there is an optimal choice of paths, powers and a schedule such that the latency is T. Let the measure of interference caused by these paths be denoted by $I^\dagger$ and their dilation by $D^\dagger$. In Section 2 we showed that it holds $I^\dagger = \mathcal{O}(\log \Delta \cdot \log n \cdot T)$. Obviously $D^\dagger = \mathcal{O}(T)$ holds, too.

If I^* and D^* are the values such that $\max\{I, D\}$ is minimal, our path selection algorithm chooses paths such that $I = \mathcal{O}(I^* \log n)$ whp and $D = \mathcal{O}(D^*)$. A schedule by Algorithm 3 using these paths has length $\mathcal{O}(I + D \log^2 n) = \mathcal{O}(I^* \log n + D^* \log^2 n) = \mathcal{O}((I^\dagger + D^\dagger) \log^2 n) = \mathcal{O}(\log \Delta \cdot \log^3 n \cdot T)$ whp. Thus we reached an approximation factor of $\mathcal{O}(\log \Delta \cdot \log^3 n)$ whp. For instances restricted to the Euclidean plane, we even get an approximation factor for $\mathcal{O}(\log \Delta \cdot \log^2 n)$ whp.

References

1. Meyer auf der Heide, F., Vöcking, B.: A packet routing protocol for arbitrary networks. In: Mayr, E.W., Puech, C. (eds.) STACS 1995. LNCS, vol. 900, pp. 291–302. Springer, Heidelberg (1995)
2. Banerjee, S., Misra, A.: Minimum energy paths for reliable communication in multihop wireless networks. In: Proceedings of the 3rd ACM International Symposium Mobile Ad-Hoc Networking and Computing (MOBIHOC), pp. 146–156 (2002)
3. Bansal, N., Liu, Z.: Capacity, delay and mobility in wireless ad-hoc networks. In: Conference of the IEEE Communications Society (INFOCOM), vol. 2, pp. 1553–1563 (2003)
4. Chafekar, D., Anil Kumar, V.S., Marathe, M.V., Parthasarathy, S., Srinivasan, A.: Cross-layer latency minimization in wireless networks with SINR constraints. In: Proceedings of the 8th ACM International Symposium Mobile Ad-Hoc Networking and Computing (MOBIHOC), pp. 110–119 (2007)
5. Chafekar, D., Anil Kumar, V.S., Marathe, M.V., Parthasarathy, S., Srinivasan, A.: Approximation algorithms for computing capacity of wireless networks with SINR constraints. In: Proceedings of the 27th Conference of the IEEE Communications Society (INFOCOM), pp. 1166–1174 (2008)

6. Fanghänel, A., Keßelheim, T., Räcke, H., Vöcking, B.: Oblivious interference scheduling. In: PODC 2009 (submitted, 2009)
7. Gupta, P., Kumar, P.R.: The capacity of wireless networks. IEEE Transactions on Information Theory 46, 388–404 (2000)
8. Thite, S., Balakrishnan, H., Barrett, C.L., Kumar, V.S.A., Marathe, M.V.: The distance-2 matching problem and its relationship to the MAC-layer capacity of ad hoc wireless networks. IEEE Journal on Selected Areas in Communications 22(6), 1069–1079 (2004)
9. Kozat, U.C., Tassiulas, L.: Network layer support for service discovery in mobile ad hoc networks. In: Proceedings of the 22nd Conference of the IEEE Communications Society (INFOCOM), pp. 1965–1975 (2003)
10. Krumke, S.O., Marathe, M.V., Ravi, S.S.: Models and approximation algorithms for channel assignment in radio networks. Wireless Networks 7(6), 575–584 (2001)
11. Leighton, F.T., Maggs, B.M., Ranade, A.G., Rao, S.B.: Randomized routing and sorting on fixed-connection networks. Journal of Algorithms (1994)
12. Leighton, F.T., Maggs, B.M., Rao, S.B.: Packet routing and job-shop scheduling in O(congestion+dilation) steps. Combinatorica (1994)
13. Moscibroda, T., Oswald, Y.A., Wattenhofer, R.: How optimal are wireless scheduling protocols? In. In: Proceedings of the 26th Conference of the IEEE Communications Society (INFOCOM), pp. 1433–1441 (2007)
14. Moscibroda, T., Wattenhofer, R.: The complexity of connectivity in wireless networks. In: Proceedings of the 25th Conference of the IEEE Communications Society (INFOCOM), pp. 1–13 (2006)
15. Moscibroda, T., Wattenhofer, R., Zollinger, A.: Topology control meets SINR: The scheduling complexity of arbitrary topologies. In: Proceedings of the 7th ACM International Symposium Mobile Ad-Hoc Networking and Computing (MOBIHOC), pp. 310–321 (2006)
16. Raghavan, P., Tompson, C.D.: Randomized rounding: a technique for provably good algorithms and algorithmic proofs. Combinatorica 7(4), 365–374 (1987)
17. Ramanathan, S., Lloyd, E.L.: Scheduling algorithms for multi-hop radio networks. ACM SIGCOMM Computer Communication Review 22(4), 211–222 (1992)
18. Srinivasan, A., Teo, C.-P.: A constant-factor approximation algorithm for packet routing, and balancing local vs. global criteria. In: STOC 1997: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing, New York, USA, pp. 636–643 (1997)
19. Wieselthier, J.E., Nguyen, G.D., Ephremides, A.: Energy-efficient broadcast and multicast trees in wireless networks. Mobile Networks and Applications (MONET) 7(6), 481–492 (2002)

Efficient Methods for Selfish Network Design*

Dimitris Fotakis[1], Alexis C. Kaporis[2,3], and Paul G. Spirakis[2,3]

[1] School of Electrical and Computer Engineering
National Technical University of Athens, 15780 Athens, Greece
[2] Department of Computer Engineering and Informatics
University of Patras, 26500 Patras, Greece
[3] Research Academic Computer Technology Institute
N. Kazantzaki Str., University Campus, 26500 Patras, Greece
fotakis@cs.ntua.gr, kaporis@ceid.upatras.gr, spirakis@cti.gr

Abstract. Intuitively, Braess's paradox states that *destroying* a part of a network may *improve* the common latency of selfish flows at Nash equilibrium. Such a paradox is a pervasive phenomenon in real-world networks. Any administrator, who wants to improve equilibrium delays in selfish networks, is facing some basic questions: (i) Is the network *paradox-ridden*? (ii) How can we delete some edges to *optimize* equilibrium flow delays? (iii) How can we modify edge latencies to *optimize* equilibrium flow delays?

Unfortunately, such questions lead to NP-hard problems in general. In this work, we impose some natural restrictions on our networks, e.g. we assume strictly increasing linear latencies. Our target is to formulate *efficient algorithms* for the three questions above. We manage to provide:

- A polynomial-time algorithm that decides if a network is paradox-ridden, when latencies are linear and strictly increasing.
- A reduction of the problem of deciding if a network with arbitrary linear latencies is paradox-ridden to the problem of generating all optimal basic feasible solutions of a Linear Program that describes the optimal traffic allocations to the edges with constant latency.
- An algorithm for finding a subnetwork that is almost optimal wrt equilibrium latency. Our algorithm is *subexponential* when the number of paths is polynomial and each path is of polylogarithmic length.
- A polynomial-time algorithm for the problem of finding the best subnetwork, which outperforms any known approximation algorithm for the case of strictly increasing linear latencies.
- A polynomial-time method that turns the optimal flow into a Nash flow by deleting the edges not used by the optimal flow, and performing minimal modifications to the latencies of the remaining ones.

Our results provide a deeper understanding of the computational complexity of recognizing the Braess's paradox most severe manifestations, and our techniques show novel ways of using the probabilistic method and of exploiting convex separable quadratic programs.

* The 3rd author was partially supported by the FET Unit of EC, under contract no. FP6-021235-2 (ARRIVAL), and by the ICT Programme of the EU, under contract no. ICT-2008-215270 (FRONTS). Part of this work was done while the 1st author was with the Dept. of Information and Communication Systems Eng., University of the Aegean, Greece.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 459–471, 2009.

1 Introduction

A typical instance of *selfish routing* consists of a directed network with a source s and a destination t, with each edge having a non-decreasing function that determines the edge's latency as a function of its traffic, and a rate of traffic divided among an infinite population of players, each willing to route a negligible amount of traffic through a $s-t$ path. The players seek to minimize the sum of edge latencies on their path. Observing the traffic caused by others, each player selects a $s-t$ path of minimum latency. Thus, they reach a *Nash equilibrium* (aka a *Wardrop equilibrium*), where all players route their traffic on paths of equal minimum latency. Under some general assumptions on the latency functions, a Nash equilibrium flow (or simply a *Nash flow*) exists and the common (and the total) players' latency in a Nash flow is unique (see e.g. [25,28]).

Motivation and Previous Work. A Nash equilibrium may not optimize the network performance, measured by the *total latency* incurred by all players. The main tool for quantifying and understanding the performance degradation due to the players' non-cooperative and selfish behaviour has been the *Price of Anarchy* (PoA), which was suggested in a groundbreaking work by Koutsoupias and Papadimitriou [17]. The PoA is the ratio of the total latency of the Nash flow to the optimal total latency. Roughgarden [26] proved that the PoA is independent of the network topology and at most $\rho(\mathcal{D})$, where ρ only depends on the class of latency functions $\mathcal{D}$ (e.g. ρ is 4/3 for linear latencies).

With the PoA very well understood, a few natural approaches for reducing it have been investigated. A simple approach that does not require any network modifications is *Stackelberg routing* [16], where the administrator exploits a small fraction of coordinated traffic to improve the quality of the Nash flow reached by the remaining selfish traffic. For parallel-link networks with arbitrary latencies and for general networks with polynomial latencies, the coordinated traffic can be allocated so that the PoA decreases smoothly to 1 as the fraction of the coordinated traffic increases (see e.g. [27,15,4], and [9] for the case of atomic players with unsplittable traffic). Unfortunately, there are instances for which the PoA remains unbounded under any allocation of the coordinated traffic [4], and instances where enforcing the optimal flow requires a large fraction of coordinated traffic [13]. A different approach is to introduce edge-dependent per-unit-of-traffic *tolls*, that influence the players' selfish choices and induce the optimal flow as the Nash flow of the modified instance. In the *refundable tolls* setting, where tolls affect the players' cost but not the network performance, tolls that enforce the optimal flow can be computed efficiently even if the players have different latency-vs-tolls valuations (see e.g. [7,8,14], see also [6,10] on the performance of refundable tolls for atomic players). However, the idea of tolls is not appealing to the players, since large tolls that significantly increase the players' disutility may be required to enforce the optimal flow (see e.g. [8]).

A simpler way of improving network performance at equilibrium is to exploit the essence of the Braess's paradox [5], namely that removing some network edges may decrease the latency of the Nash flow (see Fig. 1 for an example). Thus, given

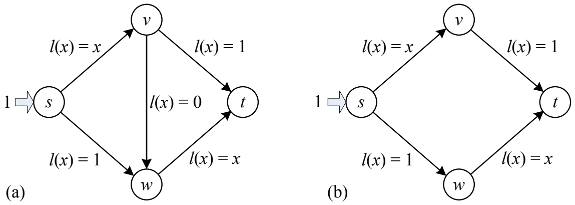


Fig. 1. (a). The optimal flow routes 1/2 unit of traffic on the upper path (s, v, t) and 1/2 unit on the lower path (s, w, t), and achieves a total latency of 3/2. In the Nash flow, all traffic goes through the path (s, v, w, t). The players' latency is 2, and the PoA is 4/3. (b). Without the edge (v, w), the Nash flow coincides with the optimal flow. The network (a) is *paradox-ridden*, and the network (b) is its *best subnetwork*.

an instance of selfish routing, the administrator seeks for the *best subnetwork*, i.e. the subnetwork minimizing the players' latency at equilibrium. Compared to Stackelberg routing and refundable tolls, edge removal is simpler and more appealing. For the administrator, blocking the traffic on some edges is easier and less expensive to implement than setting up a mechanism for collecting tolls on every edge and refunding them to the players. As for the players, edge removal is applied only if it results in a (significant) improvement on their equilibrium latency, which is preferable to either tolls, that increase their disutility, or a Stackelberg strategy, that allocates the coordinated traffic to slower paths.

Recent work indicates that edge removal can improve the performance of real-world networks (see e.g. [28]). In this vein, Valiant and Roughgarden [30] proved that the Braess's paradox occurs with high probability on random networks, and that for a natural distribution of linear latencies, edge removal improves the equilibrium latency by a factor arbitrarily close to 4/3 (i.e. the worst-case PoA for linear latencies) with high probability. Unfortunately, Roughgarden [29] proved that it is NP-hard not only to find the best subnetwork, but also to compute any meaningful approximation to the equilibrium latency on the best subnetwork. In particular, he showed that even for linear latencies, it is NP-hard to distinguish between *paradox-free* instances, where edge removal cannot improve the equilibrium latency, and *paradox-ridden* instances, where the total latency of the Nash flow on the best subnetwork is equal to the optimal total latency (i.e. edge removal can decrease the PoA to 1). This implies that for any $\varepsilon > 0$, it is NP-hard to approximate the equilibrium latency on the best subnetwork within a factor of $4/3 - \varepsilon$ for linear latencies, and within a factor of $\lfloor n/2 \rfloor - \varepsilon$ for general latencies, where n denotes the number of nodes. In fact, the only known algorithm for approximating the equilibrium latency on the best subnetwork is the trivial one, which does not remove any edges and achieves an approximation ratio of 4/3 for linear latencies and $\lfloor n/2 \rfloor$ for general latencies.

Contribution. The motivating question for this work is whether there are some practically interesting settings where a set of edges, whose removal significantly improves the equilibrium latency, can be computed efficiently. Rather surprisingly, we answer this question in the affirmative for several interesting cases. To the best of our knowledge, our results are the first of theoretical nature which indicate that the Braess's paradox can be efficiently detected and eliminated in many interesting cases. Throughout this paper, we mostly focus on the important case of linear latencies, even though some of our results can be generalized to other classes of latency functions (e.g. polynomial latencies).

We first consider the problem of *recognizing paradox-ridden* instances. Even though this problem is NP-complete for arbitrary linear latencies [29], we show that it becomes *polynomially solvable* for the important case of strictly increasing linear latencies[1]. Recognizing a paradox-ridden instance is equivalent to deciding whether the instance admits an optimal flow that is a Nash flow on its subnetwork (cf. Lemma 1). Then removing all edges not used by the optimal flow yields the best subnetwork. However, an instance may admit many different optimal flows. In fact, the NP-hardness proofs in [29] employ instances with exponentially many optimal flows. On the other hand, if the optimal flow is unique, we can recognize paradox-ridden instances by computing it and checking whether it is a Nash flow on its subnetwork. Based on this observation, we present a *polynomial-time* algorithm that recognizes *paradox-ridden* instances with strictly increasing linear latencies (cf. Theorem 1). Furthermore, we reduce the problem of recognizing a paradox-ridden instance with arbitrary linear latencies to the problem of generating all optimal basic feasible solutions of a Linear Program that describes the optimal traffic allocations to the constant latency edges (cf. Theorem 2).

Then we proceed to the more general problem of computing the *best subnetwork* and its equilibrium latency. For instances with polynomially many paths, each of polylogarithmic length, and arbitrary linear latencies, we present a subexponential-time approximation scheme. For any $\varepsilon > 0$, the algorithm computes a subnetwork with an ε-Nash flow in which the players' latencies are within an additive term of $\varepsilon/2$ from the equilibrium latency on the best subnetwork. The running time is exponential in $\text{poly}(\log m)/\varepsilon^2$, where m is the number of edges (cf. Theorem 3). The analysis is based on a novel application of the Probabilistic Method [1] motivated by Althöfer's Lemma [2] and its application to the computation of approximate Nash equilibria for bimatrix games [21,20]. In particular, we apply the Probabilistic Method and show that any flow on any network admits an ε-approximate "sparse" flow, which assigns traffic to $\mathrm{O}(\log m/\varepsilon^2)$ paths (cf. Lemma 2). The proof has to take advantage of the network structure, since the number of paths may be exponential in m. Hence, our result comprises a novel (and more efficient) extension of Althöfer's Lemma to the network setting. In addition, the application to the best subnetwork approximation deals with a congestion game with an infinite number of players, and is fundamentally different from the application of Althöfer's Lemma to approximation of Nash equilibria for bimatrix games. In fact, to the best of our knowledge, this is the first time that similar techniques are applied in the context of selfish routing.

For instances with strictly increasing linear latencies that are not paradox-ridden, we show that there is an instance-dependent $\delta > 0$, such that the

[1] Constant latency edges represent links of practically infinite capacity. Therefore real-world networks are most unlikely to contain many of them, if they contain any.

equilibrium latency is within a factor of $4/3 - \delta$ from the equilibrium latency on the best subnetwork. Since we can efficiently compute the best subnetwork for paradox-ridden instances, we can use the trivial algorithm for the remaining ones, and approximate the equilibrium latency on the best subnetwork within a factor strictly smaller than the inapproximability threshold[2] of 4/3 (cf. Theorem 4).

If the instance is not paradox-ridden however, it is not possible to turn the optimal flow into a Nash flow by just removing edges. Enforcing the optimal flow is possible, if in addition to removing edges, the administrator can modify the latency functions. In Section 5, we present a polynomial-time algorithm for the problem of minimally modifying the latency functions of the edges used by the optimal flow so that the optimal flow is enforced as a Nash flow on the subnetwork used by the optimal flow with the modified latencies (cf. Theorem 5).

Other Related Work. For the problem of finding the best subnetwork in the atomic model, Azar and Epstein [3] obtained strong inapproximability results similar to those in [29]. Interestingly, the Braess's paradox can be dramatically more severe in multi-commodity instances than in single-commodity ones. More precisely, Lin *et al.* [18] proved that for single-commodity instances with general latency functions, the removal of at most k edges cannot improve the equilibrium latency by a factor greater than $k + 1$. On the other hand, Lin *et al.* [19] presented a 2-commodity instance where the removal of a single edge improves the equilibrium latency by a factor of $2^{\Omega(n)}$. As for the impact of the network topology, Milchtaich [23] proved that the Braess's paradox does not occur in (single-commodity) series-parallel networks, which is precisely the class of networks that do not contain the network in Fig. 1.a as a topological minor.

2 Model, Preliminaries, and Problem Definitions

A *selfish routing instance* is a tuple $\mathcal{G} = (G(V, E), (\ell_e)_{e \in E}, r)$, where $G(V, E)$ is a directed network with a source s and a destination t, $\ell_e : \mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$ is a non-decreasing latency function associated with each edge e, and $r > 0$ is the rate of traffic entering the network at s and leaving the network at t. Let $n = |V|$ and $m = |E|$, and let $\mathcal{P}$ denote the set of simple $s - t$ paths in G. We assume that the edge latency functions $\ell_e(x)$ are continuous, differentiable, and convex in the interval $[0, r]$. We mostly focus on *linear* latency functions $\ell_e(x) = a_e x + b_e$, with rational coefficients $a_e, b_e \geq 0$. Such a function is *constant* if $a_e = 0$.

Given a selfish routing instance $\mathcal{G} = (G(V, E), (\ell_e)_{e \in E}, r)$, any subgraph $H(V, E')$, $E' \subseteq E$, obtained from G by edge deletions is called a *subnetwork* of G. H has the same source s and destination t as G, and the edges of H

[2] The reduction of [29, Theorem 3.3] constructs instances where almost all edges have constant latency 0. Using $\ell(x) = \varepsilon x$, for some very small $\varepsilon > 0$, instead of 0, we can show that even for strictly increasing linear latencies, it is NP-hard to approximate the equilibrium latency on the best subnetwork within a factor considerably smaller than 4/3 for all instances. In this sense, our result is best possible.

preserve their latencies in $\mathcal{G}$. Each instance $\mathcal{H} = (H(V,E'), (\ell_e)_{e\in E'}, r)$, where $H(V,E')$ is a subnetwork of $G(V,E)$, is called a *subinstance* of $\mathcal{G}$.

Flows. A ($\mathcal{G}$-feasible) *flow* f is a non-negative vector indexed by $\mathcal{P}$ so that $\sum_{p\in\mathcal{P}} f_p = r$. For a flow f, let $f_e = \sum_{p:e\in p} f_p$ be the amount of flow that f routes on edge e. Flows f and g are *different* if there is an edge e with $f_e \neq g_e$. An edge e is used by flow f if $f_e > 0$. Given a flow f, the latency of each edge e is $\ell_e(f_e)$, and the latency of each path p is $\ell_p(f) = \sum_{e\in p} \ell_e(f_e)$. For an instance $\mathcal{G}$ defined on a network $G(V,E)$ and a flow f, we let $E_f = \{e \in E : f_e > 0\}$ be the set of edges used by f, and $G_f(V,E_f)$ be the subnetwork of G corresponding to f. A flow f is acyclic if G_f contains no cycles.

The *total latency* of a flow f is $C(f) = \sum_{p\in\mathcal{P}} f_p \ell_p(f) = \sum_{e\in E} f_e \ell_e(f_e)$. The *optimal* flow of instance $\mathcal{G}$, denoted o, minimizes the total latency among all $\mathcal{G}$-feasible flows. We let $L^*(\mathcal{G}) = C(o)/r$ be the average latency in the optimal flow. We note that for every subinstance $\mathcal{H}$ of $\mathcal{G}$, $L^*(\mathcal{H}) \geq L^*(\mathcal{G})$. For the latency functions considered in this paper, an optimal flow can be computed efficiently, while for strictly increasing latencies, the optimal flow is unique (in the sense that all optimal flows route the same amount of traffic on every edge).

Nash Flows. The traffic is divided among an infinite population of players, each willing to route a negligible amount of traffic through a minimum latency $s-t$ path. A flow f is a *Nash flow*, if it routes all traffic on minimum latency paths. Formally, f is a Nash flow if for every path p with $f_p > 0$, and every path p', $\ell_p(f) \leq \ell_{p'}(f)$. Therefore, in a Nash flow f, all players incur a common latency $L(f) = \min_{p:f_p>0} \ell_p(f)$ on their paths, and the total latency is $C(f) = rL(f)$.

For the latency functions considered in this paper, every instance $\mathcal{G}$ admits at least one Nash flow, and the common players' latency (and thus the total latency) is the same for all Nash flows (see e.g. [28]). For instance $\mathcal{G}$, we let $L(\mathcal{G})$ (resp. $rL(\mathcal{G})$) be the common players' latency (resp. total latency) for some Nash flow of $\mathcal{G}$. We refer to $L(\mathcal{G})$ (resp. $rL(\mathcal{G})$) as the equilibrium latency (resp. equilibrium total latency) of $\mathcal{G}$. We note that for every subinstance $\mathcal{H}$ of $\mathcal{G}$, $L^*(\mathcal{G}) \leq L(\mathcal{H})$, and that there may be subinstances $\mathcal{H}$ with $L(\mathcal{H}) < L(\mathcal{G})$ (see e.g. Fig. 1). For the class of latency functions considered in this paper, a Nash flow can be computed efficiently, while for strictly increasing latencies, the Nash flow is unique (see e.g. [28, Cor. 2.6.4]).

ε-Nash flows. The definition of a Nash flow can be naturally generalized to that of an "almost Nash" flow. Formally, for some $\varepsilon > 0$, a flow f is an ε-Nash flow if for every path p with $f_p > 0$, and every path p', $\ell_p(f) \leq \ell_{p'}(f) + \varepsilon$.

Price of Anarchy. The *Price of Anarchy* (PoA) of a selfish routing instance $\mathcal{G}$, denoted $\rho(\mathcal{G})$, is the ratio of the equilibrium total latency to the optimal total latency. By the discussion above, $\rho(\mathcal{G}) = L(\mathcal{G})/L^*(\mathcal{G})$.

Other Notation and Conventions. For any integer $k \geq 1$, we let $[k] = \{1,\ldots,k\}$. For an event E in a sample space, we let $\mathbb{P}[E]$ denote the probability of event E happening. For a random variable X, we let $\mathbb{E}[X]$ denote the *expectation* of X. For convenience and wlog., we normalize the traffic rate to 1.

Then $L(\mathcal{G})$ equals both the common players' latency and the total latency at equilibrium, and $L^*(\mathcal{G})$ equals both the optimal average latency and the optimal total latency of $\mathcal{G}$. With the traffic rate normalized to 1, we sometimes identify a selfish routing instance with the corresponding network.

Paradox-Free and Paradox-Ridden Instances. An instance $\mathcal{G}$ defined on a network G is *paradox-free* if for every subnetwork H of G, $L(H) \geq L(G)$. Paradox-free instances do not suffer from the Braess's paradox and their PoA cannot be improved by edge removal. An instance $\mathcal{G}$ is *paradox-ridden* if there is a subnetwork H of G such that $L(H) = L^*(G) = L(G)/\rho(G)$. Namely, the PoA of paradox-ridden instances can decrease to 1 by edge removal.

Best Subnetwork. Given instance $\mathcal{G}$, the *best subnetwork* H^B is a subnetwork of G that minimizes the equilibrium latency, i.e. H^B has $L(H^B) \leq L(H)$ for any subnetwork H of G.

Problem Definitions. We now introduce three basic problems regarding selfish network design:

- **Paradox-Ridden Recognition** (ParRid) : Given an instance $\mathcal{G}$, decide if $\mathcal{G}$ is paradox-ridden.
- **Best Subnetwork Equilibrium Latency** (BSubEL) : Given an instance $\mathcal{G}$ defined on a network G, find the best subnetwork H^B of G and its equilibrium latency $L(H^B)$.
- **Minimum Latency Modification** (MinLatMod) : Given an instance $\mathcal{G}$ defined on a network $G(V, E)$ with a polynomial latency $\ell_e(x) = \sum_{i=0}^{d} a_{e,i} x^i$, $a_{e,i} \geq 0$, for each $e \in E$, find modified latencies $\tilde{\ell}_e(x) = \sum_{i=0}^{d} \tilde{a}_{e,i} x^i$, $\tilde{a}_{e,i} \geq 0$, $e \in E_o$, so that the Euclidean distance of the vectors $(a_{e,i})_{e \in E_o, i \in [d]}$ and $(\tilde{a}_{e,i})_{e \in E_o, i \in [d]}$ is minimum, and for the instance $\tilde{\mathcal{G}}_o$ defined on the network $G_o(V, E_o)$ with latencies $\tilde{\ell}_e(x)$, o is a Nash flow with common latency $L^*(\mathcal{G})$.

3 Recognizing Paradox-Ridden Instances

In this section, we present a polynomial-time algorithm for ParRid on instances with strictly increasing linear latencies. We start with a lemma that reduces ParRid to the problem of checking if some optimal flow o is a Nash flow on G_o.

Lemma 1. *An instance $\mathcal{G}$ defined on a network $G(V, E)$ is paradox-ridden iff there is an optimal flow o that is a Nash flow on the subnetwork $G_o(V, E_o)$.*

For instances with strictly increasing linear latencies, the optimal flow is unique and can be efficiently computed. Then, checking whether the optimal flow o is a Nash flow on G_o can be performed by a shortest path computation.

Theorem 1. ParRid *can be decided in polynomial time for instances with strictly increasing linear latency functions.*

Proof. Computing the unique optimal flow o for an instance $\mathcal{G}$ with strictly increasing linear latencies can be performed in polynomial time (see e.g. [24]). To check whether o is a Nash flow on the subnetwork $G_o(V, E_o)$, we compute the length $d(v)$ of the shortest $s - v$ path wrt the edge lengths $\{\ell_e(o_e)\}_{e \in E_o}$ for all vertices $v \in V$. Then o is a Nash flow if for every edge $(u, v) \in E_o$, $d(v) = d(u) + \ell_{(u,v)}(o_{(u,v)})$ (see e.g. [29, Proposition 2.10]). □

Dealing with Constant Latencies. Next we formulate a general sufficient condition, under which ParRid can be decided in polynomial time for instances with arbitrary linear latencies. Let $\mathcal{G}$ be an instance defined on a network $G(V, E)$, let $E^c = \{e \in E : a_e = 0\}$ be the set of edges with constant latencies, let $E^i = E \setminus E^c$ be the set of edges with strictly increasing latencies, and let $\mathcal{O}$ be the set of different optimal flows of $\mathcal{G}$.

All optimal flows assign the same traffic to the edges with strictly increasing latencies, and can differ only on edges with constant latencies. Given a fixed optimal flow o, we formulate a Linear Program whose feasible solutions correspond to all ($\mathcal{G}$-feasible) flows that agree with the optimal flows on the edges in E^i :

$$\min \sum_{e \in E^c} f_e b_e \,, \quad \text{s.t.}$$

$$\sum_{u:(v,u) \in E^i} o_{(v,u)} + \sum_{u:(v,u) \in E^c} f_{(v,u)} = \sum_{u:(u,v) \in E^i} o_{(u,v)} + \sum_{u:(u,v) \in E^c} f_{(u,v)} \quad \forall v \in V \setminus \{s,t\}$$

$$\sum_{u:(s,u) \in E^i} o_{(s,u)} + \sum_{u:(s,u) \in E^c} f_{(s,u)} = 1 \qquad \text{(LP)}$$

$$\sum_{u:(u,t) \in E^i} o_{(u,t)} + \sum_{u:(u,t) \in E^c} f_{(u,t)} = 1$$

$$f_e \geq 0 \qquad \forall e \in E^c$$

(LP) has a variable f_e for each edge $e \in E^c$, while all o-related terms are fixed and determined by o. An optimal solution to (LP) corresponds to a $\mathcal{G}$-feasible flow that agrees with o on all edges in E^i and allocates traffic to the edges in E^c so that the total latency is minimized. Hence, every optimal solution to (LP) corresponds to an optimal flow. On the other hand, every optimal flow o' has $o_e = o'_e$ for all $e \in E^i$, and is translated into an optimal solution to (LP) by setting $f_e = o'_e$ for all $e \in E^c$. Therefore, there is a one-to-one correspondence between the optimal solutions to (LP) and the optimal flows in $\mathcal{O}$.

Given an optimal flow o, the discussion above reduces the problem of checking if there is a $o' \in \mathcal{O}$ that is a Nash flow on $G_{o'}$ to the problem of generating all optimal solutions of (LP) and checking whether some of them can be translated into a Nash flow on the corresponding subnetwork. This can be performed in polynomial time if (LP)'s optimal solution is unique (see e.g. [22, Theorem 2] on how to efficiently decide uniqueness of the optimal solution). Thus,

Theorem 2. ParRid *can be decided in polynomial time for instances with linear latency functions where (LP) has a unique optimal solution.*

In fact, it suffices to generate all optimal basic feasible solutions. This is true because (LP) allocates traffic to constant latency edges only. Hence, if a feasible solution f can be translated into a Nash flow on the corresponding subnetwork, this holds for any other feasible solution f' with $\{e : f'_e > 0\} \subseteq \{e : f_e > 0\}$. Therefore, the approach above can be extended to instances where (LP) has a small number of basic feasible solutions (i.e. polynomially many in m). This class includes instances with a constant number of constant latency edges.

4 Approximating the Best Subnetwork

Networks with Polynomially Many Short Paths. We present a subexponential-time approximation scheme for BSubEL on networks with polynomially many paths, each of polylogarithmic length. We first show that any flow (on any network) admits an ε-approximate "sparce" flow, which assigns traffic to $\mathrm{O}(\log m/\varepsilon^2)$ paths. The proof builds on the proof of Althöfer's Lemma [2].

Lemma 2. *Let $\mathcal{G}$ be an instance on a network $G(V, E)$, and let f be any $\mathcal{G}$-feasible flow. For any $\varepsilon > 0$, there exists a $\mathcal{G}$-feasible flow $\tilde{f}$ that assigns positive traffic to at most $\lfloor \log(2m)/(2\varepsilon^2)\rfloor + 1$ paths, such that $|\tilde{f}_e - f_e| \leq \varepsilon$, for $e \in E$.*

Proof. For convenience, we let $\mu = |\mathcal{P}|$ denote the number of paths in G, and index the $s - t$ paths in G by integers in $[\mu]$. Since the traffic rate is normalized to 1, we can interpret the flow f as a probability distribution on the set of paths $\mathcal{P}$. We prove that if we select $k > \log(2m)/(2\varepsilon^2)$ paths uniformly at random with replacement according to (the probability distribution) f, and assign to each path j a flow equal to the number of times j is selected divided by k, we obtain a flow that is an ε-approximation to f with positive probability. By the *Probabilistic Method* [1], such a flow exists.

Let ε be any fixed positive number, and let $k = \lfloor \log(2m)/(2\varepsilon^2)\rfloor + 1$. We define k independent identically distributed random variables $P_1, \dots, P_k$, each taking an integer value in $[\mu]$ according to distribution f. Namely, for all $i \in [k]$ and $j \in [\mu]$, $\mathbb{P}[P_i = j] = f_j$. For each path $j \in [\mu]$, let F_j be a random variable defined as $F_j = |\{i \in [k] : P_i = j\}|/k$. By linearity of expectation, $\mathbb{E}[F_j] = f_j$. For each edge e and each random variable P_i, we define an indicator variable $F_{e,i}$ that is 1 if e is included in the path indicated by P_i, and 0 otherwise. Since the random variables $\{P_i\}_{i \in [k]}$ are independent, for every fixed edge e, the variables $\{F_{e,i}\}_{i \in [k]}$ are independent as well. In addition, for every edge e, let $F_e = \frac{1}{k}\sum_{i=1}^{k} F_{e,i}$. We observe that $F_e = \sum_{j : e \in j} F_j$, and that $\mathbb{E}[F_e] = f_e$.

Since $\sum_{j=1}^{\mu} F_j = 1$, we can interpret the value of each F_j as an amount of flow assigned to path j, and the value of each F_e as an amount of flow assigned to edge e. Then the random variables $F_1, \dots, F_\mu$ define a ($\mathcal{G}$-feasible) flow on G that assigns positive traffic to at most k paths and agrees with f on expectation. By applying the Chernoff-Hoeffding bound [12], we obtain that for every edge e,

$$\mathbb{P}[|F_e - f_e| > \varepsilon] \leq 2\mathrm{e}^{-2\varepsilon^2 k} < 1/m\,,$$

where we use that $k > \log(2m)/(2\varepsilon^2)$. By applying the union bound, we obtain that $\mathbb{P}[\exists e : |F_e - f_e| > \varepsilon] < m(1/m) = 1$. Therefore, for any integer $k > \log(2m)/(2\varepsilon^2)$, there is positive probability that the ($\mathcal{G}$-feasible) flow $(F_1, \ldots, F_\mu)$ satisfies $|F_e - f_e| \leq \varepsilon$ for all $e \in E$. By the Probabilistic Method, there exists a flow $\tilde{f}$ with the properties of $(F_1, \ldots, F_\mu)$. □

For any $\varepsilon > 0$, let $\epsilon_1 > 0$ depend on ε and on some parameters of $\mathcal{G}$. By Lemma 2, there exists an ϵ_1-approximation $\tilde{f}$ to a Nash flow f on the best subnetwork $L(H^B)$ that assigns positive traffic to at most $\lfloor \log(2m)/(2\epsilon_1^2) \rfloor + 1$ paths. If G has polynomially many paths, $\tilde{f}$ can be found in subexponential time by exhaustive search. Next we show that if all paths in G are relatively short, $\tilde{f}$ is an ε-Nash flow on $G_{\tilde{f}}$, and all players' latencies in $\tilde{f}$ are at most $L(H^B) + \varepsilon/2$. Thus we obtain a subexponential approximation scheme for BSubEL.

Theorem 3. *Let $\mathcal{G} = (G(V,E), (a_e x + b_e)_{e \in E}, 1)$ be an instance with linear latencies, let $\alpha = \max_{e \in E}\{a_e\}$, and let H^B be the best subnetwork of G. For some constants $d_1, d_2 > 0$, let $|\mathcal{P}| \leq m^{d_1}$ and $|p| \leq \log^{d_2} m$, for all $p \in \mathcal{P}$. Then, for any $\varepsilon > 0$, we can compute in time $m^{\mathrm{O}(d_1 \alpha^2 \log^{2d_2+1}(2m)/\varepsilon^2)}$ a flow $\tilde{f}$ that is an ε-Nash flow on $G_{\tilde{f}}$ and satisfies $\ell_p(\tilde{f}) \leq L(H^B) + \varepsilon/2$, for all paths p in $G_{\tilde{f}}$.*

Proof sketch. Let f be an acyclic Nash flow on the best subnetwork H^B, and for any $\varepsilon > 0$, let $\epsilon_1 = \varepsilon/(2\alpha \log^{d_2}(2m))$. Wlog. we assume that H^B is precisely G_f. By Lemma 2, there exists a $\mathcal{G}$-feasible acyclic flow $\tilde{f}$ on H^B that assigns positive flow to at most $k = \lfloor \log(2m)/(2\epsilon_1^2) \rfloor + 1 = \lfloor 2\alpha^2 \log^{2d_2+1}(2m)/\varepsilon^2 \rfloor + 1$ paths, and satisfies $|f_e - \tilde{f}_e| \leq \epsilon_1$ for all edges e in H^B, and $\tilde{f}_e = 0$ for all edges e not in H^B. Using that f is a Nash flow on H^B with $L(f) = L(H^B)$, we show that for any path p in the subnetwork $G_{\tilde{f}}$ determined by the edges used by $\tilde{f}$, $|\ell_p(f) - L(H^B)| \leq \varepsilon/2$. Therefore, there exists a $\mathcal{G}$-feasible flow $\tilde{f}$ that assigns positive flow to at most k paths, is an ε-Nash flow on $G_{\tilde{f}}$, and satisfies $|\ell_p(f) - L(H^B)| \leq \varepsilon/2$, for all paths p in $G_{\tilde{f}}$. A flow with the properties of $\tilde{f}$ can be computed in time $m^{\mathrm{O}(d_1 k)}$ by exhaustive search. □

Instances with Strictly Increasing Latencies. For such instances, we show how to efficiently approximate the equilibrium latency on the best subnetwork within a factor less than the inapproximability threshold of 4/3.

Theorem 4. *For instances with strictly increasing linear latencies,* BSubEL *can be approximated in polynomial time within a factor of $4/3 - \delta$, where $\delta > 0$ depends on the instance.*

Proof sketch. Let $\mathcal{G}$ be an instance with strictly increasing linear latencies defined on $G(V,E)$, and let H^B be the best subnetwork of $\mathcal{G}$. If $\mathcal{G}$ is paradox-ridden, by Theorem 1, we can recognize it and compute H^B and $L(H^B)$ in polynomial time. Hence for paradox-ridden instances, we have an approximation ratio of 1.

If $\mathcal{G}$ is not paradox-ridden, we use the trivial algorithm that returns the entire network G. Since $\mathcal{G}$ is not paradox-ridden, $L(H^B) > L^*(G)$. Setting $\delta = \rho(G)(L(H^B) - L^*(G))/L(H^B) > 0$, we obtain that $L(G)/L(H^B) = \rho(G) - \delta$. Since G has linear latencies, $\rho(G) \leq 4/3$, and the theorem follows. □

5 Enforcing the Optimal Flow by Latency Modifications

Despite our positive results, there are instances where either finding the best subnetwork is hard, or the equilibrium latency on the best subnetwork is not close to the optimal average latency. For such instances, we present a polynomial-time algorithm that enforces the optimal flow by performing a minimal amount of latency modifications on the edges used by the optimal flow.

Theorem 5. MinLatMod *can be solved in polynomial time for instances with polynomial latency functions.*

Proof. Let $\mathcal{G}$ be an instance defined on a network $G(V, E)$ with a polynomial latency function $\ell_e(x) = \sum_{i=0}^{d} a_{e,i} x^i$, $a_{e,i} \geq 0$, for each $e \in E$. We can efficiently compute an optimal flow o within any specified accuracy (see e.g. [11]) and the corresponding subnetwork $G_o(V, E_o)$.

Let $\alpha = (a_{e,i})_{e \in E_o, i \in [d]}$ be coefficients vector of the latency functions for the edges used by the optimal flow o. We seek a modified coefficients vector $\tilde{\alpha} = (\tilde{a}_{e,i})_{e \in E_o, i \in [d]}$ so that the Euclidean distance of α and $\tilde{\alpha}$ is minimized, and for the instance $\tilde{\mathcal{G}}_o$ defined on G_o with latency functions $\tilde{\ell}_e(x) = \sum_{i=0}^{d} \tilde{a}_{e,i} x^i$, $\tilde{a}_{e,i} \geq 0$, $e \in E_o$, the flow o is a Nash flow with common latency $L^*(G)$. The best vector $\tilde{\alpha}$ is given by the optimal solution to the following Quadratic Program:

$$\begin{aligned} &\min \sum_{e \in E_o} \sum_{i=1}^{d} (a_{e,i} - \tilde{a}_{e,i})^2 && && \\ \text{s.t.} \quad &\sum_{e \in p} \sum_{i=0}^{d} \tilde{a}_{e,i}\, o_e^i = L^*(G) && \forall \text{paths } p \text{ in } G_o && \text{(QP)} \\ &\tilde{a}_{e,i} \geq 0 && \forall e \in E_o\,,\ \forall i \in [d] && \end{aligned}$$

The equality constraints ensure that all paths in G_o have a common latency $L^*(G)$ in o wrt the modified latency functions $\tilde{\ell}$. Thus o is a Nash flow with common latency $L^*(G)$ for the modified instance $\tilde{\mathcal{G}}_o$. (QP) always admits a feasible solution (see e.g. [25, Cor. 2.7]). Moreover, (QP) is a convex separable Quadratic Program, and can be solved in polynomial time within any specified accuracy (see e.g. [11]). □

Remark 1. We can use the same approach to compute a modified coefficients vector that turns the optimal flow o into a Nash flow on G_o wrt to the modified latencies with *any prescribed common latency* Λ.

References

1. Alon, N., Spencer, J.: The Probabilistic Method. John Wiley, Chichester (1992)
2. Althöfer, I.: On Sparse Approximations to Randomized Strategies and Convex Combinations. Linear Algebra and Applications 99, 339–355 (1994)

3. Azar, Y., Epstein, A.: The Hardness of Network Design for Unsplittable Flow with Selfish Users. In: Erlebach, T., Persinao, G. (eds.) WAOA 2005. LNCS, vol. 3879, pp. 41–54. Springer, Heidelberg (2006)
4. Bonifaci, V., Harks, T., Schäfer, G.: Stackelberg Routing in Arbitrary Networks. In: Papadimitriou, C., Zhang, S. (eds.) WINE 2008. LNCS, vol. 5385, pp. 239–250. Springer, Heidelberg (2008)
5. Braess, D.: Über ein Paradox aus der Verkehrsplanung. Unternehmensforschung 12, 258–268 (1968)
6. Caragiannis, I., Kaklamanis, C., Kanellopoulos, P.: Taxes for Linear Atomic Congestion Games. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 184–195. Springer, Heidelberg (2006)
7. Cole, R., Dodis, Y., Roughgarden, T.: How Much Can Taxes Help Selfish Routing? J. Comput. System Sci. 72(3), 444–467 (2006)
8. Fleischer, L., Jain, K., Mahdian, M.: Tolls for Heterogeneous Selfish Users in Multicommodity Networks and Generalized Congestion Games. In: Proc. of FOCS 2004, pp. 277–285 (2004)
9. Fotakis, D.: Stackelberg Strategies for Atomic Congestion Games. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 299–310. Springer, Heidelberg (2007)
10. Fotakis, D., Spirakis, P.: Cost-Balancing Tolls for Atomic Network Congestion Games. In: Deng, X., Graham, F.C. (eds.) WINE 2007. LNCS, vol. 4858, pp. 179–190. Springer, Heidelberg (2007)
11. Hochbaum, D.S., Shanthikumar, J.G.: Convex Separable Optimization is not Much Harder than Linear Optimization. J. ACM 37(4), 843–862 (1990)
12. Hoeffding, W.: Probability Inequalities for Sums of Bounded Random Variables. Journal of the American Statistical Association 58(301), 13–30 (1963)
13. Kaporis, A.C., Spirakis, P.G.: The Price of Optimum in Stackelberg Games on Arbitrary Single Commodity Networks and Latency Functions. In: Proc. of SPAA 2006, pp. 19–28 (2006)
14. Karakostas, G., Kolliopoulos, S.: Edge Pricing of Multicommodity Networks for Heterogeneous Selfish Users. In: Proc. of FOCS 2004, pp. 268–276 (2004)
15. Karakostas, G., Kolliopoulos, S.: Stackelberg Strategies for Selfish Routing in General Multicommodity Networks. Algorithmica 53(1), 132–153 (2009)
16. Korilis, Y.A., Lazar, A.A., Orda, A.: Achieving Network Optima Using Stackelberg Routing Strategies. IEEE/ACM Trans. on Networking 5(1), 161–173 (1997)
17. Koutsoupias, E., Papadimitriou, C.: Worst-Case Equilibria. In: Meinel, C., Tison, S. (eds.) STACS 1999. LNCS, vol. 1563, pp. 404–413. Springer, Heidelberg (1999)
18. Lin, H., Roughgarden, T., Tardos, É.: A Stronger Bound on Braess's Paradox. In: Proc. of SODA 2004, pp. 340–341 (2004)
19. Lin, H., Roughgarden, T., Tardos, É., Walkover, A.: Braess's Paradox, Fibonacci Numbers, and Exponential Inapproximability. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 497–512. Springer, Heidelberg (2005)
20. Lipton, R.J., Markakis, E., Mehta, A.: Playing Large Games Using Simple Strategies. In: Proc. of EC 2003, pp. 36–41 (2003)
21. Lipton, R.J., Young, N.E.: Simple Strategies for Large Zero-Sum Games with Applications to Complexity Theory. In: Proc. of STOC 1994, pp. 734–740 (1994)
22. Mangasarian, O.L.: Uniqueness of Solution on Linear Programming. Linear Algebra and its Applications 25, 151–162 (1979)
23. Milchtaich, I.: Network Topology and the Efficiency of Equilibrium. Games and Economic Behavior 57, 321–346 (2006)

24. Minoux, M.: A Polynomial Algorithm for Minimum Quadratic Cost Flow Problems. European J. of Operational Research 18(3), 377–387 (1984)
25. Roughdarden, T., Tardos, É.: How Bad is Selfish Routing? J. ACM 49(2), 236–259 (2002)
26. Roughgarden, T.: The Price of Anarchy is Independent of the Network Topology. In: Proc. of STOC 2002, pp. 428–437 (2002)
27. Roughgarden, T.: Stackelberg Scheduling Strategies. SIAM J. on Computing 33(2), 332–350 (2004)
28. Roughgarden, T.: Selfish Routing and the Price of Anarchy. MIT Press, Cambridge (2005)
29. Roughgarden, T.: On the Severity of Braess's Paradox: Designing Networks for Selfish Users is Hard. J. Comput. System Sci. 72(5), 922–953 (2006)
30. Valiant, G., Roughgarden, T.: Braess's Paradox in Large Random Graphs. In: Proc. of EC 2006, pp. 296–305 (2006)

Smoothed Analysis of Balancing Networks*

Tobias Friedrich[1,2], Thomas Sauerwald[1], and Dan Vilenchik[3]

[1] International Computer Science Institute, Berkeley, CA, USA
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[3] Computer Science Division, University of California, Berkeley, CA, USA

Abstract. In a load balancing network each processor has an initial collection of unit-size jobs, tokens, and in each round, pairs of processors connected by balancers split their load as evenly as possible. An excess token (if any) is placed according to some predefined rule. As it turns out, this rule crucially effects the performance of the network. In this work we propose a model that studies this effect. We suggest a model bridging the uniformly-random assignment rule, and the arbitrary one (in the spirit of smoothed-analysis) by starting from an arbitrary assignment of balancer directions, then flipping each assignment with probability α independently. For a large class of balancing networks our result implies that after $\mathcal{O}(\log n)$ rounds the discrepancy is *whp* $\mathcal{O}((1/2 - \alpha)\log n + \log\log n)$. This matches and generalizes the known bounds for $\alpha = 0$ and $\alpha = 1/2$.

1 Introduction

In this work we are concerned with two topics whose name contains the word "smooth", but in totally different meaning. The first is *balancing (smoothing) networks*, the second is *smoothed analysis*. Let us start by introducing these two topics, and then introduce our contribution – interrelating the two.

1.1 Balancing (Smoothing) Networks

In the standard abstraction of *smoothing* (balancing) networks [2], processors are modeled as the vertices of a graph and connection between them as edges. Each process has an initial collection of unit-size jobs (which we call tokens). Tokens are routed through the network by transmitting tokens along the edges according to some local rule. The quality of such network is measured by the maximum difference between the number of tokens at any two vertices (after the balancing operations have ended).

The local scheme of communication we study is a *balancer* gate: the number of tokens is split as evenly possible between the communicating vertices with the excess token (if such remains) routed to the vertex towards which the balancer points. More formally, the balancing network consists of n vertices $v_1, v_2, \ldots, v_n$,

* Tobias Friedrich and Thomas Sauerwald were partially supported by postdoctoral fellowships from the German Academic Exchange Service (DAAD).

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 472–483, 2009.

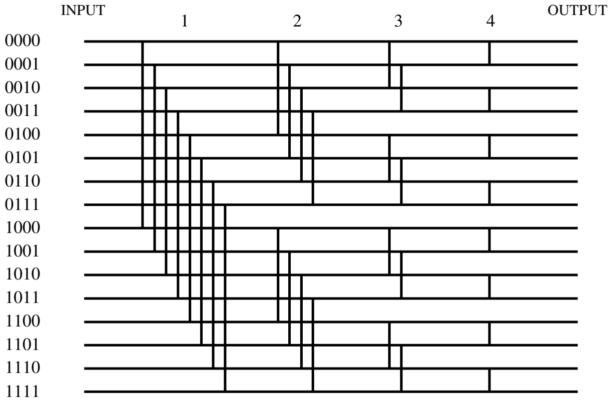


Fig. 1. The network CCC_{16}

and m matchings (either perfect or not) $M_1, M_2, \ldots, M_m$. We associate with every matching edge a balancer gate (that is we think of the edges as directed edges). At the beginning of the first iteration, x_j tokens are placed in vertex v_j, and at every iteration $r = 1, \ldots, m$, the vertices of the network perform a balancing operation according to the matching M_r (that is, vertices v_i and v_j interact if $(v_i, v_j) \in M_r$).

One motivation for considering smoothing networks comes from the server-client world. Each token represents a client request for some service; the service is provided by the servers residing at the vertices. Routing tokens through the network must ensure that all servers receive approximately the same number of tokens, no matter how unbalanced the initial number of tokens is (cf. [2]). More generally, smoothing networks are attractive for multiprocessor coordination and load balancing applications where low-contention is a requirement; these include *producers-consumers* [10] and distributed numerical computations [3]. Together with *counting networks*, smoothing networks have been studied quite extensively since introduced in the seminal paper of [2].

[11, 12] initiated the study of the CCC *network* (cube-connected-cycles, see Figure 1) as a smoothing network. For the special case of the CCC, sticking to previous conventions, we adopt a "topographical" view of the network, thus calling the vertices *wires*, and looking at the left-most side of the network as the "input" and the right-most as the "output". In the CCC, two wires at layer ℓ are connected by a balancer if the respective bit strings of the wires differ exactly in bit ℓ. In [15] it was observed that the CCC is isomorphic to the well-known block network [2, 6]. Therefore, we refer to the CCC-network throughout this paper, though many results in the area are actually stated for the block network. The CCC is a canonical network in the sense that it has the smallest possible depth of $\log n$ (smaller depth cannot ensure any discrepancy independent of the initial one). Moreover, it has been used in more advanced constructions such as the *periodic (counting) network* [2, 6].

As it turns out, the initial setting of the balancers' directions is crucial. Two popular options are an arbitrary orientation or a uniformly random one. A maximal discrepancy of $\log n$ was established for the CCC_n for an arbitrary initial orientation [12]. For a random initial orientation of the CCC_n, [11] show a discrepancy of $2.36\sqrt{\log n}$ for the CCC_n (this holds *whp*[1] over the random initialization), which was improved by [15] to $\log\log n + \mathcal{O}(1)$ (and a matching lower bound).

Results for more general networks have been derived in [16] for arbitrary orientations. For expander graphs, they show an $\mathcal{O}(\log n)$-discrepancy after $\mathcal{O}(\log n)$-rounds. This was recently strengthened assuming the orientations are set randomly and in addition the matchings themselves are chosen randomly [9]. Specifically, for expander graphs constant discrepancy can be achieved *whp* within $\mathcal{O}(\log n\,(\log\log n)^3)$ rounds.

1.2 Smoothed Analysis

Let us now turn to the second meaning of "smoothed". Smoothed analysis comes to bridge between the random instance, which typically has a very specific "unrealistic" structure, and the completely arbitrary instance, which in many cases reflects just the worst case scenario, and thus over-pessimistic in general. In the smoothed analysis paradigm, first an adversary generates an input instance, then this instance is randomly perturbed.

The smoothed analysis paradigm was introduced by in 2001 [18] to help explain why the simplex algorithm for linear programming works well in practice but not in (worst-case) theory. They considered instances formed by taking an arbitrary constraint matrix and perturbing it by adding independent Gaussian noise with variance ε to each entry. They showed that, in this case, the shadow-vertex pivot rule succeeds in expected polynomial time. Independently, [4] studied the issue of Hamiltonicity in a dense graph when random edges are added. In the context of graph optimization problems we can also mention [8, 13], in the context of k-SAT [5, 7], and in various other problems [1, 14, 17, 19].

In our setting we study the following question: what if the balancers were not set completely adversarially but also not in a completely random fashion. Besides the mathematical and analytical challenge that such a problem poses, in real network applications one may not always assume that the random source is unbiased, or in some cases one will not be able to quantitatively measure the amount of randomness involved in the network generation. Still it is desirable to have an estimate of the typical behavior of the network. Although we do not claim that our smoothed-analysis model captures all possible behaviors, it does give a rigorous and tight characterization of the tradeoff between the quality of load balancing and the randomness involved in setting the balancers' directions, under rather natural probabilistic assumptions.

[1] Writing *whp* we mean with probability tending to 1 as n goes to infinity.

As far as we know, no smoothed analysis framework was suggested to a networking related problem. Formally, we suggest the following framework.

1.3 The Model

Our model is similar (and, as we will shortly explain, a generalization of) the periodic balancing circuits studied in [16]. It will be helpful for the reader to bear in mind the following legend: we use superscripts (in round brackets) to denote a time stamp, and subscripts to denote an index. In subscripts, we use the vertices of the graph as indices (thus assuming some ordering of the vertex set). For example, $\mathbf{A}^{(i)}_{u,v}$ stands for the (u, v)-entry in matrix $\mathbf{A}^{(i)}$, which corresponds to time/round i.

Let $M^{(1)}, \ldots, M^{(T)}$ be an arbitrary sequence of T (not necessarily perfect) matchings. With each matching $M^{(i)}$ we associate a matrix $\mathbf{P}^{(i)}$ with $\mathbf{P}^{(i)}_{uv} = 1/2$ if u and v are matched in $M^{(i)}$, $\mathbf{P}^{(i)}_{uu} = 1$ if u is not matched in $M^{(i)}$, and $\mathbf{P}^{(i)}_{uv} = 0$ otherwise.

In round i, every two vertices matched in $M^{(i)}$ perform a balancing operation. That is, the sum of the number of tokens in both vertices is split evenly between the two, with the remaining token (if exists) placed in the vertex pointed by the matching edge.

Remark 1. *In **periodic balancing networks** (see [16] for example) an ordered set of d (usually perfect) matchings is fixed. Every round of balancing is a successive application of the d matchings. Our model is a (slight) generalization of the latter.*

Let us now turn to the smoothed-analysis part. Given a balancing network consisting of a set T of directed matchings, an **α-perturbation** of the network is a flip of direction for every edge with probability α independently of all other edges.

Setting $\alpha = 0$ gives the completely "adversarial model", and $\alpha = 1/2$ is the complete random case.

Remark 2. *For our results, it suffices to consider $\alpha \in [0, 1/2]$. The case $\alpha \geqslant 1/2$ can be reduced to the case $\alpha \leqslant 1/2$ by flipping the initial orientation of all balancers and taking $1 - \alpha$ instead of α. It is easy to see that both distributions are identical.*

1.4 Our Contribution

For a load vector $\mathbf{x}$, its discrepancy is defined to be $\max_{u,v} |\mathbf{x}_u - \mathbf{x}_v|$. We use e_u to denote the unit vector whose all entries are 0 except the u^{th}. For a matrix A, $\lambda(A)$ stands for the second largest eigenvalue of A (in absolute value). Unless stated otherwise, $\|z\|$ stands for the ℓ_2-norm of the vector z.

Theorem 1 *Let G be some balancing network with matchings $M^{(1)}, \ldots, M^{(T)}$. For any two time stamps t_1, t_2 satisfying $t_1 < t_2 \leqslant T$, and any input vector with* *initial discrepancy K, the discrepancy at time step t_2 in α-perturbed G is whp at most*

$$(t_2 - t_1) + 3\left(\tfrac{1}{2} - \alpha\right) t_1 + \Lambda_1 + \Lambda_2,$$

where

$$\Lambda_1 = \max_{w \in V} 4\sqrt{\log n \, \textstyle\sum_{i=1}^{t_1} \sum_{[u:v] \in M^{(i)}} \left((e_u - e_v) \left(\prod_{j=i+1}^{t_2} \mathbf{P}^{(i)} \right) e_w \right)^2 },$$
$$\Lambda_2 = \lambda \left(\textstyle\prod_{i=1}^{t_2} \mathbf{P}^{(i)} \right) \sqrt{n} K.$$

Before we proceed let us motivate the result stated in Theorem 1. There are two factors that effect the discrepancy: the fact that tokens are indivisible (and therefore the balancing operation may not be "perfect", plus the direction of the balancer – which wire gets the extra token), and how many balancing rounds are there. On the one hand, the more rounds there are the more balancing operations are carried, and the smoother the output is. On the other hand, the longer the process runs, its susceptibility to rounding errors and arbitrary placement of excess tokens increases. This is however only a seemingly tension, as indeed the more rounds there are, the smoother the output is. Nevertheless, in the analysis (at least as we carry it), this tension plays part. Specifically, optimizing over these two contesting tendencies is reflected in the choice of t_1 and t_2. Λ_2 is the contribution resulting from the number of balancing rounds being bounded, and Λ_1, along with the first two terms, account for the indivisibly of the tokens. In the cases that will interest us, t_1, t_2 will be chosen so that Λ_1, Λ_2 will be low-order terms compared to the first two terms.

Our Theorem 1 also implies the following results:

- For the aforementioned periodic setting Theorem 1 implies the following: after $\mathcal{O}(\log(Kn)/\nu)$ rounds ($\nu = (1 - \lambda(\mathbf{P}))^{-1}$), $\mathbf{P}$ is the matrix of one period, K the initial discrepancy) the discrepancy is *whp* at most

$$\mathcal{O}\left(\frac{d \log(Kn)}{\nu} \cdot \left(\frac{1}{2} - \alpha \right) + \frac{d \log\log n}{\nu} \right).$$

  Setting $\alpha = 0$ (and assuming K is polynomial in n) we get the result of [16], and for $\alpha = 1/2$ we get the result of [9]. (The restriction on K being polynomial can be lifted but at the price of more cumbersome expressions in Theorem 1. Arguably, the interesting cases are anyway when the total number of tokens, and in particular K, is polynomial). Complete details in the full version.
- For the CCC_n, after $\log n$ rounds the discrepancy is *whp* at most

$$3\left(\tfrac{1}{2} - \alpha\right) \log n + \log\log n + \mathcal{O}(1).$$

Let us now turn to the lower bound.

Theorem 2 *Consider a* CCC_n *with the all-up orientation of the balancers and assume that the number of tokens at each wire is uniformly distributed over* $\{0, 1, \dots, n-1\}$ *(independently at each wire). The discrepancy of the* α*-perturbed network is whp at least*

$$\max\{(\tfrac{1}{2} - \alpha)\log n - 2\log\log n, (1 + o(1))(\log\log n)/2\}.$$

Theorem 2 is proven in Section 2, preceding the proof of Theorem 1 (Section 3), serving as a good introduction to the more complicated proof of Theorem 1. Two more points to note regarding the lower bound:

- For $\alpha = 0$, our lower bound matches the experimental findings of [11], which examined $\mathsf{CCC}_{2^{24}}$, all balancers pointing up, and the input is a random number between 1 and 100,000. Their observed average discrepancy was roughly $(\log n)/2$.
- The input distribution that we use for the lower bound is arguably more natural than the tailored and somewhat artificial ones used in previous lower bound proofs [12, 15].

Finally, we state a somewhat more technical result that we obtain, which lies in the heart of the proof of the lower bound and sheds light on the mechanics of the CCC in the average case input. In what follows, for a balancer b, we let $\mathsf{Odd}(\mathsf{b})$ be an indicator function which is 1 if b had an excess token. By $\mathcal{B}_i$ we denote the set of balancers that effect wire i (that is, there is a simple path in the network going from an input wire, through such a balancer, and ending up at wire i).

Lemma 3. *Consider a* CCC_n *network with any fixed orientation of the balancers. Assume a uniformly distributed input over* $\{0, 1, \dots, n-1\}$*. Every balancer* b *in layer* ℓ*,* $1 \leqslant \ell \leqslant \log n$*, satisfies the following properties:*

- $\mathbf{Pr}[\mathsf{Odd}(\mathsf{b}) = 1] = 1/2$*, and*
- *for every* i*,* $\{\mathsf{Odd}(\mathsf{b}) \mid \mathsf{b} \in \mathcal{B}_i\}$ *is a set of independent random variables.*

For lack of space, the proof of this lemma, as well as other technical details that are missing throughout the paper, can be found in the full version of the paper. Let us just remark that the lemma holding under such strict conditions is rather surprising. First, it is valid regardless of the given orientation. Secondly, and somewhat counter-intuitively, the Odd's of the balancers that effect the same output wire are independent.

2 Lower Bound - Proof of Theorem 2

The proof outline is the following. Given an input vector $\mathbf{x}$ (uniformly distributed over the range $\{0, \dots, n-1\}$), we shall calculate the expected divergence from the average load $\mu = \|\mathbf{x}\|_1/n$. The expectation is taken over both the smoothing operation and the input. After establishing the “right” order of divergence (in expectation) we shall prove a concentration result. One of the main keys to estimating the expectation is Lemma 3 saying that if the input is uniformly distributed as above, then for every balancer b, $\mathbf{Pr}\left[\mathsf{Odd}(\mathsf{b})=1\right] = 1/2$ (the probability is taken only over the input).

Before proceeding with the proof, let us introduce some further notation. Let y_1 be the number of tokens exiting on the top output wire of the network. For any balancer b, $\Psi(\mathsf{b})$ is an indicator random variable which takes the value $-1/2$ if the balancer b was perturbed, and $1/2$ otherwise. $\mathcal{B}(\ell)$ is the set of balancers in layer ℓ, and $\mathsf{b} \rightsquigarrow y_1$ stands for "there is a path of consecutive layers from balancer b to the output of wire 1".

Using the "standard" backward (recursive) unfolding (see also [11, 15] for a concrete derivation for the CCC_n) we obtain that,

$$y_1 = \mu + \sum_{\ell=1}^{\log n} 2^{-\log + \ell} \sum_{\mathsf{b}\in\mathcal{B}(\ell)\wedge \mathsf{b}\rightsquigarrow y_1} \mathsf{Odd}(\mathsf{b}) \cdot \Psi(\mathsf{b}).$$

The latter already implies that the discrepancy of the entire network is at least

$$y_1 - \mu = \sum_{\ell=1}^{\log n} 2^{-\log n + \ell} \sum_{\mathsf{b}\in\ell} \mathsf{Odd}(\mathsf{b}) \cdot \Psi(\mathsf{b}),$$

because there is at least one wire whose output has at most μ tokens (a further improvement of a factor of 2 will be obtained by considering additionally the bottom output wire and prove that on this wire only a small number of tokens exit). Write $y_1 - \mu = \sum_{\ell=1}^{\log n} S_\ell$, defining for each layer $1 \leqslant \ell \leqslant \log n$,

$$S_\ell := 2^{-\log n + \ell} \sum_{\mathsf{b}\in\mathcal{B}_\ell\wedge \mathsf{b}\rightsquigarrow y_1} \mathsf{Odd}(\mathsf{b}) \cdot \Psi(\mathsf{b}). \tag{1}$$

2.1 Proof of $(\frac{1}{2} - \alpha)\log n - 2\log\log n$

We now turn to bounding the expected value of S_ℓ. Using the following facts: (a) the $\mathsf{Odd}(\mathsf{b})$ and $\Psi(\mathsf{b})$ are independent (b) Lemma 3 which gives $\mathbf{E}\left[\mathsf{Odd}(\mathsf{b})\right] = 1/2$ (c) the simple fact that $\mathbf{E}\left[\Psi(\mathsf{b})\right] = \frac{1}{2} - \alpha$ (d) the fact that in layer ℓ there are $2^{\log n - \ell}$ balancers which affect output wire 1 (this is simply by the structure of the CCC_n), we get

$$\begin{aligned}\mathbf{E}\left[S_\ell\right] &= 2^{-\log n + \ell} \sum_{\mathsf{b}\in\mathcal{B}_\ell\wedge \mathsf{b}\rightsquigarrow y_1} \tfrac{1}{2}\cdot(1-2\alpha) \\ &= 2^{-\log n + \ell} \cdot 2^{\log n - \ell} \cdot \tfrac{1}{2} \cdot \left(\tfrac{1}{2} - \alpha\right) = \tfrac{1}{2}\left(\tfrac{1}{2} - \alpha\right).\end{aligned}$$

This in turn gives that

$$\mathbf{E}\left[y_1 - \mu\right] = \mathbf{E}\left[\sum_{\ell=1}^{\log n} S_\ell\right] = \tfrac{1}{2}\left(\tfrac{1}{2} - \alpha\right)\log n.$$

Our next goal is to claim that typically the discrepancy behaves like the expectation; in other words, a concentration result. Specifically, we apply Hoeffdings bound to each layer S_ℓ separately. It is applicable as the random variables $2^{-\log n+\ell} \cdot \mathsf{Odd}(\mathsf{b}) \cdot \Psi(\mathsf{b})$ are independent for balancers within the same layer (such balancers concern disjoint sets of input wires, and the input was chosen independently for each wire). For the bound to be useful we need the range of values for the random variables to be small. Thus, in the probabilistic argument, we shall be concerned only with the first $\log n - \log\log n$ layers (the last $\log\log n$ layers we shall bound deterministically). We use the following Hoeffding bound:

Lemma 4 (Hoeffdings Bound). *Let $Z_1, Z_2, \ldots, Z_n$ be a sequence of independent random variables with $Z_i \in [a_i, b_i]$ for each i. Then for any number $\varepsilon \geqslant 0$,*

$$\mathbf{Pr}\left[\left|\textstyle\sum_{i=1}^n Z_i - \mathbf{E}\left[\sum_{i=1}^n Z_i\right]\right| \geqslant \varepsilon\right] \leqslant 2 \cdot \exp\left(-\frac{2\varepsilon^2}{\sum_{i=1}^n (b_i - a_i)^2}\right).$$

We plug in,

$$Z_{\mathsf{b}} = 2^{-\log n+\ell} \cdot \mathsf{Odd}(\mathsf{b}) \cdot \Psi(\mathsf{b}), \quad \varepsilon = 2^{(\ell - \log n + \log\log n)/2}, \quad (b_i - a_i)^2 = \left(2^{\ell - \log n}\right)^2,$$

and the sum is over $2^{\log n - \ell}$ balancers in layer ℓ. Therefore,

$$\mathbf{Pr}\left[|S_\ell - \mathbf{E}[S_\ell]| \geqslant 2^{(\ell-\log n+\log\log n)/2}\right] \leqslant 2\exp\left(-\frac{2\, 2^{\ell-\log n+\log\log n}}{2^{\ell-\log n}}\right) \leqslant n^{-1}.$$

In turn, with probability at least $1 - \log n/n$ (take the union bound over at most $\log n$ S_ℓ terms):

$$\sum_{\ell=1}^{\log n - \log\log n} S_\ell \geqslant \tfrac{1}{2}\left(\tfrac{1}{2} - \alpha\right)(\log n - \log\log n) - \sum_{\ell=1}^{\log n-\log\log n} 2^{(\ell-\log n+\log\log n)/2}.$$

The second term is just a geometric series with quotient $\sqrt{2}$, and therefore can be bounded by $\frac{1}{1-1/\sqrt{2}} < 4$.

For the last $\log\log n$ layers, we have that for every ℓ, $|S_\ell|$ cannot exceed $\frac{1}{2}$, and therefore their contribution, in absolute value is at most $\frac{1}{2}\log\log n$. Wrapping it up, *whp*

$$y_1 - \mu = \sum_{\ell=1}^{\log n} S_\ell \geqslant \tfrac{1}{2}\left(\tfrac{1}{2} - \alpha\right)(\log n - \log\log n) - 4 - \tfrac{1}{2}\log\log n.$$

The same calculation implies that the number of tokens at the *bottom*-most output wire deviates from μ in the same way (just in the opposite direction).

Hence, the discrepancy is *whp* lower bounded by (using the union bound over the top and bottom wire, and not claiming independence)

$$y_1 - y_n \geqslant \left(\tfrac{1}{2} - \alpha\right)\log n - 8 - \left(\tfrac{3}{2} - \alpha\right)\log\log n \geqslant \left(\tfrac{1}{2} - \alpha\right)\log n - 2\log\log n.$$

2.2 Proof of $(1 + o(1)) \log \log n/2$

The proof here goes along similar lines to Section 2.1, only that now we choose the set of balancers we apply it to more carefully. By the structure of the CCC_n, the last x layers form the parallel cascade of $n/2^x$ independent CCC subnetworks each of which has 2^x wires (by independent we mean that the set of balancers is disjoint).

We call a subnetwork *good* if after an α-perturbation of the all-up initial orientation, all the balancers were not flipped (that is, still point up).

The first observation that we make is that *whp* (for a suitable choice of x, to be determined shortly) at least one subnetwork is good. Let us prove this fact.

The number of balancers effecting the top (or bottom) wire in one of the subnetworks is $\sum_{\ell=1}^{x} 2^\ell \leqslant 2^{x+1}$. In total, there are no more than $2 \cdot 2^{x+1}$ effecting both wires. The probability that none of these balancers was flipped is (using our assumption $\alpha \leqslant 1/2$) $(1-\alpha)^{2^{x+2}} \geqslant 2^{-2^{x+2}}$. Choosing $x = \log \log n - 1$, this probability is at least $n^{-1/2}$; there are at least $n/\log n$ such subnetworks, thus the probability that none is good is at most

$$\left(1 - n^{-1/2}\right)^{n/\log n} = o(1).$$

Fix one good subnetwork and let μ' be the average load at the input to that subnetwork. Repeating the arguments from Section 2.1 (with $\alpha = 0$, $\log n$ rescaled to $x = \log \log n - 4$, and now using the second item in Lemma 3 which guarantees that the probability of $\mathsf{Odd}(\cdot) = 1$ is still $1/2$, for any orientation of the balancers) gives that in the top output wire of the subnetwork there are *whp* at least $\mu' + (\log \log n)/4 - \mathcal{O}(\log \log \log n)$ tokens, while on the bottom output wire there are *whp* at most $\mu' - (\log \log n)/4 + \mathcal{O}(\log \log \log n)$ tokens. Using the union bound, the discrepancy is *whp* at least their difference, that is at least $(\log \log n)/2 - \mathcal{O}(\log \log \log n)$.

3 Upper Bound - Proof of Theorem 1

We shall derive our bound by measuring the difference between the number of tokens at any vertex and the average load (as we did in the proof of the lower bound for the CCC_n). Specifically we shall bound $\max_i |y_i^{(t)} - \mu|$, $y_i^{(t)}$ being the number of tokens at vertex i at time t (we use $\mathbf{y}^{(t)} = (y_i)_{i \in V}$ for the vector of loads). There are two contributions to the divergence from μ (which we analyze separately):

- The divergence of the idealized process from μ due to its finiteness.
- The divergence of the actual process from the idealized process due to indivisibility.

The idea to compare the actual process to an idealized one was suggested in [16] and was analyzed using well-known convergence results of Markov chains. Though we were inspired by the basic setup from [16] and the probabilistic analysis from [9], our setting differs in a crucial point: when dealing with the case $0 < \alpha < 1/2$, we get a delicate mixture of the deterministic and the random model. The random variables in our analysis are not symmetric anymore which leads to additional technicalities.

Formally, let $\xi^{(t)}$ be the load vector of the idealized process at time t, then by the triangle inequality ($\mathbf{1}$ is the all-one vector)

$$\|\mathbf{y}^{(t)} - \mu\mathbf{1}\|_\infty \leqslant \|\mathbf{y}^{(t)} - \xi^{(t)}\|_\infty + \|\xi^{(t)} - \mu\mathbf{1}\|_\infty.$$

Proposition 5. *Let G be some balancing network with matchings $M^{(1)}, \ldots, M^{(T)}$. Then,*

- $\|\xi^{(t)} - \mu\mathbf{1}\|_\infty \leqslant \Lambda_2$,
- *whp over the α-perturbation operation,* $\|\mathbf{y}^{(t)} - \xi^{(t)}\|_\infty \leqslant (t_2 - t_1) + 3\left(\frac{1}{2} - \alpha\right) t_1 + \Lambda_1$.

Theorem 1 then follows. The proof of the first item in Proposition 5 is a rather standard spectral argument (details in the full version). Let us outline the proof of the second item:

3.1 Proof of Proposition 5: Bounding $\|\mathbf{y}^{(t)} - \boldsymbol{\xi}^{(t)}\|_\infty$

The proof of this part resembles in nature the proof of Theorem 2. Assuming an ordering of G's vertices, for a balancer b in round t, $\mathsf{b} = (u, v)$, $u < v$, we set $\Phi_{u,v}^{(t)} = 1$ if the initial direction (before the perturbation) is $u \to v$ and -1 otherwise (in the lower bound we considered the all-up orientation thus we had no use of these random variables). As in Section 2, for a balancer b in round t, the random variable $\Psi_{\mathsf{b}}^{(t)}$ is $-1/2$ if the balancer is perturbed and $1/2$ otherwise. Finally, recall that $\mathsf{Odd}(\mathsf{b}) = 1$ if there is an excess token, and 0 otherwise. Using these notation we define a *rounding vector* $\rho^{(t)}$, which accounts for the rounding errors in step t. Formally,

$$\rho_u^{(t)} = \begin{cases} \mathsf{Odd}(y_u^{(t-1)} + y_v^{(t-1)}) \cdot \Psi_{u,v}^{(t)} \cdot \Phi_{u,v}^{(t)} & \text{if } u \text{ and } v \text{ are matched in } M^{(t)}, \\ 0 & \text{otherwise.} \end{cases}$$

Now we can write the actual process as follows:

$$\mathbf{y}^{(t)} = \mathbf{y}^{(0)}\mathbf{P}^{(t-1)} + \rho^{(t)}. \tag{2}$$

Let $M_{\mathsf{Even}}^{(t)}$ be the set of balancers at time t with no excess token, and $M_{\mathsf{Odd}}^{(t)}$ the ones with. Also, let e_i be the vector whose entries are 0 except the i^{th} which is 1. We can rewrite $\rho^{(t)}$ as follows:

$$\rho^{(t)} = \textstyle\sum_{(u,v) \in M_{\mathsf{Odd}}^{(t)}} \Psi_{u,v}^{(t)} \cdot \Phi_{u,v}^{(t)} \cdot \left(e_i - e_j\right).$$

Unfolding equation (2), similarly to [16], yields then

$$\mathbf{y}^{(t)} = \mathbf{y}^{(0)}\mathbf{P}^{[1,t]} + \sum_{i=1}^{t} \rho^{(i)}\mathbf{P}^{[i+1,t]}. \tag{3}$$

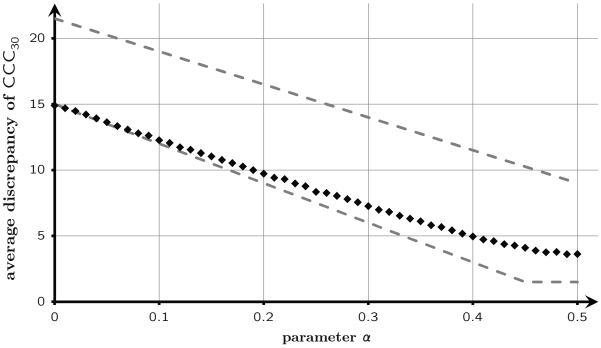


Fig. 2. Discrepancy for various α-values of $\mathsf{CCC}_{2^{30}}$ with random input from $[0, 2^{30}]$. The dotted line describes the experimental results, the broken lines are our theoretical lower and upper bounds.

Observe that $\mathbf{y}^{(0)}\mathbf{P}^{[1,t]}$ is just $\xi^{(t)}$ (as $\xi^{(0)} = \mathbf{y}^{(0)}$), and therefore

$$\mathbf{y}^{(t)} - \xi^{(t)} = \textstyle\sum_{i=1}^{t} e^{(i)}\mathbf{P}^{[i+1,t]} = \sum_{i=1}^{t} \sum_{(u,v)\in M_{\mathsf{Odd}}^{(i)}} \mathit{\Psi}_{u,v}^{(i)} \cdot \mathit{\Phi}_{u,v}^{(i)} \cdot (e_u - e_v) \cdot \mathbf{P}^{[i+1,t]}.$$

In turn,

$$\left(\mathbf{y}^{(t)} - \xi^{(t)}\right)_v = \textstyle\sum_{i=1}^{t} \sum_{(u,w)\in M_{\mathsf{Odd}}^{(i)}} \mathit{\Psi}_{u,v}^{(i)} \cdot \mathit{\Phi}_{u,v}^{(i)} \cdot \left(\mathbf{P}_{u,v}^{[i+1,t]} - \mathbf{P}_{w,v}^{[i+1,t]}\right). \qquad (4)$$

Our next task is to bound equation (4) to receive the desired term from Proposition 5. We do that similar in spirit to the way we went around in Section 2.1. We break this sum into its first t_1 summands (whose expected sum we calculate and to which we apply a large-deviation-bound). The remaining $(t - t_1)$ terms are bounded deterministically. The remainder of the proof can be found in the full version of this paper.

4 Experimental Result

We examined experimentally how well a $\mathsf{CCC}_{2^{30}}$ balances a random input from $[0, 2^{30}]$, for different α values between 0 and 1/2. Figure 2 presents the average discrepancy over 100 runs, together with the following slightly better bounds on the *expected* discrepancy $\mathit{\Delta}$ in the random-input case:

- $\mathit{\Delta} \leqslant (\frac{1}{2} - \alpha) \cdot (\log n - \lceil \log\log n \rceil) + \lceil \log\log n \rceil + 4$,
- $\mathit{\Delta} \geqslant \max\{(1/2 - \alpha)\log n,\ 1/2\,(1 - \frac{1}{n})\,(\lfloor \log\log n \rfloor - 1)\}$.

Bibliography

[1] Arthur, D., Vassilvitskii, S.: Worst-case and smoothed analysis of the icp algorithm, with an application to the k-means method. In: 47th IEEE Symp. on Found. of Comp. Science (FOCS 2006), pp. 153–164 (2006)

[2] Aspnes, J., Herlihy, M., Shavit, N.: Counting networks. J. of the ACM 41(5), 1020–1048 (1994)

[3] Bertsekas, D., Tsitsiklis, J.: Parallel and Distributed Computation: Numerical Methods. Athena Scientific (1997)

[4] Bohman, T., Frieze, A., Martin, R.: How many random edges make a dense graph hamiltonian? Random Structures and Algorithms 22(1), 33–42 (2003)

[5] Coja-Oghlan, A., Feige, U., Frieze, A., Krivelevich, M., Vilenchik, D.: On smoothed k-CNF formulas and the walksat algorithm. In: 20th ACM-SIAM Symp. on Discrete Algorithms (SODA 2009) (2009)

[6] Dowd, M., Perl, Y., Rudolph, L., Saks, M.: The periodic balanced sorting network. J. of the ACM 36(4), 738–757 (1989)

[7] Feige, U.: Refuting smoothed 3CNF formulas. In: 48th IEEE Symp. on Found. of Comp. Science (FOCS 2007), pp. 407–417 (2007)

[8] Flaxman, A., Frieze, A.: The diameter of randomly perturbed digraphs and some applications. Random Structures and Algorithms 30, 484–504 (2007)

[9] Friedrich, T., Sauerwald, T.: Near-perfect load balancing by randomized rounding. In: 41st Annual ACM Symposium on Theory of Computing, STOC 2009 (to appear, 2009)

[10] Herlihy, M., Shavit, N.: The Art of Multiprocessor Programming. Morgan Kaufmann, San Francisco (2008)

[11] Herlihy, M., Tirthapura, S.: Randomized smoothing networks. J. Parallel and Distributed Computing 66(5), 626–632 (2006a)

[12] Herlihy, M., Tirthapura, S.: Self-stabilizing smoothing and counting networks. Distributed Computing 18(5), 345–357 (2006b)

[13] Krivelevich, M., Sudakov, B., Tetali, P.: On smoothed analysis in dense graphs and formulas. Random Structures and Algorithms 29(2), 180–193 (2006)

[14] Manthey, B., Rüdiger, R.: Smoothed analysis of binary search trees. Theoret. Computer Sci. 378(3), 292–315 (2007)

[15] Mavronicolas, M., Sauerwald, T.: The impact of randomization in smoothing networks. In: 27th Annual ACM Principles of Distributed Computing (PODC 2008), pp. 345–354 (2008)

[16] Rabani, Y., Sinclair, A., Wanka, R.: Local divergence of Markov chains and the analysis of iterative load balancing schemes. In: 39th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1998), pp. 694–705 (1998)

[17] Röglin, H., Vöcking, B.: Smoothed analysis of integer programming. Math. Program. 110(1), 21–56 (2007)

[18] Spielman, D., Teng, S.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. of the ACM 51(3), 385–463 (2004)

[19] Vershynin, R.: Beyond hirsch conjecture: Walks on random polytopes and smoothed complexity of the simplex method. In: 47th IEEE Symp. on Found. of Comp. Science (FOCS 2006), pp. 133–142 (2006)

Names Trump Malice: Tiny Mobile Agents Can Tolerate Byzantine Failures

Rachid Guerraoui[1] and Eric Ruppert[2]

[1] EPFL
[2] York University

Abstract. We introduce a new theoretical model of ad hoc mobile computing in which agents have restricted memory, highly unpredictable movement and no initial knowledge of the system. Each agent's memory can store $O(1)$ bits, plus a unique identifier, and $O(1)$ other agents' identifiers. Inputs are distributed across n agents, and all agents must converge to the correct output value. We give a universal construction that proves the model is surprisingly strong: It can solve any decision problem in $NSPACE(n \log n)$. Moreover, the construction is robust with respect to Byzantine failures of a constant number of agents.

1 Introduction

Technological advances have made it possible to build ad hoc networks of mobile tiny computing devices that can perform control and monitoring applications without any specific infrastructure. Much work has been devoted to designing and experimenting on such networks, but little has focussed on devising theoretical models to capture their inherent power and limitations. Without such a model, it is impossible to verify the correctness of an algorithm or determine whether some performance bottleneck results from an inherent complexity lower bound or is merely an artifact of a particular environment. Ideally, a theoretical model should be weak enough to cover a large spectrum of environments, yet strong enough to capture what can indeed be computed in such systems.

An elegant candidate for a theoretical framework is the *population protocol* model of Angluin *et al.* [1], which makes absolutely minimal assumptions. They assumed totally asynchronous agents, no system infrastructure, $O(1)$ bits of memory per agent and made no assumptions about agents' mobility patterns except for a fairness condition, which ensures agents are not forever disconnected. To study computability in the model, they assumed inputs are distributed across all agents and, after a sequence of pairwise interactions, each agent must converge to the correct output. Angluin *et al.* [3] proved the model can solve exactly those decision problems expressible by first-order formulas in Presburger arithmetic.

Unfortunately, this class of solvable problems is fairly small. For instance, it does not include multiplication. Moreover, even for this restricted class, algorithms tolerate no failures or, at worst, a fixed number of benign failures [6]. However, in the hostile environments where ad hoc mobile networks are deployed, arbitrary failures of some agents are often expected. Even one such failure can prevent any non-trivial computation by population protocols. (See Sect. 3.)

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 484–495, 2009.

A key obstacle to more sophisticated computations and fault-tolerance in the population protocol model is anonymity: all agents behave identically. Although a tiny device's memory is often very constrained, it is usually sufficient to store a unique identity. Typically, a modern tiny device consists of a micro-controller such as the MSP430 [11], which has 16 KB of RAM, possibly with a flash-memory component, such as the M25PX64 [9], which stores a few additional megabytes that are more expensive to access. Devices are often equipped with a unique identifier. For example, they might contain Maxim's DS2411 chip, which stores just 64 bits of ROM and is set by the factory to store a unique serial number.

Our goal is to define a minimal model of constrained, yet uniquely identified, tiny devices. Our model resembles population protocols, but we assume all n agents have unique identifiers (ids) and can store $O(1)$ other agents' ids. We call our model the *community protocol* model, thinking of a community as a collection of unique individuals, in contrast to a population, which is merely an agglomeration of a nameless multitude. We assume ids are stored in ROM (as in the DS2411 chip), so that Byzantine agents cannot alter their ids. We restrict the usage of ids to their fundamental purpose, *identification*, by assuming algorithms can only compare ids. (An algorithm cannot, for example, perform arithmetic on ids.) In addition to *having* ids, the ability of agents to *remember* other ids is crucial as, otherwise, the model would be as weak as population protocols.

Our main result describes how n agents can use a community protocol to simulate a Turing machine. As in the population protocol model, a single algorithm must work for all values of n. Thus, the agents, whose collective memory capacity stores $O(n \log n)$ bits, can solve any decision problem that is in $NSPACE(n \log n)$, even though that memory is scattered across agents with unpredictable movement. (We focus on decision problems, but the results easily generalize to computing functions.) Furthermore, our simulation is resilient to a constant number of Byzantine failures. So, although community protocols have only slightly more memory than population protocols, they are much more powerful: they solve a much wider class of problems and tolerate Byzantine failures.

There are several challenges to overcome in designing the simulation. The system's initially unstructured memory must be organized to allow tasks like garbage collection. Similarly, the agents must organize themselves for a division of labour. Furthermore, both of these tasks must be resilient to Byzantine failures. Another key difficulty is the impossibility of determining when these initial set-up tasks are complete, so it must be possible to restart any tasks that depend on them (while ensuring that Byzantine agents cannot cause spurious restarts).

We use the condition-based approach [8] to characterize the power of community protocols. A Byzantine agent can immediately change or discard its portion of the input. Thus, there must be preconditions that ensure such changes cannot affect the outcome. For example, to compute the majority value of input bits, there must be a precondition that the difference between the number of 0's and the number of 1's is at least $2f$; otherwise f Byzantine agents flipping their input bit would cause any algorithm to produce an incorrect output.

Theorem 1. *Any decision problem in $NSPACE(n \log n)$ can be computed by a community protocol in a way that tolerates $f = O(1)$ Byzantine agents, given preconditions ensuring the output does not change if up to f of the input characters are changed and up to $3f+1$ of the input characters are deleted.*

This leaves a slight gap: clearly, changes to f inputs must not affect the output, but our construction requires slightly stronger preconditions. The proof of Theorem 1 generalizes: if each agent could store more ids, they could tolerate more failures: if each agent can store $O(f^2)$ ids, then any problem in $NSPACE(fn \log n)$ can be solved tolerating f Byzantine failures (again, with preconditions). In contrast, most work on Byzantine failures focusses on the number of shared objects or messages, not the amount of local memory used.

2 Computation Models

After a brief, informal description of population protocols, we define community protocols. We also describe a version of pointer machines used in our proof.

In the **population protocol** model [1], a system is a collection of agents. Each agent can be in one of a finite number of possible states. Each agent has an input value, which determines its initial state. When two agents meet, they exchange information about their states and simultaneously update their own states, according to a joint transition function. Each possible agent state has an associated output value. The input assignment, transition function and the association of an output value with each state provides a complete specification of an algorithm. Algorithms are *uniform*: they do not depend on the size of the system. An execution begins with a collection of agents assigned their initial values and proceeds by an infinite series of pairwise interactions. An execution is *fair* if every system configuration that is forever reachable is eventually reached.

For a finite alphabet A, A^* denotes the set of finite strings over A, and A^+ denotes the set of non-empty finite strings over A. Let Σ be the finite set of input values for individual agents. Let Y be the set of possible outputs. An algorithm (stably) *computes* a function $f : \Sigma^+ \rightarrow Y$ if, for all $x \in \Sigma^+$, every fair execution starting with the characters of x assigned as inputs to $|x|$ agents eventually stabilizes to output $f(x)$ (*i.e.*, from some time onward, all agents output $f(x)$).

We extend the population protocol model to obtain our **community protocol** model by assigning unique ids to agents. Let U be an infinite ordered set containing all possible ids. A community protocol *algorithm* is specified by: (1) a finite set, B, of possible basic states, (2) an integer $d \geq 0$ representing the number of ids that can be remembered by an agent, (3) an input map $\iota : \Sigma \rightarrow B$, (4) an output map $\omega : B \rightarrow Y$, and (5) a transition relation $\delta \subseteq Q^4$, where $Q = B \times (U \cup \{\perp\})^d$. The state of an agent stores an element of B, together with up to d ids. The first of the d slots will be initialized with an agent's unique id. If any of the d slots is not currently storing an id, it contains a null id $\perp \notin U$. Thus, Q is the set of possible agent states. The transition relation indicates what happens when two agents interact: if $(q_1, q_2, q_1', q_2') \in \delta$, it means that when two agents in states q_1 and q_2 meet, they can move into states q_1' and q_2', respectively.

As in population protocols, algorithms are uniform: they cannot use any bound on the number of agents, so U is infinite. Thus, the d slots in an agent's state intended for ids could store arbitrary amounts of information. To avoid this, we require that agents store only ids they have learned from other agents. From a practical perspective, this implies that if the algorithm is actually implemented in a real system, $O(\log n)$ bits of memory per agent will suffice. This is analogous to the common assumption in models like random access machines that the number of bits that fit into one memory word is logarithmic in the size of the problem. To keep the model minimal, we require algorithms' operations on ids to be *comparison-based.* These two constraints are formalized as follows. (1) If $(q_1, q_2, q'_1, q'_2) \in \delta$ and id appears in q'_1 or q'_2 then id appears in q_1 or q_2. (2) Consider a transition $(q_1, q_2, q'_1, q'_2) \in \delta$. Let $u_1 < u_2 < \cdots < u_k$ be all ids that appear in any of the four states q_1, q_2, q'_1 and q'_2. Let $v_1 < v_2 < \cdots < v_k$ be ids. If $\rho(q)$ is the state obtained from q by replacing all occurrences of each id u_i by v_i, then we require that $(\rho(q_1), \rho(q_2), \rho(q'_1), \rho(q'_2))$ is also in δ.

A *configuration* of the algorithm consists of a finite vector of elements from Q. Let $X \subseteq \Sigma^+$ be the set of all possible input strings. An *initial configuration* for n agents is a vector in Q^n of the form $(\langle \iota(x_j), u_j, \bot, \bot, \ldots, \bot \rangle)_{j=1}^n$ where $u_1, \ldots, u_n$ are distinct elements of U and x_j is the input to the jth agent. The input string represented by this initial configuration is $x = x_{\pi(1)} x_{\pi(2)} \ldots x_{\pi(n)} \in X$, where π is the permutation of $\{1, \ldots, n\}$ such that $u_{\pi(1)} < u_{\pi(2)} < \cdots < u_{\pi(n)}$. (In other words, the input string x is the string of input symbols ordered by agent ids.)

We first define fair executions for the failure-free case. If $C = (q_1, \ldots, q_n)$ and $C' = (q'_1, \ldots, q'_n)$ are two configurations, we say C *reaches* C' *in a single step* (denoted $C \rightarrow C'$) if there are indices $i \neq j$ such that $(q_i, q_j, q'_i, q'_j) \in \delta$ and for all k different from i and j, $q_k = q'_k$. A failure-free *execution* on input string $x \in X$ is an infinite sequence of configurations $C_0, C_1, \ldots$, such that C_0 is an initial configuration for x and $C_i \rightarrow C_{i+1}$ for all $i \geq 0$. In reality, several pairs of agents may interact simultaneously, but simultaneous interactions can simply be listed, in any order, in the execution since they are independent of one another. A failure-free execution is called *fair* if, for each C that appears infinitely often and for each C' such that $C \rightarrow C'$, C' also appears infinitely often.

We now define executions in the presence of Byzantine agents. Since the order of agents within configuration vectors is unimportant (and invisible to the algorithm itself), assume for convenience that the Byzantine agents occupy the last components of the vectors. An execution with $t \leq n$ Byzantine failures is an infinite sequence of configurations $C_0, C_1, \ldots$ such that C_0 is an initial configuration of the algorithm, and for all $i \geq 0$, either (1) $C_i \rightarrow C_{i+1}$ or (2) the first $n - t$ components are unchanged between C_i and C_{i+1} and the ids of the last t components are unchanged. Case (2) allows Byzantine agents to undergo arbitrary state changes (without altering their ids). In particular, they can change the ids in the other $d - 1$ slots of their states to any elements of U.

Byzantine agents are exempted from fairness requirements. In particular, a Byzantine agent might never interact with any agent. For simplicity assume that, in any execution, each Byzantine agent can only have a finite (but unbounded) number of different states. Thus, the entire system only has a finite (but unbounded) number of different configurations in any execution. Consider an execution $C_0, C_1, \ldots$ with t Byzantine agents. Let D_i be the first $n - t$ components of C_i. The execution is *fair* if, whenever infinitely many of the D_i's are equal to D and $D \rightarrow D'$, then infinitely many of the D_i's are equal to D'.

An algorithm *computes* a function $f : \Sigma^+ \rightarrow Y$ tolerating f Byzantine failures if, for all input strings $x \in X$, every fair execution $C_0, C_1, \ldots$ with $t \leq f$ failures starting from any initial configuration C_0 for x converges to $f(x)$, *i.e.*, there is an i such that for every $j > i$ and every state $(b, u_1, \ldots, u_d)$ of a correct agent in C_j, $\omega(b) = f(x)$. (A population protocol is a special case with $d = 0$.)

Instead of simulating a Turing machine directly, our community protocol simulates a pointer machine, which has a memory structured more like a community protocol's. Several types of pointer machines were developed independently. (See [5] for a survey.) Here, we describe the slightly revised version of Schönhage's **storage modification machine** (SMM) model [10] used in our construction. An SMM represents a single computer, not a distributed system. Its memory stores a finite directed graph of constant out-degree, with a distinguished node called the *centre*. The edges of the graph are called *pointers*. The edges out of each node are labelled by distinct *directions* drawn from a finite set Δ. Any string $x \in \Delta^*$ can be used as a reference to the node (denoted $p^*(x)$) that is reached from the centre by following the sequence of pointers labelled by the characters of x. The basic operations of an SMM allow the machine to create nodes, change pointers and follow paths of pointers. Formally, an SMM is specified by a finite input alphabet $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$, the set Δ, and a programme, which is a finite list of consecutively numbered instructions. Inputs to the SMM are finite strings from Σ^*. Programmes may use instructions of the following types.

- **new** creates a new node, makes it the centre, and sets all its outgoing pointers to the old centre.
- **recentre** x, where $x \in \Delta^+$, changes the centre of the graph to $p^*(x)$.
- **set** $x\delta$ **to** y, where $x, y \in \Delta^*$ and $\delta \in \Delta$, changes the pointer of node $p^*(x)$ that is labelled by δ to point to node $p^*(y)$.
- **if** $x = y$ **then goto** ℓ, where $x, y \in \Delta^*$, jumps to line ℓ if $p^*(x) = p^*(y)$.
- **input** $\ell_1, \ell_2, \ldots, \ell_r$, where $\ell_1, \ldots, \ell_r$ are line numbers, consumes the next input character (if there is one) and jumps to line ℓ_i if that character is σ_i.
- **output** o, where $o \in \{0, 1\}$, causes the machine to halt and output o.

When a node becomes unreachable from the centre, it can be dropped from the graph, since it plays no further role in the computation. Space complexity is measured by the maximum number of (reachable) nodes present at any one time during the execution. Our instruction set differs slightly from Schönhage's. We have omitted instructions that can be trivially implemented from ours and input/output instructions for online computation. We have given a separate name to the **recentre** instruction, because it is handled separately in our construction.

As shown by van Emde Boas [12], an SMM can simulate a Turing machine.

Theorem 2. *Any language decided by a Turing machine using $O(S \log S)$ space can be decided by an SMM using S nodes* [12].

We introduce a *nondeterministic SMM* (NSMM), adding choose instructions:

- **choose** ℓ_0, ℓ_1, where ℓ_0 and ℓ_1 are line numbers, causes the machine to transfer control either to line ℓ_0 or to line ℓ_1 nondeterministically.

We define acceptance of a string by an NSMM as for nondeterministic Turing machines: an NSMM accepts a string x if and only if *some* execution on input x outputs 1. The following theorem can be proved in exactly the same way as Theorem 2, so it will suffice to simulate an NSMM using community protocols.

Theorem 3. *Any language decided by a nondeterministic Turing machine using $O(S \log S)$ space can be decided by an NSMM using S nodes.*

3 Population Protocols Cannot Handle Byzantine Agents

A function $f : X \to Y$ is called *trivial* if the output can be determined from any single input character, *i.e.,* for any strings $x, y \in X$ that both contain a common character, $f(x) = f(y)$. Clearly, population protocols can compute any trivial function, even with Byzantine failures. We prove the converse also holds.

Theorem 4. *Any function computable by a population protocol tolerating one Byzantine agent is trivial.*

Proof. Consider an algorithm that computes f, tolerating one Byzantine failure. Let x and y be two strings with a common character a. We prove $f(x) = f(y)$. Let $n = |x| + |y| - 1$ and $p_1, p_2, \ldots, p_n$ be agents. Let E be a fair, failure-free execution where $p_1, \ldots, p_{|x|}$ have the characters of x as inputs and $p_{|x|+1}, \ldots, p_n$ have the characters of y, except for one copy of a, as inputs. Furthermore, assume p_1 has input a in E. (The input vector of E may or may not be in X.)

Let E' be the execution of $|x|$ agents $p_1, \ldots, p_{|x|}$ with input string x, where all agents except $p_{|x|}$ behave as in E, and $p_{|x|}$ behaves in a Byzantine manner, simulating the actions of all of the agents $p_{|x|}, \ldots, p_n$ in E. The fairness of E' follows from the fairness of E. Thus, in E', p_1 must stabilize to the output $f(x)$. Since p_1 cannot distinguish E from E', p_1 must stabilize to $f(x)$ in E too.

A symmetric argument (comparing E to an execution where $p_1, p_{|x|+1}, \ldots, p_n$ have input y and $p_{|x|+1}$ is Byzantine, simulating the actions of $p_2, \ldots, p_{|x|+1}$) shows p_1 stabilizes to $f(y)$ in E. Thus, $f(x) = f(y)$. □

4 Simulating an NSMM with a Community Protocol

We now outline the proof of Theorem 1, starting with a high-level overview. By Theorem 3, it suffices to design a community protocol that simulates an NSMM that uses $O(n)$ nodes. Due to space restrictions, we omit detailed proofs.

First, all agents organize themselves into a virtual ring. They then divide up into $O(f)$ clusters of approximately equal size using the ring structure. Each cluster collectively undertakes a simulation of the NSMM on the entire input string. In the simulation, each agent stores $O(f)$ nodes of the NSMM's graph data structure. Each agent can determine, by itself, whether another agent belongs to its own cluster, to prevent other clusters' Byzantine agents from interfering with its cluster's simulation. Thus, a majority of clusters will simulate the NSMM correctly. Each agent takes the majority output value of all clusters as its output.

Agents cannot know when the ring structure has stabilized, since a previously inert agent can begin interacting at any time. Thus, agents cannot wait until the structure has stabilized before starting to simulate the NSMM. Instead, they begin the simulations right away. Then, whenever the ring structure changes, the simulations are restarted. (Our algorithm ensures that Byzantine agents cannot cause spurious restarts.) We prove that simulations that begin after the ring structure has stabilized eventually converge to the correct answer.

The mechanism for restarting simulations is also used to handle the non-determinism of the NSMM. The system must output 1 if *some* execution of the NSMM outputs 1. Whenever the cluster's simulation outputs 0, it restarts, continuing to search for an execution that outputs 1. Some executions of the NSMM may run forever. To handle this, the simulation can make a non-deterministic decision to restart the simulation at any time. Fairness ensures that, if some execution of the NSMM outputs 1, correct clusters will eventually find it.

Building the Ring Structure. All agents can be imagined as forming an abstract ring, where the agents are ordered (clockwise) by their ids. The successor of the agent with the largest id is the agent with the smallest id. Our protocol makes use of this ring in several ways. In particular, an agent will sometimes have to to traverse the ring, meeting (almost) all the agents in order.

Initially, agents have no information about the abstract ring, so they must learn about it. Without failures, it would suffice for each agent to learn its successor's id. However, a Byzantine agent could disrupt traversals of the ring by lying about its successor or refusing to meet an agent traversing the ring. To avoid these problems, we build some redundancy into the agents' knowledge of the ring. Let $s = 8f + 4$. Each agent p has a *successors* field which stores the ids of p's s closest successors in the abstract ring that p has learned of so far.

An agent can learn about its successors by meeting them directly. If q is correct, it eventually meets all of its predecessors who can record q's id in their *successors* fields. However, if q is Byzantine, q may meet some of its predecessors, but not others. To facilitate orderly ring traversals, we ensure q's id eventually either appears in the *successors* fields of many of its predecessors (in which case no traversal skips q) or very few of them (in which case every traversal skips q).

To achieve this agreement about the ring structure, agents gossip about their successors. If p sees a new id x in another agent's *successors* field, it can non-deterministically choose to begin gathering evidence that x is a real id: it then remembers ids of agents that have x in their *successors* fields. If p meets $f + 1$ such agents, it concludes that at least one correct agent believes x is the id of a real agent, and can add x to its own *successors* field. The following lemma is an easy consequence of the fact that Byzantine agents cannot forge their own ids.

Lemma 1. *A correct agent's successors field always contains only ids of actual agents in the system and its successors field eventually stabilizes.*

The next lemma describes the degree of agreement achieved by gossiping when some *successors* fields have stabilized. We use $furthest(p)$ to denote the last id of *p.successors* (taken in clockwise order around the ring, starting from p).

Lemma 2. *Let $\mathcal{C}$ be a set of correct agents whose successors fields have stabilized at time T. For some $\mathcal{B} \subseteq \overline{\mathcal{C}}$ the following properties always hold after T.*
(1) For each correct agent p, after p.successors has stabilized, it contains the ids of all agents of $\mathcal{C} \cup \mathcal{B}$ that are located after p and before the agent with id $furthest(p)$ in the abstract ring.
(2) For each agent $q \notin \mathcal{C} \cup \mathcal{B}$, at most f of the agents in $\mathcal{C}$ have the id of q in their successors fields.

Traversing the Ring. Our construction will require an agent p to traverse the ring structure, performing some action upon each agent it meets. Ideally, p's traversal should visit every correct agent. However, p cannot wait forever to meet a particular agent, since that agent could be Byzantine. Thus, we are satisfied if p performs its action on almost all other agents. To do the traversal, p maintains a sorted list of s ids from one segment of the ring in an array field called $current[1..s]$. To advance the traversal, the first element of *current* is removed, all others are shifted left and a new id is added in $current[s]$. To begin a traversal, *p.current* is initialized to *p.successors*. Agent p then waits until it meets at least $s-f$ of those s agents, remembering each of *their successors* fields in its own state. (This is the most memory-intensive part of the construction: p must remember $\Theta(f^2)$ ids.) Agent p then adds to *p.current* the next id in the ring (beyond $p.current[s]$) that appears in at least $3f+2$ of those $s-f$ agents' *successors* fields. We show that this makes it impossible for the f Byzantine agents to misdirect the traversal. The traversal proceeds by iterating this process.

Naturally, traversals will not work properly before the ring structure has stabilized. However, the traversals do work correctly once *most successors* fields have stabilized. To make this more precise, consider any execution. Let T be the latest time such that changes are made to the *successors* fields of exactly $f+1$ correct agents after T. (This time exists, by Lemma 1.) Let $\mathcal{C}$ be the set of correct agents whose *successors* fields do not change after T. By definition of T, $|\mathcal{C}| \geq n-2f-1$. Define $\mathcal{B}$ to satisfy Lemma 2 for these choices of T and $\mathcal{C}$. The following lemma, which relies heavily on Lemma 2's guarantee that there is eventually a high degree of agreement among agents' *successors* fields, states that a traversal's advances are orderly if the traversal is started after T.

Lemma 3. *Suppose a correct agent p begins a traversal after T and after its own successors field has stabilized. Then, at all times, p.current contains all agents of $\mathcal{C} \cup \mathcal{B}$ that lie within a segment of the abstract ring (sorted in clockwise order). Furthermore, each advance will insert into $p.current[s]$ the id of the next agent of $\mathcal{C} \cup \mathcal{B}$ that is after (in clockwise order) the id previously stored in $p.current[s]$.*

Before advancing a traversal, p must meet $s-f$ of the agents whose ids are in *p.current*. Since the traversal is intended to accomplish some task, there may be an additional *external condition*, defined by the algorithm, which must be satisfied before the traversal can advance. The following lemma says that if that external condition is eventually satisfied, the traversal will indeed make progress.

Lemma 4. *If a correct agent p begins a traversal after T and after $p.successors$ has stabilized, then, in any suffix of the execution, either the external condition for advancing is unsatisfied infinitely often, or an advance eventually occurs.*

Global Resets. When the ring structure changes, agents recompute their clusters and re-start all simulated executions. Each time an agent p changes its *successors* field, it initiates a *global reset*, which tells (almost) every agent in the system to perform a *local reset* (described below). Agent p does a global reset by traversing the ring, giving a *reset request* to each agent it visits. To ensure Byzantine agents cannot cause spurious local resets, an agent performs a local reset only after receiving reset requests from $f+1$ different agents, not all of which can be Byzantine. Before the ring structure stabilizes, a global reset has unpredictable behaviour. However, Lemma 1 ensures correct agents eventually stop initiating global resets, and this implies the next lemma.

Lemma 5. *Each correct agent performs a finite number of local resets.*

By Lemma 3, global resets give reset requests to most agents after T. Thus, most agents receive $f+1$ reset requests after T and do a local reset:

Lemma 6. *Each correct agent in $\mathcal{C} \cup \mathcal{B}$ performs a local reset after T.*

Local Resets and Clustering. Local resets occur because the ring structure has changed. Agents then recompute the clusters and restart the simulations of the NSMM. Clustering (eventually) divides agents into $g = 4f + 3$ clusters of roughly the same size by chopping the ring into g sections. When an agent performs a local reset, it first recomputes the cluster boundaries. If the agent is the leader of its cluster (*i.e.*, the agent with the lowest id in the cluster), then it restarts the simulation of the NSMM within its cluster. When a non-leader does a local reset, it also asks the leader of its cluster to restart the simulation.

We first describe how an agent computes the boundaries of all clusters after its local reset. It initiates *two* traversals, called the *slow traversal* and the *swift traversal.* The slow traversal advances once every $g+1$ advances of the swift traversal. Whenever the swift traversal overtakes the slow traversal, the agent increments a counter and remembers the id of the agent where this crossing occurred as a cluster boundary. (As an analogy, the locations on a clock face where the minute hand passes the hour hand divide the face into 11 equal segments.)

By Lemma 6, all agents in $\mathcal{C}$ recompute clusters after T. Each agent remembers the cluster boundaries by storing the lowest id in each cluster. By Lemma 3, they will perform their slow and swift traversals identically in this final clustering, and therefore all compute identical cluster boundaries. In this final clustering after T, a cluster is called *correct* if it contains only agents in $\mathcal{C}$. We prove that correct clusters simulate the NSMM correctly after stabilization. A careful computation of exactly where the swift traversal passes the slow traversal proves these clusters are roughly equal in size.

Lemma 7. *Each cluster includes at least* $\frac{n-2f-1}{g} - 2$ *agents.*

Simulating an NSMM within a Cluster. A cluster's simulation of the NSMM is driven by its leader, p. Other agents in the cluster store the NSMM's graph data structure, which contains up to cn nodes for some constant c. Each agent stores up to $2cg$ nodes, ensuring that the collective capacity of the agents in the cluster is big enough to store cn nodes, by Lemma 7. A pointer to a node is represented by an id-index pair: the id identifies the agent storing the node and the index identifies one of the nodes stored by that agent. Agent p stores in its state a programme counter that describes what line of the NSMM programme it is simulating and keeps an id-index pair that locates the graph's centre. It simulates each instruction of the NSMM's programme one by one as follows.

To handle **input** instructions, p traverses the ring, starting from the agent with the lowest id. When the simulation reaches an **input** instruction, p advances the traversal and consumes the input of one agent. Ideally, the input symbols of all agents would be consumed. However, a traversal visits only those agents in $\mathcal{C} \cup \mathcal{B}$, so the simulation will miss inputs from any agents outside that set. Furthermore, any Byzantine agents in $\mathcal{B}$ might refuse to cooperate with the traversal, so p's input traversal must skip their inputs. Unfortunately, p does not know whether traversed agents belong to $\mathcal{C}$ or $\mathcal{B}$. So p guesses which agents to skip during the traversal, never skipping more than f of them. If p guesses incorrectly, it might choose *not* to skip an agent that is, in fact, Byzantine, and the Byzantine agent may refuse to meet p and provide an input character to the simulation. In this case, p gives up and restarts the simulation from scratch. Fairness guarantees that, if an accepting execution of the NSMM exists, the simulation will eventually find it when it correctly guesses which agents to skip. The overall effect of this input scheme is that the simulation might miss inputs from up to $3f+1$ agents (because there may be up to $2f+1$ agents outside $\mathcal{C} \cup \mathcal{B}$ and the simulation will skip up to f agents in $\mathcal{C} \cup \mathcal{B}$) and, among the agents from which inputs *are* received, up to f may be altered by Byzantine agents.

To simulate a **recentre** x instruction, p follows the pointers described by the string x one by one. Along the way, p stores the id-index pair locating each node visited. It advances along the path by waiting to meet the agent with that id, and then copies the new id-index pair from that agent's state. When p has finished traversing the path, it updates its *centre* field to point to the new centre.

To simulate a **set** $x\delta$ **to** y instruction, p first follows the pointers represented by the string x to find the agent q storing the node whose pointer must be updated. It then follows the pointers represented by y. Both paths are followed in the way described in the previous paragraph. Then, p changes the pointer stored in q to point to the node that it found at the end of the second path.

To simulate an **if** $x = y$ **then goto** ℓ' instruction, p follows both paths of pointers as described above, compares the resulting id-index pairs, and updates its programme counter accordingly.

For a **new** instruction, p locates an agent that stores fewer than $2cg$ nodes that are reachable from the centre. Agent p runs a garbage collection to detect unreachable nodes. First, p iterates across all agents in the cluster, marking all nodes within their memories as unvisited. Then, p executes a depth-first search (DFS) from the centre of the graph data structure. It then waits until it finds an agent q with a node that was not visited during the DFS and replaces that node by the old centre. There will always be such an agent q: the NSMM uses cn nodes in the worst case and there is space within the cluster to store $2cg(\frac{n-2f-1}{g}-2) = c(2n-4f-2g-2) \geq cn$ nodes. Agent p then creates a new centre node in q and updates its own *centre* field.

If the simulation reaches an **output**(1) instruction, then the simulation has successfully found an accepting execution, and no further simulation is necessary. The cluster leader p sets a field of its memory called *clusterOutput* to 1. (This field is initialized to 0 whenever p restarts a simulation.) If the simulation reaches an **output**(0) instruction, it is treated as a failed attempt to find an execution that leads to an output of 1, and p restarts the simulation of the NSMM, continuing to look for some other execution of the NSMM that outputs 1.

To simulate a **choose** ℓ_0, ℓ_1 instruction, p simply changes its programme counter to line ℓ_0 or line ℓ_1, making the choice nondeterministically.

To abandon simulated executions that never halt, p can non-deterministically choose to restart the simulated execution at any time.

Producing the Output. Consider any correct cluster. Let p be its leader. Each agent in the cluster performs a local reset after T, by Lemma 6, and will realize that it is a member of the cluster. The cluster's simulation of the NSMM is restarted when the last of these local resets occurs (and *p.clusterOutput* is reset to 0). After this, the simulations of executions by the NSMM within the cluster are correct since the cluster contains no Byzantine agents. Thus, *p.clusterOutput* will eventually be changed to 1 only if there is an execution of the NSMM that outputs 1 for some input string obtained from the actual input by deleting at most 3f+1 input characters and changing f of them (as described above). Conversely, if such an accepting execution of the NSMM exists, the cluster will eventually simulate one, by fairness. Thus, the preconditions imply that *p.clusterOutput* eventually converges to the correct output value.

Whenever an agent meets a cluster leader, it copies that leader's current *clusterOutput* field into its own memory. The output map ω of the community protocol has each agent output the value that appears in a majority of its own local copies of the clusters' outputs. Since $|\overline{\mathcal{C}}| \leq 2f+1$, a majority of the $g = 4f+3$ clusters are correct, so all agents converge to the correct output.

5 Conclusion

Many variants of the population protocol model have been studied. See [4] for a survey. Most of the work on population protocols assumes no failures. Delporte-Gallet *et al.* [6] gave a general construction that allows population protocols to tolerate $O(1)$ benign failures (halting failures and transient failures). Here, we used a similar technique of dividing agents into groups, each of which can simulate the entire protocol. However, the mechanisms needed to cope with Byzantine failures are much more involved. Angluin, Aspnes and Eisenstat

[2] described a population protocol that computes majority tolerating $O(\sqrt{n})$ Byzantine failures. However, it is designed for a much more restricted setting, where the scheduler chooses the next interaction randomly and uniformly.

Our model of mobile tiny devices tolerates Byzantine failures, while exploiting the full power of the agents' memory. This work found connections between modern mobile systems, automata theory and Turing machines, using pointer machines as a bridge. Many open questions remain. Can the small gap between the preconditions that are sufficient for our construction and the ones that are necessary for computation be closed? Can f Byzantine failures be tolerated if each agent can store fewer than $\Theta(f^2)$ ids? Because it is so general, our simulation would be extremely slow, but it might be possible to construct much faster simulations with additional assumptions (for example, with a probability distribution on the interactions). Finally, is the ordering on U required to cope with Byzantine failures, or would tests for equality between ids suffice (as in the failure-free case [7])?

Acknowledgements. The reviewers provided very helpful feedback for this version (and the future full version). This research was funded in part by NSERC.

References

1. Angluin, D., Aspnes, J., Diamadi, Z., Fischer, M., Peralta, R.: Computation in networks of passively mobile finite-state sensors. Distributed Computing 18(4), 235–253 (2006)
2. Angluin, D., Aspnes, J., Eisenstat, D.: A simple population protocol for fast robust approximate majority. Distributed Computing 21(2), 87–102 (2008)
3. Angluin, D., Aspnes, J., Eisenstat, D., Ruppert, E.: The computational power of population protocols. Distributed Computing 20(4), 279–304 (2007)
4. Aspnes, J., Ruppert, E.: An introduction to population protocols. In: Middleware for Network Eccentric and Mobile Applications. Springer, Heidelberg (2009)
5. Ben-Amram, A.M.: Pointer machines and pointer algorithms: an annotated bibliography, http://www2.mta.ac.il/~amirben
6. Delporte-Gallet, C., Fauconnier, H., Guerraoui, R., Ruppert, E.: When birds die: Making population protocols fault-tolerant. In: Proc. 2nd IEEE International Conference on Distributed Computing in Sensor Systems, pp. 51–66 (2006)
7. Guerraoui, R., Ruppert, E.: Even small birds are unique: Population protocols with identifiers. Technical Report CSE-2007-04, Dept of Computer Science and Engineering, York University (2007)
8. Mostefaoui, A., Rajsbaum, S., Raynal, M.: Conditions on input vectors for consensus solvability in asynchronous distributed systems. J. ACM 50(6), 922–954 (2003)
9. Numonyx. M25PX64 data sheet (2007), http://www.numonyx.com/Documents/Datasheets/M25PX64.pdf
10. Schönhage, A.: Storage modification machines. SIAM J. Comput. 9(3), 490–508 (1980)
11. Texas Instruments. MSP430 ultra-low-power microcontrollers (2008), http://focus.ti.com/lit/ml/slab034n/slab034n.pdf
12. van Emde Boas, P.: Space measures for storage modification machines. Inf. Process. Lett. 30(2), 103–110 (1989)

Multi-armed Bandits with Metric Switching Costs

Sudipto Guha[1,⋆] and Kamesh Munagala[2,⋆⋆]

[1] Department of Computer and Information Sciences, University of Pennsylvania,
Philadelphia PA 19104-6389
sudipto@cis.upenn.edu

[2] Department of Computer Science, Duke University, Durham NC 27708-0129
kamesh@cs.duke.edu

Abstract. In this paper we consider the stochastic multi-armed bandit with metric switching costs. Given a set of locations (arms) in a metric space and prior information about the reward available at these locations, cost of getting a sample/play at every location and rules to update the prior based on samples/plays, the task is to maximize a certain objective function constrained to a distance cost of L and cost of plays C. This fundamental and well-studied problem models several optimization problems in robot navigation, sensor networks, labor economics, etc.

In this paper we develop a general duality-based framework to provide the first $O(1)$ approximation for metric switching costs; the actual constants being quite small. Since these problems are Max-SNP hard, this result is the best possible. The overall technique and the ensuing structural results are independently of interest in the context of bandit problems with complicated side-constraints. Our techniques also improve the approximation ratio of the budgeted learning problem from 4 to $3+\epsilon$.

1 Introduction

A prevalent paradigm in sensor networks is to use and refine crude probabilistic models of sensed data at various nodes via judiciously sensing and transmitting values [13,12]; such a paradigm is used, for instance, in the ecological monitoring and forecasting application in the Duke forest [15]. Consider running an *extreme value* query in a sensor network [31]: The base station has crude prior models on the values sensed at each node in the network, and wishes to refine these estimates in order to select a node that is sensing extreme values. The key constraint in this process is energy: It costs energy to measure a value at a particular node, and it costs energy to transfer this process to a new node. The goal is to optimally refine the estimates from the perspective of the query, subject to a budget on the energy consumed in this process.

[⋆] Research supported in part by an Alfred P. Sloan Research Fellowship, and an NSF CAREER Award CCF-0644119.

[⋆⋆] Research supported by NSF via an Alfred P. Sloan Research Fellowship, a CAREER award and grant CNS-0540347.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 496–507, 2009.

Such an estimation problem with switching costs was traditionally motivated in several contexts: Price-setting under demand uncertainty [29], decision making in labor markets [23,27,4], and resource allocation among competing projects [3,17]. For instance [4,3], consider a worker who has a choice of working in k firms. A priori, she has no idea of her productivity in each firm. In each time period, her wage at the current firm is an indicator of her productivity there, and partially resolves the uncertainty in the latter quantity. Her expected wage in the period, in turn, depends on the underlying (unknown) productivity value. At the end of each time step, she can change firms in order to estimate her productivity at different places; however at the end of a "trial" period of duration C, she must choose one firm and stick with it. Her payoff is her expected wage in the finally chosen firm. The added twist is that changing firms does not come for free, and incurs a cost to the worker. What should the strategy of the worker be to maximize her (expected) payoff? A similar problem can be formulated from the perspective of a firm trying out different workers with a priori unknown productivity to fill a post – again, there is a cost to switch between workers. *We make the assumption that these costs define a metric.* This assumption is also clearly valid in a sensor network where the communication cost between nodes (and hence the cost to transfer the estimation process to a new node) depends on geographic distance.

The above problems can be modeled using the celebrated stochastic multi-armed bandit framework [6,7,32] with an additional *switching cost* constraint [1,2,4,3,9]: We are given a bandit with n independent arms[1]. When arm i is played, we observe a reward drawn from an underlying distribution D_i. A priori, this distribution is unknown; however, a prior $\mathcal{D}_i$ is specified over possible distributions. (For instance, if D_i is a Bernoulli $\{0, a\}$ distribution where the probability of a is an unknown value $t \in [0, 1]$, a possible prior $\mathcal{D}_i$ is the Beta distribution over the parameter t.) As the arm is played, the observed rewards resolve the prior into a posterior distribution over possible reward distributions. The state space of the arm $\mathcal{S}_i$ is the set of possible posterior distributions. Suppose the arm is in state $u \in \mathcal{S}_i$, which corresponds to a certain posterior distribution. Conditioned on this, each reward value is observed with a fixed probability, and causes transition to a different state $v \in \mathcal{S}_i$. Let the expected reward observed on playing in state u be R_u. Let $\mathbf{p}_{uv}$ denote the probability of transitioning to state v given a play in state u. We assume the state space $\mathcal{S}_i$ has polynomial size, and further, the that state transitions define a DAG rooted at the state ρ_i corresponding to the prior distribution $\mathcal{D}_i$. Since the states correspond to evolution of a prior distribution, we have a **martingale property** that $R_u = \sum_{v \in \mathcal{S}_i} R_v \mathbf{p}_{uv}$. The mapping from priors to the state space is standard [7], and we omit further discussion.

The cost model is as follows: If arm i is played in state $u \in \mathcal{S}_i$, the *play cost* is c_u. Further, the arms are located in a metric space with distance function ℓ. If the previous play was for arm i and the next play is for arm j, the *distance cost*

[1] In the context of a sensor network, think of the nodes as arms, and the values they sense as random variables.

incurred is ℓ_{ij}. A **policy** is an adaptive scheme for playing one arm per step, where a play can depend on the current state of each arm (which in turn depends on the reward sequence observed from previous plays for that arm). The start state of each arm is the root node ρ_i. The key assumption in all multi-armed bandit problems is that *only the state of an arm being played changes; all other arms retain their present state.* The policy therefore defines a decision tree of depth T over the joint state space of the n arms. At each leaf node of this tree, each arm i is in some final state f_i; the payoff of the policy for this leaf node is $\max_i R_{f_i}$. This corresponds to choosing the best arm (or best sensor node) at the end of the trial period. The goal is to compute the policy (or decision tree) for playing the arms whose expected payoff over all leaf nodes is maximized. There are three constraints:

1. The total play cost is at most C in all decision paths. This cost is paid per play by the policy. If the budget C is exhausted, the policy must stop.
2. The total distance cost is at most L in all decision paths. The policy pays this cost whenever it switches the arm being played.
3. The policy has poly-size specification, and can be computed in poly-time. The input has size $O(\sum_i |\mathcal{S}_i|^2)$ corresponding to the $\mathbf{p}_{uv}$, R_u, and c_u values.

The objective discussed above, which is the reward of the finally chosen arm is termed *future utilization*. We also consider *past utilization* (known in literature as the finite horizon average reward problem) where we seek to maximize the sum of the rewards obtained by the policy in all the plays it makes. The future utilization objective is already NP-Hard to maximize even when all play costs $c_u = 1$ and a single play reveals the full information about the arm [18], even in the absence of distance costs. The latter objective was used as a motivation for the orienteering problem [8] when the deterministic reward was available only for the first time a node is visited (in their own words) “as a first step towards tackling this general problem”.

Results and Techniques. In this paper, we build upon previous work and complete the development by providing an $O(1)$ approximation for the general stochastic setting for both the future and past utilization measures. The sensor network motivation also suggests that the policy must have compact description for switching between nodes, in particular: *Is there a strategy which is non-adaptive, that is, does not visit the same arm twice and achieves comparable reward to a fully adaptive optimal strategy*? We settle this question also in the affirmative, that is even before playing any arm we can fix an ordering and an associated strategy for each arm to explore.

Our algorithm is significantly different from our recent work [20], where we obtained a factor 4 approximation for the future utilization measure, with ℓ_{ij} is a function of the form $a_i + a_j$. Our solution technique there was to solve a linear programming relaxation and round the solution to a feasible policy while losing a constant factor. Unfortunately, that technique does not extend directly, since it is not clear how to round the natural LP relaxation even for the special case of *orienteering* [8,5,11]. We get around this difficulty by considering, via the

Lagrangean, a space of *relaxed* decision policies that violate the play cost constraint but respect the distance cost constraint. We show that this relaxed space has a $2+\epsilon$ approximation using the results for orienteering. At this point, we do not use the Lagrangean to solve the original relaxation, but instead, interpret the Lagrangean as yielding an amortized way of accounting for the reward. This interpretation yields a choice of the Lagrange multiplier and a simple policy that sequentially visits and plays the arms. We analyze this policy by converting the amortization of the reward into a *global* accounting scheme based on stopping condition rather than arm by arm. This global method shows the right choice of the Lagrange multiplier in retrospect.

For the future utilization version, this technique yields a $4+\epsilon$-approximation, which reduces to $3+\epsilon$ in the absence of switching costs. We therefore also improve the factor 4 approximation for budgeted learning from our previous paper [20]. Our algorithmic technique is similar to that used in [22] in the context of the entirely different *restless bandit* problems (where the state of an arm varies even if it is not played) – this showcases the power of this technique in handling diverse stochastic bandit problems even in the presence of arbitrary side constraints.

Related Work. The future utilization problem without switching costs (a.k.a. *budgeted learning*), is well-known in literature [6,19,20,26,30,32]. Our previous paper [20] presents the first 4-approximation algorithm using an LP based approach. Goel *et al.* [19] extend this to define a *ratio index* policy that computes indexes for each arm in isolation and plays the arm with the highest index. They show that this policy is a 4.54 approximation. This is analogous to the famous *Gittins index* [16,7] that is optimal for the discounted past utilization setting. We show the precise connection between these index policies and our technique, and as a side-effect, show that *not only is our technique generalizable to incorporate switching costs (unlike indices), but in addition, the computation time needed for our algorithm is not any worse than the time needed to compute such indices.* We note that since our problems generalize orienteering [8,5,11] which is already Max-SNP hard, constant factor approximations are best possible.

Though it is widely acknowledged [4,9] that the right model for bandit problems should have a cost for switching arms, the key hurdle is that even for the discounted reward past utilization version, index policies stop being optimal. Banks and Sundaram [4] provide an illuminating discussion of this aspect, and highlight the technical difficulty in designing reasonable policies. In fact, *to the best of our knowledge there was no prior theoretical results on stochastic multi-armed bandit with metric switching costs.* We note that this problem is very different from the Lipshitz MAB considered in [25] where the *rewards* of the bandits correlate to their position in the metric space.

The multi-armed bandit has an extensively studied problem since its introduction by Robbins in [28]. From this starting point the literature diverges into a number of (often incomparable) directions, based on the objective and the information available. In context of theoretical results, the typical goal [10,24,14] has been to assume that the agent has absolutely no knowledge of ρ_i (model free assumption) and then the task has been to minimize the “regret” or the lost reward, that is comparing the performance of the agent to a superagent who has full knowledge, and plays the best arm from the start. It is easy to show that with sufficiently small cost budgets, the regret against an all-powerful adversary is too large, and hence the need for the Bayesian approach [6,7,32] considered here. The difficulty in the Bayesian approach is computational – these problems can be solved by dynamic programming over the joint (product) state space $\prod_i |\mathcal{S}_i|$ which is exponential in the number of arms. We therefore seek approximation algorithms in this setting.

Roadmap. Due to space limitations, we present a $4 + \epsilon$ approximation to the future utilization measure. The other results can be found in the full version of this paper [21].

2 Future Utilization Problem: Notation

Recall that we are given n arms. The distance cost of switching from arm i to arm j is $\ell_{ij} \in \mathcal{Z}^+$; this defines a metric. The arm i has a state space $\mathcal{S}_i$, which is a DAG with root (or initial state) ρ_i. If the arm i is played in a state $u \in \mathcal{S}_i$, it transitions to state $v \in \mathcal{S}_i$ with probability $\mathbf{p}_{uv}$; the play costs $c_u \in \mathcal{Z}^+$ and yields expected reward R_u. Since the states correspond to evolution of a prior distribution, we have a **martingale property** that $R_u = \sum_{v \in \mathcal{S}_i} R_v \mathbf{p}_{uv}$.

The system starts at an arm i_0. A policy, given the outcomes of the actions so far (which decides the current states of all the arms), makes one of the following decisions (i) play the arm it is currently on; (ii) play a different arm (paying the switching cost); or (iii) stop and choose the current arm i in state $u \in \mathcal{S}_i$, obtaining its reward R_u as the payoff. Any policy is subject to **rigid** constraints that the total play cost is at most C and the total distance cost is at most L on all decision paths. Although our algorithm will only stop and choose an arm i when it is already at arm i, we allow the policies to choose any arm as long as the arm was visited sometime in the past.

The **goal** is to find the policy with maximum expected payoff. We focus on constructing, in polynomial time, policies with possibly implicit poly-size specification in the input size $O(\sum_{i=1}^{n} |\mathcal{S}_i|^2)$. Let OPT denote both the optimal solution as well as its value.

3 Lagrangean Relaxation

We describe a sequence of relaxations to the optimal policy. We first delete all arms j such that $\ell_{i_0 j} > L$. No feasible policy can reach such an arm without exceeding the distance cost budget. Let $\mathcal{P}$ denote the set of policies on the remaining arms that can perform one of three actions: (1) Choose current arm and ignore this arm in the future; (2) Play current arm; or (3) Switch to different (unchosen) arm. Such policies have no constraints on the play costs, but are required to have distance cost L on all decision paths. Further, a policy $P \in \mathcal{P}$ can choose multiple arms, and obtain payoff equal to the sum of the rewards

R_u of the corresponding states. Any arm can be chosen at most once, and if so, cannot be played in the future. The original problem had a stricter requirement that only one arm could be chosen, and if a choice is made, the plays stop.

Given a policy $P \in \mathcal{P}$ define the following quantities in expectation over the decision paths: Let $I(P)$ denote the expected number of arms finally chosen; $R(P)$ be the expected reward obtained from the chosen arms; $C(P)$ denote the expected play cost. Note that any policy $P \in \mathcal{P}$ needs to have distance cost at most L on *all* decision paths. Consider the following optimization problem:

$$(M1): \qquad \max_{P\in\mathcal{P}} \left\{ R(P) \,\middle|\, \frac{C(P)}{C} + I(P) \le 2 \right\}$$

Proposition 1. *OPT is feasible for* $(M1)$.

Proof. We have $I(OPT) = 1$, $C(OPT) \le C$; since $OPT \in \mathcal{P}$, this shows it is feasible for $(M1)$.

Let the optimum solution of $(M1)$ be OPT' and the corresponding policy be P^* such that $R(P^*) = OPT' \ge OPT$. Note that P^* need not be feasible for the original problem, since, for instance, it enforces the play cost constraint only in expectation over the decision paths. We now consider the Lagrangean of the above for $\lambda > 0$, and define the problem $M2(\lambda)$:

$$M2(\lambda): \qquad \max_{P\in\mathcal{P}} \; f_\lambda(P) = 2\lambda + \max_{P\in P} \left(R(P) - \lambda \left(\frac{C(P)}{C} + I(P) \right) \right)$$

Definition 1. $V(\lambda) = \max_{P\in P} \left(R(P) - \lambda \left(\frac{C(P)}{C} + I(P) \right) \right)$.

We re-iterate that the only constraint on the set of policies $\mathcal{P}$ is that the distance cost is at most L on all decision paths. The critical insight, which explicitly uses the fact that in the MAB the state of an inactive arm does not change, is the following:

Lemma 1. *For any* $\lambda \ge 0$*, given any* $P \in \mathcal{P}$*, there exists a* $P' \in \mathcal{P}$ *that never revisits an arm that it has already played and switched out of, such that* $f_\lambda(P') \ge f_\lambda(P)$.

Proof. We will use the fact that $\mathcal{S}_i$ is finite in our proof. Suppose $P \in \mathcal{P}$ revisits an arm. Consider the deepest point in the decision tree P where at decision node α, it is at arm i, took a decision to move to arm j (child decision node β) and later, on one of the decision paths descending from this point, revisits arm i. Call (α, β) to be "offending".

Note that starting at decision node β, the decision tree P_β satisfies the condition that no arm is revisited. Compress all the actions corresponding to staying in arm j starting at β to a single decision "super-node". This "super node" corresponds to the outcome of plays on arm j starting at β. But the *critical* part is that the plays of the arm j does not affect the state of the other arms (due to independence and the fact that inactive arms do not change state). Also the subtrees that were children of this super-node do not have any action related to arm j by assumption. These subtrees are followed with different probabilities that sum up to 1.

Therefore we can choose the subtree T of the super-node which has the maximum value of $f_\lambda(T)$, and use this subtree irrespective of the outcomes of arm j. This yields a new policy P' so that $f_\lambda(P') \geq f_\lambda(P)$, and furthermore, the distance cost of P' is at most L in all decision paths, so that P' is feasible. By the repeated application of the above, the subtree P_β can be changed to a *path* over super-nodes (which correspond to actions at an arm which is never revisited), without decreasing $f_\lambda(P)$. We will refer to P_β as a "path" in this sense.

Now suppose the arm i corresponding to the play at decision node α, which is the parent of decision node β, is played in this path P_β. Consider moving the entire super-node corresponding to arm i in P_β, to just after node α, so that arm i is played according to this super-node before visiting arm j (as dictated by the decision node β). By independence of arms, the intermediate plays in P_β before reaching arm i do not affect the state of arm i. Hence the move preserves the states of all the arms. Further, the distance cost of the new policy is only smaller, since the cost of switching into and out of arm i is removed. Note that (α, β) is not "offending" any more, and we have not introduced any new offending pairs of decision nodes. By repeated application, the proposition follows.

Note that the above is *not* true for policies restricted to be feasible for $(M1)$. This is because the step where we use sub-tree T regardless of the outcome of the super-node corresponding to j need not preserve the (implicit) constraint $I(P) \leq 2$, since this depends on whether node j is chosen or not by the super-node. The Lagrangean $M2(\lambda)$ makes the overall objective additive in the (new) objective values of the super-nodes, with the only constraint being that the distance cost is preserved. Since this cost is preserved in each decision branch, it is preserved by using the best sub-tree T regardless of the outcome of the super-node corresponding to j.

Let $\mathcal{P}_i$ denote the set of all policies that play only arm i. Such a policy at any point in time can (i) play the arm; (ii) stop and choose the arm; or (iii) stop and not choose the arm. Note that the policy can choose the arm at most once in any decision path, but is otherwise unconstrained. Further note that expected play cost $C(P)$ and reward $R(P)$ are defined for such a policy, but not distance cost. The start state of the policy is $\rho_i \in \mathcal{S}_i$, and the state space is $\mathcal{S}_i$.

Definition 1 $Q_i(\lambda) = \max_{P \in \mathcal{P}_i} R(P) - \lambda \left(\frac{C(P)}{C} + I(P) \right)$.

Lemma 2. *$Q_i(\lambda)$ can be computed in time polynomial in $|\mathcal{S}_i|$.*

Proof. This is a straightforward dynamic program. Let $\text{Gain}(u)$ to be the maximum of the objective of the single-arm policy conditioned on starting at $u \in \mathcal{S}_i$. If u has no children, then if we "play" at node u, then there is no further actions for the policy and so the $\text{Gain}(u) = -\lambda \frac{c_u}{C}$ in this case. Stopping and not doing anything corresponds to $\text{Gain}(u) = 0$. "Choosing" this arm corresponds to a gain of $R_u - \lambda$. Therefore we set $\text{Gain}(u) = \max\{0, R_u - \lambda\}$ in this case.

If u had children, playing would correspond to a gain of $-\lambda\frac{c_u}{C} + \sum_v \mathbf{p}_{uv}$ Gain(v). Choosing the arm would correspond to gain $R_u - \lambda$. Therefore we have:

$$\text{Gain}(u) = \max\left\{0, \qquad -\lambda\frac{c_u}{C} + \sum_v \mathbf{p}_{uv} gain(v), \qquad R_u - \lambda\right\}$$

We have Gain$(\rho_i) = Q_i(\lambda)$. This will clearly take polynomial time in $|\mathcal{S}_i|$.

Recall that the optimal solution to $M2(\lambda)$ is $2\lambda + V(\lambda)$. An immediate consequence of Lemma 1 is the following:

Corollary 1. *Define a graph $G(V,E)$, where node $i \in V$ corresponds to arm i. The distance between nodes i and j is ℓ_{ij}, and the reward of node i is $Q_i(\lambda)$. The optimum solution $V(\lambda)$ of $M2(\lambda)$ is the optimal solution to the orienteering problem on G starting at node i_0 and respecting rigid distance budget L.*

Proof. Consider any n-arm policy $P \in \mathcal{P}$. By Lemma 1, the decision tree of the policy can be morphed into a sequence of "super-nodes", one for playing each arm, such that the decision about which arm to play next is independent of the outcomes for the current arm. The policy maximizing $f_\lambda(P)$ will therefore choose the best policy in $\mathcal{P}_i$ for each single arm as the "super-node" (obtaining objective value precisely $Q_i(\lambda)$), and visit these subject to the constraint that the distance cost is at most L. This is precisely the orienteering problem on the graph defined above.

Theorem 1. *For some constant $\epsilon > 0$, assume $Q_i(\lambda) \in \left[\epsilon\frac{\lambda}{n}, \lambda\right]$ for all arms i. Then, $V(\lambda)$ has a $2(1+\epsilon)$-approximation (that we will denote $W(\lambda)$) in time polynomial in $\sum_{i=1}^n |\mathcal{S}_i|$.*

Proof. This follows from the 3-approximation algorithm of Chekuri *et al.* [11] for the orienteering problem. Their result holds only when the $Q_i(\lambda)$ are integers in a polynomial range. If the $Q_i(\lambda) \in \left[\epsilon\frac{\lambda}{n}, \lambda\right]$, then round them in powers of $(1+\epsilon)$ to make them integers, and then apply the factor $2+\epsilon$ approximation algorithm.

4 Choosing and Interpreting the Penalty λ

Recall that optimal value to the problem $(M1)$ is at least OPT. We first relate OPT to the optimal value $2\lambda + V(\lambda)$ of the problem $M2(\lambda)$. We ignore the factor $(1+\epsilon)$ in the result in Theorem 1.

Lemma 3. *For any $\lambda \geq 0$, we have $2\lambda + V(\lambda) \geq OPT$.*

Proof. This is simply weak duality: For the optimal policy P^* to $(M1)$, we have $I(P^*) + C(P^*)/C \leq 2$. Since this policy is feasible for $M2(\lambda)$, the claim follows.

Lemma 4. *For constant $\delta > 0$, we can choose a λ^* in polynomial time so that for the resulting orienteering tour (of value $W(\lambda)$) on the subset S of arms constructed by the approximation algorithm, we have: (1) $\lambda^* \geq \frac{OPT}{4}(1-\delta)$; and (2) $W(\lambda) = \sum_{i \in S} Q_i(\lambda^*) \geq \frac{OPT}{4}(1-\delta)$.*

Proof. First note that as λ increases, the value of any policy $P \in \mathcal{P}_i$ reduces, which implies $Q_i(\lambda)$ decreases. For $\lambda \geq \sum_{i=1}^{n} \sum_{u \in \mathcal{S}_i} R_u$, the optimal policy for arm i does not play the arm, so that $Q_i(\lambda) = 0$ for all arms i. For $\lambda = 0$, we have $Q_i(\lambda) > 0$. This implies the following algorithm to find λ^* such that $W(\lambda^*) \approx \lambda^*$. Start with $\lambda = \sum_{i=1}^{n} \sum_{u \in \mathcal{S}_i} R_u$ and decrease it in powers of $(1+\epsilon)$. For each value of λ, compute all $Q_i(\lambda)$. Throw out all arms with $Q_i(\lambda) < \epsilon \frac{\lambda}{n}$. This decreases the value of any orienteering tour by at most $\epsilon\lambda$. Now, if some arm i reachable from the root has $Q_i(\lambda) > \lambda$, then the value of the tour is larger than λ, so that λ can be increased further. Otherwise, we have $Q_i(\lambda) \in \left[\epsilon \frac{\lambda}{n}, \lambda\right]$ for all i Theorem 1 yields a 2 approximation to the reward. Let the value of this approximation be $W(\lambda)$. We have: $W(\lambda) \geq \frac{1}{2}(V(\lambda) - \epsilon\lambda)$. By Lemma 3:

$$W(\lambda) \geq \frac{1}{2}(V(\lambda) - \epsilon\lambda) \geq \frac{1}{2}(OPT - (2+\epsilon)\lambda) \Rightarrow (1 + \frac{\epsilon}{2})\lambda + W(\lambda) \geq \frac{OPT}{2}$$

As λ is decreased, we encounter a point where: $W(\lambda) \leq \lambda$, and for $\lambda' = \lambda(1-\epsilon)$, we have $W(\lambda') > \lambda'$. Set $\lambda^* = \lambda'$. For these choices, we must have: $\lambda = \lambda'(1+\epsilon) \geq \frac{OPT}{4}(1-\epsilon)$, and $W(\lambda') \geq \frac{OPT}{4}(1-\epsilon)$. Now choose $\lambda^* = \lambda'$ and δ as a appropriate function of ϵ.

As in Theorem 1, we will ignore the constant $\delta > 0$ in the remaining analysis.

4.1 Amortized Accounting of the Reward

Consider the single-arm policy $P_i^* \in \mathcal{P}_i$ that corresponds to the value $Q_i(\lambda^*)$. This policy performs one of three actions for each state $u \in \mathcal{S}_i$: (i) Play the arm at cost c_u; (ii) Choose the arm in the current state, and stop; or (iii) Stop, but do not choose the arm. For this policy, note that $R(P_i^*) = Q_i(\lambda^*) + \lambda^*(I(P_i^*) + C(P_i^*)/C)$. This implies the reward $R(P_i^*)$ of this policy can be amortized, so that for state $u \in \mathcal{S}_i$, the reward is collected as follows:

1. An upfront reward $Q_i(\lambda^*)$ when the play initiates at the root $\rho \in \mathcal{S}_i$.
2. A reward of $\lambda^* c_u / C$ for playing the arm in $u \in \mathcal{S}_i$.
3. A reward of λ^* when the policy stops and chooses the arm in state $u \in \mathcal{S}_i$.

Lemma 5. *If the arm i is played starting at the root $\rho \in \mathcal{S}_i$ according to policy $\mathcal{P}_i^*$, and the reward is generated according to the above amortized method, then the expected reward of the policy is precisely $Q_i(\lambda^*)$.*

Note that the above is not true if the policy P_i^* is executed incompletely.

5 Final Policy and Analysis

The final policy is now extremely simple, and shown in Figure 1. For analysis, first make the following modification to the final policy: If the stopping condition (2d) is encountered and the current arm is i, play the policy P_i^* to completion and then stop and choose arm i. This exceeds the budget C. The original policy

Policy for Future utilization

1. Define the problem ($M1$). Approximately solve the Lagrangean $M2(\lambda^*)$ for λ^* computed in Lemma 4. The policy is an orienteering tour of length at most L on a subset S^* of arms, combined with a single arm policy P_i^* for each arm $i \in S^*$ whose value is $Q_i(\lambda^*)$.
2. Traverse the orienteering tour, and for each arm i encountered, play the arm according to policy P_i^*, with the stopping conditions:
   (a) If P_i^* chooses arm i, then the final policy stops and chooses arm i.
   (b) If P_i^* stops without choosing arm i, move to the next arm on tour.
   (c) If the arms in S^* are exhausted, then stop.
   (d) If cost budget C exhausted, stop and choose the current arm i.

Fig. 1. The Final Policy for Future Utilization

differs from the modified policy in that when the budget C is exhausted and the current state is $u \in \mathcal{S}_i$, the original policy chooses the current arm i and obtains reward R_u, while the new policy plays policy P_i^* to completion (and may or may not choose the arm on different decision paths). However, by the **martingale** property of the rewards, the expected future reward of P_i^* even if it chooses the arm on all decision paths is the same as the current reward R_u of state $u \in \mathcal{S}_i$, so that the original policy has at least the expected reward of the modified policy. This analysis is also present in [20]. Therefore, we will focus on analyzing the modified policy that in Step (2d), plays the current arm i to completion according to policy P_i^*. Ignoring the ϵ, δ in Theorem 1 and Lemma 4:

Lemma 6. *The modified final policy, that in Step (2d), plays the current arm i to completion according to policy P_i^*, has reward at least $OPT/4$.*

Proof. For the modified policy, the reward can be accounted for in the amortized sense discussed in Section 4.1. First note that when the policy visits arm i and starts executing the policy P_i^* starting at state $\rho_i \in \mathcal{S}_i$, this policy executes to completion and the execution is independent of the execution of previously played arms. This implies that when the final policy visits arm i, the reward from this arm can be accounted as: Give the system a reward of $Q_i(\lambda^*)$. For every play of cost c, give a reward of $\lambda^* \frac{c}{C}$, and if the arm is chosen (so that the final policy stops and chooses this arm), give a reward of λ^*. It is clear from Lemma 5 that the expected reward of the final policy according to this amortization is the same as the expected reward of the finally chosen arm (*i.e.*, the expected reward of the policy). Now, in order to estimate this expected reward, we take a global view of this amortization. Instead of summing the amortized rewards over the random set of arms encountered, we sum these rewards based on the stopping condition the policy encounters. There are three cases:

1. The final policy exhausts all arms. In this case, the accrued amortized reward is at least $\sum_{i \in S} Q_i(\lambda^*) \geq OPT/4$.
2. The final policy runs out of budget C. In that case, since the reward per play is $\frac{\lambda^*}{C}$ per unit cost, the accrued reward is at least $\lambda^* \geq \frac{OPT}{4}$.

3. The final policy chooses some arm i. However, the amortized reward per choice is $\lambda^* \geq \frac{OPT}{4}$.

Since the above cases are exhaustive, the modified final policy yields amortized reward at least $\frac{OPT}{4}$, which is the same as the actual reward.

As shown above, the reward of the final policy was only larger before we modified Step (2d). Therefore, we have the following theorem:

Theorem 2. *For the multi-armed bandit problem with metric switching costs under future utilization, there exists a poly-time computable ordering of the arms and a policy for each arm, such that a solution which plays the arms using these policies in that fixed order without revisiting any arm, has reward at least* $\frac{1}{4} - \epsilon$ *times that of the best adaptive policy, where* $\epsilon > 0$ *is any small constant.*

The above implies an improved approximation to the *budgeted learning* problem [20,19], which is the special case where all distances are zero (so that there is no cost for switching between arms).

Corollary 2. *The budgeted learning problem has a* $3 + \epsilon$ *approximation.*

Proof. In this case, the value $V(\lambda)$ of the problem $M2(\lambda)$ is computable in polynomial time, since it is precisely $\sum_{i=1}^{n} Q_i(\lambda)$. Since $2\lambda + V(\lambda) \geq OPT$ by Lemma 3, choose λ^* so that $V(\lambda^*) \geq \frac{OPT}{3}$ and $\lambda^* \geq \frac{OPT}{3}$. Now the policy in Figure 1 is trivially a $3 + \epsilon$ approximation.

Acknowledgments. We thank Ashish Goel and Peng Shi for helpful discussions.

References

1. Agrawal, R., Hegde, M.V., Teneketzis, D.: Asymptotically efficient adaptive allocation rules for the multiarmed bandit problem with switching costs. IEEE Transactions on Optimal Control 33, 899–906 (1988)
2. Asawa, M., Teneketzis, D.: Multi-armed bandits with switching penalties. IEEE Transactions on Automatic Control 41(3), 328–348 (1996)
3. Banks, J.S., Sundaram, R.K.: Denumerable-armed bandits. Econometrica 60(5), 1071–1096 (1992)
4. Banks, J.S., Sundaram, R.K.: Switching costs and the gittins index. Econometrica 62(3), 687–694 (1994)
5. Bansal, N., Blum, A., Chawla, S., Meyerson, A.: Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In: STOC, pp. 166–174 (2004)
6. Berry, D.A., Fristed, B.: Bandit problems; Sequential Allocation of Experiments. Chapman & Hall, New York (1985)
7. Bertsekas, D.: Dynamic Programming and Optimal Control, 2nd edn. Athena Scientific (2001)
8. Blum, A., Chawla, S., Karger, D.R., Lane, T., Meyerson, A., Minkoff, M.: Approximation algorithms for orienteering and discounted-reward TSP. SIAM J. Comput. 37(2), 653–670 (2007)

9. Brezzi, M., Lai, T.-L.: Optimal learning and experimentation in bandit problems. Journal of Economic Dynamics and Control 27(1), 87–108 (2002)
10. Cesa-Bianchi, N., Freund, Y., Haussler, D., Helmbold, D.P., Schapire, R.E., Warmuth, M.K.: How to use expert advice. J. ACM 44(3), 427–485 (1997)
11. Chekuri, C., Korula, N., Pál, M.: Improved algorithms for orienteering and related problems. In: SODA, pp. 661–670 (2008)
12. Chu, D., Deshpande, A., Hellerstein, J., Hong, W.: Approximate data collection in sensor networks using probabilistic models. In: ICDE (2006)
13. Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J.M., Hong, W.: Model-driven data acquisition in sensor networks. In: VLDB (2004)
14. Flaxman, A., Kalai, A., McMahan, H.B.: Online convex optimization in the bandit setting: Gradient descent without a gradient. In: Annual ACM-SIAM Symp. on Discrete Algorithms (2005)
15. Flikkema, P.G., Agarwal, P.K., Clark, J.S., Ellis, C.S., Gelfand, A., Munagala, K., Yang, J.: Model-driven dynamic control of embedded wireless sensor networks. In: Proc. 6th Intl. Conf. on Computational Science (3), pp. 409–416 (2006)
16. Gittins, J.C., Jones, D.M.: A dynamic allocation index for the sequential design of experiments. In: Progress in statistics (European Meeting of Statisticians) (1972)
17. Gittins, J.C.: Multi-Armed Bandit Allocation Indices. Wiley, New York (1989)
18. Goel, A., Guha, S., Munagala, K.: Asking the right questions: Model-driven optimization using probes. In: Proc. of the 2006 ACM Symp. on Principles of Database Systems (2006)
19. Goel, A., Khanna, S., Null, B.: The ratio index for budgeted learning, with applications. In: Proc. ACM-SIAM Symp. on Discrete Algorithms, SODA (2009)
20. Guha, S., Munagala, K.: Sequential design of experiments via linear programming. CoRR, arxiv:0805.0766 (2008); preliminary version in STOC 2007
21. Guha, S., Munagala, K.: Multi-armed bandits with metric switching costs (2009), http://www.cs.duke.edu/~kamesh/metric.pdf
22. Guha, S., Munagala, K., Shi, P.: Approximation algorithms for restless bandit problems. CoRR, abs/0711.3861 (2007); preliminary version in SODA 2009
23. Jovanovich, B.: Job-search and the theory of turnover. J. Political Economy 87, 972–990 (1979)
24. Kalai, A., Vempala, S.: Efficient algorithms for online decision problems. In: Proc. of 16th Conf. on Computational Learning Theory (2003)
25. Kleinberg, R., Slivkins, A., Upfal, E.: Multi-armed bandit problems in metric spaces. In: Proc. 40th ACM Symposium on Theory of Computing (2008)
26. Madani, O., Lizotte, D.J., Greiner, R.: Active model selection. In: UAI 2004: Proc. 20th Conf. on Uncertainty in Artificial Intelligence, pp. 357–365 (2004)
27. Mortensen, D.: Job-search and labor market analysis. In: Ashenfelter, O., Layard, R. (eds.) Handbook of Labor Economics, vol. 2, pp. 849–919. North Holland, Amsterdam (1985)
28. Robbins, H.: Some aspects of the sequential design of experiments. Bulletin American Mathematical Society 55, 527–535 (1952)
29. Rothschild, M.: A two-armed bandit theory of market pricing. J. Economic Theory 9, 185–202 (1974)
30. Schneider, J., Moore, A.: Active learning in discrete input spaces. In: 34th Interface Symp. (2002)
31. Silberstein, A., Munagala, K., Yang, J.: Energy-efficient extreme value estimation in sensor networks. In: SIGMOD (2006)
32. Wetherill, G.B., Glazebrook, K.D.: Sequential Methods in Statistics (Monographs on Statistics and Applied Probability). Chapman & Hall, London (1986)

Algorithms for Secretary Problems on Graphs and Hypergraphs

Nitish Korula[1,⋆] and Martin Pál[2]

[1] Dept. of Computer Science, University of Illinois, Urbana, IL 61801
nkorula2@illinois.edu
[2] Google Inc., 76 9th Avenue, New York, NY 10011
mpal@google.com

Abstract. We examine online matching problems with applications to Internet advertising reservation systems. Consider an edge-weighted bipartite graph $G(L \cup R, E)$. We develop an 8-competitive algorithm for the following *secretary* problem: Initially given R, and the size of L, the algorithm receives the vertices of L sequentially, in a random order. When a vertex $l \in L$ is seen, all edges incident to l are revealed, together with their weights. The algorithm must immediately either match l to an available vertex of R, or decide that l will remain unmatched.

In [5], the authors show a 16-competitive algorithm for the transversal matroid secretary problem, which is the special case with weights on vertices, not edges. (Equivalently, one may assume that for each $l \in L$, the weights on all edges incident to l are identical.) We use a very similar algorithm, but simplify and improve the analysis to obtain a better competitive ratio for the more general problem. Our analysis is easily extended to obtain competitive algorithms for a class of similar problems, such as to find disjoint sets of edges in hypergraphs where edges arrive online. We also introduce secretary problems with adversarially chosen *groups*.

Finally, we give a $2e$-competitive algorithm for the secretary problem on graphic matroids, where, with edges appearing online, the goal is to find a maximum-weight acyclic subgraph of a given graph.

1 Introduction

Many optimization problems of interest can be phrased as picking a maximum-weight independent subset from a ground set of elements, for a suitable definition of independence. A well-known example is the (Maximum-weight) Independent Set problem on graphs, where we wish to find a set of vertices, no two of which are adjacent. A more tractable problem in this setting is the Maximum-weight Matching problem, in which we wish to find a set of edges such that no two edges share an endpoint.

In the previous examples, independent sets are characterized by forbidding certain *pairs* of elements from the ground set. A somewhat related, but different notion of independence comes from the independent sets of a matroid. For

[⋆] This work was done while the author was at Google Inc., NY.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 508–520, 2009.

example, in the uniform matroid of rank k, any set of at most k elements is independent. For graphic matroids, a set of edges in an undirected graph is independent if and only if it does not contain a cycle; the optimization goal is to find a maximum-weight acyclic subgraph of a graph G. In transversal matroids, a set of left-vertices of a bipartite graph is independent if and only if there is a matching that matches each vertex in this set to some right-vertex.

In many applications, the elements of the ground set and their weights are not known in advance, but arrive online one at a time. When an item arrives, we must immediately decide to either irrevocably accept it into the final solution, or reject it and never be able to go back to it again. We will be interested in competitive analysis, that is, comparing the performance of an online algorithm to an optimal offline algorithm which is given the whole input in advance. In this setting, even simple problems like selecting a single maximum-weight element become difficult, because we do not know if elements that come in the future will have weight significantly higher or lower than the element currently under consideration. If we make no assumptions about the input, any algorithm can be fooled into performing arbitrarily poorly by offering it a medium-weight item, followed by a high-weight item if it accepts, and a low-weight item if it rejects. To solve such problems, which frequently arise in practice, various assumptions are made. For instance, one might assume that weights are all drawn from a known distribution, or (if independent sets may contain several elements) that the weight of any single element is small compared to the weight of the best independent set.

One useful assumption that can be made is that the elements of the ground set appear in a random order. The basic problem in which the goal is to select the maximum-weight element is well known as the *Secretary Problem*. It was first published by Martin Gardner in [7], though it appears to have arisen as folklore a decade previously [6]. An optimal solution is to observe the first n/e elements, and select the first element from the rest with weight greater than the heaviest element seen in the first set; this algorithm gives a $1/e$ probability of finding the heaviest element, and has been attributed to several authors (see [6]).

Several problems have been studied in this random permutation model; these are often called secretary-type problems. Typically, given a random permutation of elements appearing in an online fashion, the goal is to find a maximum-weight independent set. For example, Kleinberg [10] gives a $1 + O(1/\sqrt{k})$-competitive algorithm for the problem of selecting at most k elements from the set to maximize their sum. Babaioff *et al.* [3] give a constant-competitive algorithm for the more general Knapsack secretary problem, in which each element has a size and weight, and the goal is to find a maximum-weight set of elements whose total size is at most a given integer B.

Babaioff *et al.* [2] had earlier introduced the so-called *matroid secretary problem*, and gave an $O(\log r)$-competitive algorithm to find the max-weight independent set of elements, where r is the rank of the underlying matroid. A 16-competitive algorithm was also given in [2] for the special case of graphic matroids; this was based on their $4d$-competitive algorithm algorithm for the important case of *transversal matroids*, where d is the maximum degree of any left-vertex. Recently, Dimitrov and Plaxton [5] improved the latter to a ratio of 16 for all transversal matroids. A significant open question is whether there exists a $O(1)$-competitive algorithm for general matroids, or for other secretary problems with non-matroid constraints.

The secretary problem and variants can be understood as online allocation questions: In the basic problem, elements correspond to agents with different valuations for a single good; the goal of the algorithm is to sell the good to the agent who values it most (thus maximizing social welfare), though agents arrive one at a time in a random order. In more complex problems such as those considered in this paper, there may be multiple goods to auction, agents may have varying valuations for different bundles of goods, etc. These lead to questions on online *mechanism design*, where the algorithm may be viewed as an auctioneer selling goods to agents arriving online; the auctioneer may wish to maximize revenue, social welfare, etc. Hajiaghayi *et al.* [8] consider online mechanism design questions and give strategyproof mechanisms for various auction problems.

These online allocation problems arise in many practical situations where decisions must be made in real-time without knowledge of the future, or with very limited knowledge. For example, a factory needs to decide which orders to fulfil, without knowing whether more valuable orders will be placed later. Buyers and sellers of houses must decide whether to go through with a transaction, though they may receive a better offer in a week or a month. Below, we give an example from online advertising systems, which we use as a recurring motivation through the paper.

Internet-based systems are now being used to sell advertising space in other media, such as newspapers, radio and television broadcasts, etc. Advertisers in these media typically plan advertising campaigns and reserve slots well in advance to coincide with product launches, peak shopping seasons, or other events. In such situations, it is unreasonable to run an auction immediately before the event to determine which ads are shown, as is done for sponsored search and other online advertising.

Consider an automatic advertising reservation system, in which the seller controls a number of *slots*, each representing a position in which an advertisement (hereafter *ad*) can be published. Advertisers/Bidders appear periodically, and report which slots they would like to place an ad in, and how much they are willing to pay for each slot. When an advertiser reports a bid, the system must immediately decide whether or not to accept it; if a bid is accepted, the ad *must* be placed in the corresponding slot, and if not, the ad is permanently rejected. Note that in disallowing the removal of an accepted ad, our model differs significantly from that of [4], in which the seller can subsequently remove an accepted ad if he makes a compensatory payment to the advertiser.

We model this system as as an online edge-weighted matching problem on a bipartite graph $G(L \cup R, E)$: the vertices of set R correspond to the set of slots, and those of set L to the ads. For each vertex $l \in L$, its neighbors in R correspond to the slots in which ad l can appear, and the weight of edge (l, r)

is the amount the advertiser is willing to pay if l appears in slot r. Initially, the seller knows the set of slots R; vertices of L appear sequentially in a random order, as advertisers bid on slots. When a vertex $l \in L$ is seen, all the edges from l to R are revealed, together with their weights; the seller must immediately decide whether to accept ad l, and if so, which of the relevant slots to place it in. The seller's goal, obviously, is to maximize his revenue. Subsequently, we refer to this problem as Bipartite Vertex-at-a-time Matching (BVM). We describe our results for BVM and other problems below.

We remind the reader that though our results are described as algorithms and we analyze competitive ratios for them, they can be viewed as mechanisms for natural online allocation problems. All the algorithms are monotonic in bids/values (that is, if an element is selected by the algorithm, it would also be selected if its value were raised further). Thus, for those problems where bidders are single-minded, we can obtain dominant-strategy truthful mechanisms.

1.1 Results and Outline

In Section 2, we obtain an 8-competitive algorithm for Bipartite Vertex-at-a-time Matching (BVM), corresponding to the basic online ad allocation problem. We give a simpler and tighter analysis for an algorithm essentially due to Dimitrov and Plaxton [5] for the special case of transversal matroids; this allows us to improve the competitive ratio from 16 to 8, even for the more general BVM problem. Recall that the elements of a transversal matroid are one partite set L (subsequently referred to as the *left vertices*) of a bipartite graph, and a set of vertices $S \subseteq L$ is independent if the graph contains a perfect matching from S to the other partite set. That is, the transversal matroid secretary problem is equivalent to the special case of BVM in which all edges incident to each $l \in L$ have the same weight. (Equivalently, the weights are on vertices of L instead of edges.) Independently of this paper, Babaioff *et al.* [1] recently gave a 4-competitive algorithm for the *weighted k-secretary* problem, which is the following restricted case of BVM: Each ad/element l has a weight $w(l)$, each slot r has a multiplier $v(r)$, and *any* ad can be assigned to *any* slot; the value of assigning l to r is $w(l) \cdot v(r)$.

Besides providing improved competitive ratios, our methods are of interest as they can be naturally applied to more general problems and appear robust to changes in the model. We illustrate this in Section 3 by extending the algorithms and intuition developed for BVM to *hypergraph* problems, with applications to more complex advertising systems in which advertisers desire *bundles* of slots, as opposed to a single slot. In particular, we obtain constant-competitive algorithms for finding independent edge sets in hypergraphs of constant edge-size.

We also introduce secretary problems with *groups*, to model applications in which we do not see a truly random permutation of elements. We assume that an adversary can group the elements arbitrarily, but once the groups are constructed, they appear in random order. When a group appears, the algorithm can see all the elements in the group. We discuss this idea further in Section 4.

Finally, in Section 5, we obtain a simple $2e$-competitive algorithm for the problem of finding independent edge-sets in graphic matroids, improving the ratio of 16 from [2]. Recently, and independently from our work, the authors of [1] gave a $3e$-competitive algorithm for this problem.

The majority of our algorithms follow the "sample-and-price" method common to many solutions to secretary problems. That is, we look at a random sample of elements containing a constant fraction of the input, and use the values observed to determine *prices* or thresholds. In the second half, we accept an element if its weight/value is above the given price. For instance, in the optimal solution to the original secretary problem, the price is set to be the highest value seen in the first $1/e$ fraction of the input, and we accept any element from the remaining set with value greater than this price. Throughout this paper, we assume that the weights of all elements are distinct; one can always ensure this through a tie-breaking scheme.

Several proofs have been omitted from this extended abstract; a longer version can be found at http://arxiv.org/abs/0902.2795 or on the authors' websites.

2 The Bipartite Vertex-at-a-Time Matching Problem

Recall that in the BVM problem, the algorithm is initially given one partite set R of a bipartite graph $G(L \cup R, E)$, together with the size of the other partite set L. The algorithm sees the vertices of L sequentially, in a random order. When a vertex $l \in L$ is seen, all edges incident to l are revealed, together with their weights. The algorithm must immediately either match l to an available vertex of R, or decide that l will remain permanently unmatched. In this section, we show that an algorithm based on that of [5] gives a competitive ratio of 8 for this problem. Before presenting the algorithm for BVM, we describe a closely related algorithm Simulate that is easier to analyze, and then show that our final algorithm does at least as well as Simulate.

Let Greedy denote the greedy algorithm below for the offline Edge-weighted bipartite matching problem. Let $w(F)$ denote the weight of a set of edges F, and OPT denote the weight of an optimum (max-weight) matching on G. It is easy to see the following proposition, that Greedy is a 2-approximation.

Proposition 1. $w(M) \geq \mathrm{OPT}/2$.

```
Greedy($G(L \cup R, E)$):
Sort edges of $E$ in decreasing order of weight.
Matching $M \leftarrow \emptyset$
For each edge $e \in E$, in sorted order
  If $M \cup e$ is a matching:
      $M \leftarrow M \cup e$
Return $M$.
```

We now describe the algorithm Simulate, which we use purely to analyze our final algorithm for BVM.

Say that an edge e is *considered* by Simulate if we flip a coin and assign e to either M_1 or M_2. We make two observations about Simulate: Once any edge incident to a vertex $l \in L$ has been considered, no other edge incident to l will be considered later. Second, once an edge incident to $r \in R$ has been added to M_1, no subsequent edge incident to r will be considered. (Note that multiple edges incident to r might be considered until one of these edges is added to M_1.)

```
SIMULATE:
Sort edges of $G(L \cup R, E)$ in decreasing order of weight.
$M_1, M_2 \leftarrow \emptyset$
Mark each vertex $l \in L$ as unassigned.
For each edge $e = (l, r) \in E$, in sorted order
   If $l$ is unassigned AND $M_1 \cup e$ is a matching:
         Mark $l$ as assigned
         Flip a coin with probability $p$ of heads
         If heads, $M_1 \leftarrow M_1 \cup e$
         Else $M_2 \leftarrow M_2 \cup e$
$M_3 \leftarrow M_2$
For each vertex $r \in R$
   If $r$ has degree $> 1$ in $M_3$
         Delete all edges incident to $r$ from $M_3$.
```

Observe that from our description of SIMULATE, M_1 is a matching, but M_2 may not be, as a vertex $r \in R$ may be incident to multiple edges of M_2. Hence, we have a final pruning step in case there are multiple edges incident to the same vertex of R; this gives us a matching M_3. We prove two statements about SIMULATE, and later show that the matching returned by our online algorithm is at least as good as M_3.

Lemma 1. $\mathbb{E}[w(M_1)] \geq p\mathrm{OPT}/2$ and $\mathbb{E}[w(M_2)] \geq (1-p)\mathrm{OPT}/2$.

Lemma 2. $\mathbb{E}[w(M_3)] \geq \frac{p^2(1-p)}{2}\mathrm{OPT}$.

Before describing our final algorithm for BVM, we show that the matching returned by an intermediate algorithm SAMPLEANDPERMUTE (below) is at least as good as M_3, which implies that we have a $\frac{2}{(1-p)p^2}$-competitive algorithm: setting $p = 2/3$, we get a 13.5-competitive algorithm. However, our pruning step allows us to take an edge for M_3 only if its right endpoint has degree 1; a more careful pruning step allows more edges in the matching. We use this fact to give a tighter analysis for the next algorithm, obtaining a competitive ratio of 8.

Note that the matching M_1 in SAMPLEANDPERMUTE is precisely the same as M_1 from SIMULATE; intuitively, in the former, we toss all the coins at once and run GREEDY, while in the latter, we toss coins while constructing the Greedy Matching. (More precisely, the two algorithms to generate the matchings are equivalent.) Similarly, the "matching" M_2 in this algorithm is essentially M_2 from SIMULATE. The difference between the two algorithms is in the pruning step: To construct M_3 in SIMULATE, we delete all edges incident to any vertex $r \in R$ with degree greater than 1; in SAMPLEANDPERMUTE, we add to M the first such edge seen in our permutation of $L - L'$. It follows immediately from Lemma 2 that $\mathbb{E}[w(M)] \geq p^2(1-p)\mathrm{OPT}/2$, but accounting for the difference in pruning allows the following tighter statement.

Lemma 3. $\mathbb{E}[w(M)] \geq \frac{p(1-p)}{2}\mathrm{OPT}$.

```
SampleAndPermute($G(L \cup R, E)$):
$L' \leftarrow \emptyset$
For each $l \in L$:
  With probability $p$, $L' \leftarrow L' \cup \{l\}$
$M_1 \leftarrow$ Greedy($G[L' \cup R]$).
For each $r \in R$:
  Set $price(r)$ to be the weight of the edge incident to $r$ in $M_1$.
$M, M_2 \leftarrow \emptyset$
For each $l \in L - L'$, in random order:
  Let $e = (l, r)$ be the highest-weight edge such that $w(e) \geq price(r)$
  Add $e$ to $M_2$.
  If $M \cup e$ is a matching, add $e$ to $M$.
```

We now present our final algorithm, a trivial modification of SampleAndPermute for the online BVM problem.

```
SampleAndPrice($|L|, R$)
$k \leftarrow Binom(|L|, p)$
Let $L'$ be the first $k$ vertices of $L$.
$M_1 \leftarrow$ Greedy($G[L' \cup R]$).
For each $r \in R$:
  Set $price(r)$ to be the weight of the edge incident to $r$ in $M_1$.
$M \leftarrow \emptyset$
For each subsequent $l \in L - L'$, :
  Let $e = (l, r)$ be the highest-weight edge such that $w(e) \geq price(r)$
  If $M \cup e$ is a matching, accept $e$ for $M$.
```

As the input to SampleAndPrice is a random permutation, L' is a subset of L in which each vertex of L is selected with probability p; it is easy to see that this algorithm is equivalent to SampleAndPermute. Therefore, $\mathbb{E}[w(M)] \geq \frac{p(1-p)}{2}\mathrm{OPT}$; setting $p = 1/2$ implies that the expected competitive ratio is 8.

3 Independent Edge Sets in Hypergraphs

We now consider more complex online allocation problems, where agents are interested in demand *bundles*. For instance, in ad reservation systems, advertisers rarely make reservations for a single ad at a time; they are more likely to plan advertising campaigns involving multiple individual ads. In many campaigns, advertisers create various ads which are related to and complement or reinforce each other; these advertisers might be interested in acquiring a bundle or set of slots for this campaign. They submit to the reservation system the bundles they are interested in, together with the price they are willing to pay; the system must either accept a request for an entire bundle or reject it, as it does not receive revenue for providing the advertiser with a part of the bundle.

These problems with bundles can be modeled by matchings in *hypergraphs*. We define two natural hypergraph problems below, and show how they capture the ad reservation problems:

In the Hypergraph Edge-at-a-time Matching (HEM) problem, we are initally given the vertex set of a hypergraph; subsequently, hyperedges appear in a random order. When an edge (together with its weight) is revealed, the algorithm must immediately decide whether or not to accept it; as before, the goal is to select a maximum-weight set of disjoint edges. For arbitrary hypergraphs, even the offline version of this problem is NP-Complete (and also hard to approximate) via an easy reduction from the Independent Set problem. However, the difficulty is related to the size of the hyperedges; if all edges contain only 2 vertices, for instance, then we are simply trying to find a matching in a (possibly non-bipartite) graph. (Even in this special case, the problem is of interest in an online setting.) Let d denote the maximum size of an edge in the hypergraph.

We provide an $O(d^2)$-competitive algorithm for the HEM problem by solving the more general Hypergraph Vertex-at-a-time Matching (HVM) problem, described as follows: We are initally given a subset R of the vertex set of a hypergraph. The remaining vertices L arrive online; each edge of the hypergraph is constrained to contain exactly one vertex of L, together with some vertices of R. The vertices of L appear online in a random order; when $l \in L$ is revealed, the algorithm also sees all edges incident to l, together with their weight. At this point, the algorithm must immediately decide whether or not to accept some edge containing l, and if so, which edge; again, the goal is for the algorithm to select a maximum-weight set of disjoint edges. Here, let d denote the maximum number of vertices of R contained in a single edge (so the largest edge has $d+1$ vertices). First, we note that the HEM problem with edge size d reduces to the HVM problem with edge size $d+1$: Let R be the vertex set of the original hypergraph, and add one vertex to L for each original edge. An edge of the new hypergraph consists of an old edge, together with the corresponding vertex of L. Observing a random permutation of L together with the incident edges is equivalent to a random permutation of the edge set of the original hypergraph. Also, the BVM problem of Section 2 is the special case of HVM when $d = 1$.

It is easy to see that if each advertiser submits a request for a single bundle, we obtain the HEM problem with vertex set corresponding to the set of slots, and the requested bundles forming the hyperedges. More generally, an advertiser may submit a request for *one of* a set of bundles, together with a price for each bundle. (For example, a grocery store might want their coupons to appear in any three out of four local newspapers. A large advertiser may be willing to sponsor one "prime time" television show and have several ads shown during the show.) This leads to the HVM problem, with vertex set L corresponding to the set of advertisers, and set R to the set of slots: We receive a random permutation of advertisers, and each advertiser informs us of the bundles she is interested in, together with a price for each bundle.

Let Greedy denote the offline algorithm for HVM that sorts edges in decreasing order of weight, and selects an edge if it is disjoint from all previously selected edges. For ease of exposition, we subsequently assume that the hypergraph is $(d+1)$-uniform; that is, that each edge contains exactly d vertices of R together with one vertex of L.

Proposition 2. Greedy *returns a* $(d+1)$*-approximation to the maximum-weight disjoint edge set.*

We again define an algorithm Simulate, as in Section 2:

```
Simulate:
Sort edges of $E$ in decreasing order of weight.
Mark each vertex $l \in L$ as unassigned.
$M_1, M_2 \leftarrow \emptyset$
For each edge $e \in E$ in sorted order:
  Let $l$ be the vertex of $L$ in $e$
  If $l$ is unassigned AND $e$ is disjoint from $M_1$:
      Mark $l$ as assigned.
      Flip a coin with probability $p$ of heads
      If heads, add $e$ to $M_1$
      If tails, add $e$ to $M_2$
$M_3 \leftarrow \emptyset$
For each $e \in M_2$:
  Add $e$ to $M_3$ if $e$ is disjoint from the rest of $M_2$.
```

As before, we let $w(F)$ denote the weight of an edge set F. The proofs of the following lemma is exactly analogous to Lemma 1.

Lemma 4. $\mathbb{E}[w(M_1)] \geq p \cdot \mathrm{OPT}/(d+1)$ and $\mathbb{E}[w(M_2)] \geq (1-p)\mathrm{OPT}/(d+1)$.
It is now slightly more complex to bound the weight of M_3 than it was for the BVM problem; for BVM, the set of edges in M_2 incident to $v \in R$ interfere only with each other, but in the hypergraph version, edges e_1 and e_2 might not intersect each other, though they may both intersect e_3, and hence all of e_1, e_2, e_3 will have to be deleted. However, we can use a similar intuition: In BVM, we charge all edges of M_2 incident to v to the heaviest such edge; in expectation, each edge is charged a constant number of times. For the HVM problem, we charge all the edges in a "connected component" to the heaviest edge in the component, and argue that (with a suitable choice of p) the average size of the components is small. More formally, we prove the following lemma:

Lemma 5. *Setting* $p = 1 - 1/2d$, $\mathbb{E}[w(M_3)] \geq \frac{\mathrm{OPT}}{12d(d+1)}$.

```
SampleAndPrice($|L|, R$)
$k \leftarrow Binom(|L|, 1 - \frac{1}{2d})$
Let $L'$ be the first $k$ elements of $L$.
$M_1 \leftarrow$ Greedy($G(L', R)$).
For each $v \in V$:
  Set $price(v)$ to be the weight of the edge incident to $v$ in $M_1$.
$M \leftarrow \emptyset$
For each subsequent $l \in L - L'$:
  Let $e$ be the highest-weight edge containing $l$ such that for each $v \in e$, $w(e) \geq price(v)$
  If $e$ is disjoint from $M$, add $e$ to $M$.
```

Our final algorithm SampleAndPrice for the HVM problem is defined above. As before, since the input is a random permutation of L, L' is a subset of L in which every vertex is selected independently with probability $1 - 1/2d$, and the matching M is at least as good as M_3 from Simulate. Therefore, we have proved the following theorem:

Theorem 1. SampleAndPrice *is an $O(d^2)$-competitive algorithm for the HVM secretary problem.*

Note that M may also contain extra edges that occur earlier in the permutation than edges they intersect; for the BVM problem, this was the difference between Lemma 2 and the stronger bound Lemma 3. We do not provide a tighter analysis similar to Lemma 3 for the HVM problem in this extended abstract, nor do we optimize the constants of Lemma 5. In particular, for the HEM problem with $d = 2$ (finding an online matching in a non-bipartite graph $G(V, E)$, given a random permutatation of E), a smaller constant can easily be obtained.

4 Secretary Problems with Groups

For some online allocation problems, the assumption of a truly random ordering of all elements may not be realistic; element positions may be correlated. For an advertising example, suppose – as we did for BVM – that individual ad requests are made to a reservation system; each request comes with a set of acceptable slots, and values for showing the ad in each of those slots. A single merchant, when planning a campaign, may submit to the reservation system multiple ads, together with the slots in which each ad can be placed, and a price for each ad-slot combination. Even if the merchants arrive in a random order, this does not correspond to a random permutation of ads. Similarly, an external event may cause several companies from the same industry to simultaneously request ad slots. In these situations, our analysis of Section 2 is not directly applicable.

We investigate secretary-type problems in which, instead of elements arriving in random order, they can be grouped by an adversary. The algorithm receives the number of groups in advance, instead of the number of elements. However, once the groups have been constructed, they arrive in random order; when a group arrives, the algorithm can see all its elements at once. Note that the groups are fixed in advance; the adversary cannot construct groups in response to the algorithm's choices or the set of groups seen so far. The effect of such grouping on the difficulty of the problem is not immediately clear: The adversary can ensure that some permutations of the element set never occur, which might make the problem more difficult. On the other hand, as the algorithm is allowed to see several elements at once, it may be easier to compute a good solution.

Theorem 2. *If $(\mathcal{X}, \mathcal{I})$ is a p-extendible system and r is the size of a largest independent set, there is an $O(p \log r)$-competitive algorithm for the $(\mathcal{X}, \mathcal{I})$ secretary problem with groups.*

Matroids are examples of 1-extendible systems; informally, in a p-extendible system, if a single element e is added to an independent set B, at most p other elements from B need to be discarded to maintain independence [9,11]. The set of matchings in a hypergraph with edge size d form a d-extendible system, and so there is an $O(d \log n)$-competitive algorithm for the HVM problem with groups. It follows that there is an $O(\log |L|)$-competitive algorithm for the BVM problem with groups. The proof of Theorem 2 is a straightforward generalization of Theorem 3.2 in [2], which shows that there is an $O(\log r)$-competitive algorithm for the basic secretary problem (without groups) in matroids of rank r.

Theorem 2 shows that the best known algorithm for the matroid secretary problem with groups matches that for the problem without groups; in each case, there is an $O(\log r)$-competitive algorithm. Most classes of matroids for which a constant-factor approximation is known for the secretary problem satisfy the so-called *α-partition* property of [1]. For all such matroids, an $e\alpha$-competitive algorithm is known for the secretary problem, and this extends directly to the version with groups. On the other hand, we have seen an 8-competitive algorithm for BVM, generalizing the transversal matroid secretary problem, though transversal matroids do *not* satisfy the α-partition property for any constant α. Thus, a natural question is whether one can obtain a constant-competitive algorithm for the transversal matroid secretary problem with groups or BVM with groups, corresponding to the ad reservation problem where multiple ads (such as those from a single advertiser) may arrive simultaneously.

We note that the natural Sample-and-Price algorithm does not work, but make the following conjecture, for which we provide some evidence in the full version:

Conjecture 1. There is an $O(1)$-competitive for the BVM problem with groups.

5 Graphic Matroids

In this section, we describe a $2e$-competitive algorithm for the Graphic Matroid Secretary problem. Here, we are initially given the set of vertices V of an undirected edge-weighted graph $G = (V, E)$, and the size of its edge set $|E|$. Edges of the graph appear in a random order; the goal is to accept a maximum-weight subset of edges F that does not contain any cycles. As always, the decision to accept an edge must be made upon its arrival, and cannot be revoked.

This problem is equivalent to finding the maximum-weight spanning tree (if G is connected) and is also equivalent to finding the maximum-weight independent set in the graphic matroid defined by the graph G. Babaioff et al. [2] give a 16-competitive algorithm for the secretary version of this problem based on a related algorithm for transversal matroids. We give a simple reduction to the classical secretary problem, losing a factor of 2 in the reduction. We thus obtain a $2e \approx 5.436$-competitive algorithm for the Graphic Matroid Secretary problem.[1]

[1] Independently, a $3e$-competitive algorithm was recently given by [1].

Fix an ordering $v_1, v_2, \ldots, v_n$ on the vertices of G. Consider two directed graphs: graph G_0 is obtained by orienting every edge of G from higher numbered to lower numbered vertex, and graph G_1 by orienting every edge from lower to higher numbered vertex.

Our online algorithm initially flips a fair coin $X \in \{0, 1\}$. For each vertex v independently, it runs a secretary algorithm to find the maximum-weight edge leaving v in G_X. The output of the algorithm is F', the union of all edges accepted by the individual secretary algorithms. Since the graph G_X is acyclic and each vertex has at most one outgoing edge in F', the set of edges returned must be acyclic even in the undirected sense.

Theorem 3. *The algorithm above is* $2e$*-competitive for the graphic matroid secretary problem.*

6 Open Problems

1. An improved understanding of groups – and their contribution to the difficulty of secretary-type problems – is likely to be of interest. In particular, can one find an $O(1)$-competitive algorithm for the BVM problem with groups?
2. Few lower bounds for these problems are known beyond $1/e$ for the original secretary problem; obtaining such bounds may require new techniques.
3. Obtain an $O(1)$-competitive algorithm for the general matroid secretary problem.

Acknowledgments. We would like to thank Florin Constantin, Jon Feldman, and S. Muthukrishnan for helpful discussions on BVM and related problems. We thank Chandra Chekuri for discussions and pointers to [9,11].

References

1. Babaioff, M., Dinitz, M., Gupta, A., Immorlica, N., Talwar, K.: Secretary Problems: Weights and Discounts. In: Proc. of ACM-SIAM SODA, pp. 1245–1254 (2009)
2. Babaioff, M., Immorlica, N., Kleinberg, R.: Matroids, Secretary Problems, and Online Mechanisms. In: Proc. of ACM-SIAM SODA, pp. 434–443 (2007)
3. Babaioff, M., Immorlica, N., Kempe, D., Kleinberg, R.: A Knapsack Secretary Problem with Applications. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) APPROX 2007. LNCS, vol. 4627, pp. 16–28. Springer, Heidelberg (2007)
4. Constantin, F., Feldman, J., Muthukrishnan, S., Pál, M.: Online Ad Slotting with Cancellations. In: Proc. of ACM-SIAM SODA, pp. 1265–1274 (2009)
5. Dimitrov, N.B., Plaxton, C.G.: Competitive Weighted Matching in Transversal Matroids. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 397–408. Springer, Heidelberg (2008)

6. Ferguson, T.S.: Who Solved the Secretary Problem? J. Stat. Sci. 4, 282–289 (1989)
7. Gardner, M.: Mathematical Games column. Scientific Amer. 35 (February/March 1960)
8. Hajiaghayi, M.T., Kleinberg, R., Parkes, D.C.: Adaptive limited-supply online auctions. In: Proc. 5th ACM Conf. on Electronic commerce, pp. 71–80 (2004)
9. Jenkyns, T.A.: The efficiency of the "greedy" algorithm. In: Proc. of 7th South Eastern Conference on Combinatorics, Graph Theory and Computing, pp. 341–350 (1976)
10. Kleinberg, R.: A multiple-choice secretary problem with applications to online auctions. In: Proc. of ACM-SIAM SODA, pp. 630–631 (2005)
11. Mestre, J.: Greedy in Approximation Algorithms. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 528–539. Springer, Heidelberg (2006)

Leader Election in Ad Hoc Radio Networks: A Keen Ear Helps

Dariusz R. Kowalski[1],* and Andrzej Pelc[2],**

[1] Department of Computer Science, The University of Liverpool,
Liverpool L69 3BX, United Kingdom
darek@liv.ac.uk

[2] Département d'informatique, Université du Québec en Outaouais,
Gatineau, Québec J8X 3X7, Canada
pelc@uqo.ca

Abstract. We address the fundamental distributed problem of leader election in ad hoc radio networks modeled as undirected graphs. Nodes are stations having distinct integer labels, and each node knows only its own label and a polynomial upper bound on all labels. A signal from a transmitting node reaches all neighbors. What distinguishes radio networks from message-passing networks is that a message is received successfully by a node, if and only if, exactly one of its neighbors transmits in this round. If two neighbors of a node transmit simultaneously in a given round, none of the messages is heard by the receiving node. In this case we say that a *collision* occurred at this node.

An important capability of nodes of a radio network is *collision detection*: the ability of nodes to distinguish a collision from the background noise occurring when no neighbor transmits. (This ability is the "keen ear" of the nodes.) Can collision detection speed up leader election in arbitrary radio networks? We give a positive answer to this question. More precisely, our main result is a deterministic leader election algorithm working in time $O(n)$ in all n-node networks, if collision detection is available, while it is known that deterministic leader election requires time $\Omega(n \log n)$, even for complete networks, if there is no collision detection. This is the first computational task whose execution for arbitrary radio networks is shown to be faster with collision detection than without it.

1 Introduction

The background and the problem. A radio network is modeled as an n-node undirected graph whose nodes are stations having distinct labels. Labels are integers from an interval $\{1,\ldots,N\}$, where $N = O(n^\gamma)$, for a constant $\gamma > 1$. We consider *ad hoc* networks in which each node knows only its own label and an upper bound N on all labels, but does not know the topology of the network or its size. Nodes do not even know their immediate neighborhood or their degree. Communication proceeds in synchronous rounds. In each round each node acts either as a transmitter or as a receiver.

* This work was supported by the Engineering and Physical Sciences Research Council [grant number EP/G023018/1].

** Partially supported by NSERC discovery grant and by the Research Chair in Distributed Computing at the Université du Québec en Outaouais.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 521–533, 2009.

A signal from a transmitting node reaches all neighbors. A message is *heard* (received successfully) by a node, if and only if it acts as a receiver and exactly one of its neighbors transmits in this round. If two neighbors of a node u transmit simultaneously in a given round, none of the messages is heard by this node and we say that a *collision* occurred at u.

An important capability of nodes of a radio network is *collision detection*: the ability of nodes to distinguish a collision from "silence", which is in fact the background noise occurring when no neighbor transmits. (This ability is the "keen ear" of nodes: a collision slightly increases the level of noise always existing in the channel and detecting this difference requires a more sensitive receiving device.) Algorithmic aspects of radio communication have been studied both assuming collision detection and without supposing this capability. Some algorithmic techniques crucially depend on collision detection: even in the case when no message can be heard due to collisions, nodes can engage in "conversations" and transmit important control information using noise and silence as bits. This information can help to prepare future successful message transmissions. On the other hand, it has been shown that in some cases collision detection can be simulated in networks whose nodes do not have this capability; see, e.g., the procedure Echo designed in [18].

Hence it is natural to ask how crucial is collision detection for the efficiency of performing tasks in radio networks. More precisely, can the availability of collision detection speed up the execution of a computational task in ad hoc radio networks? It has been long known that in the special case of *single-hop* radio networks (also known as a *multiple-access channel*), i.e., those with the topology of a complete graph, the answer to this question is positive. For complete graphs, deterministic leader election can be done in time $\Theta(\log n)$ with collision detection [3,14,24], while it requires time $\Omega(n \log n)$ in the model without this capability, by an adaptation of techniques from [7]. Leader election is a task involving symmetry breaking: initially all nodes have the same status *non-leader* and the goal is for all nodes but one to keep this status and for the remaining single node to get the status *leader*. All nodes must learn the leader's identity.

However, the more general question if any task in *arbitrary* radio networks can be performed faster with collision detection than without it, remained open. In other words, does there exist an important computational task such that every algorithm solving it for all networks without collision detection is slower than some algorithm solving it for all networks with collision detection? This paper gives a positive answer to this question.

Our results. We show that deterministic leader election in arbitrary ad hoc radio networks is faster with collision detection than without it. More precisely, we show a deterministic leader election algorithm working in time $O(n)$ in all n-node networks, if collision detection is available. This complexity is optimal. On the other hand, a simple adaptation of techniques from [7] shows that deterministic leader election requires time $\Omega(n \log n)$ even for complete networks, if collision detection is not available in the system. Our linear time algorithm is based on several novel techniques introduced to handle communication in radio networks with collision detection: remote token elimination, fuzzy-separating families, distributed fuzzy-degree clustering.

Related work. Algorithmic aspects of radio communication in networks of arbitrary topology have been intensely studied in the last two decades, starting with the seminal paper [4]. A lot of attention has been devoted to efficient algorithms for such tasks as broadcasting, in particular in ad hoc radio networks, both in the deterministic [7,8,9] and in the randomized [1,8,19,20] setting. The above papers do not assume collision detection. The model with collision detection has been less studied in the context of radio broadcasting: cf. [5] for arbitrary networks and [10] for geometric networks.

Leader election, which is the task considered in the present paper, is a classic topic in distributed computing, and has been widely studied in the early history of this domain (cf. [21]). Most of the results on leader election in the radio model concern *single-hop* networks of known size n. Some of these results were originally obtained for other distributed problems but have corollaries for leader election. For the time of deterministic leader election without collision detection, matching bounds $\Omega(n \log n)$ and $O(n \log n)$ follow from [7], with the upper bound being non-constructive. A constructive upper bound $O(n\ \text{polylog}(n))$ follows from [15]. For the time of deterministic algorithms with collision detection, matching bounds are also known: $\Omega(\log n)$ follows from [13], and $O(\log n)$ follows from [3,14,24]. For the expected time of randomized algorithms without collision detection, the same matching bounds are known: $\Omega(\log n)$ follows from [20] and $O(\log n)$ from [2]. Finally, randomized leader election with collision detection can be done faster: matching bounds $\Omega(\log \log n)$ (for fair protocols) and $O(\log \log n)$ were proved in [25]. For more references and a detailed study of classes of randomized protocols and energy issues for leader election in single-hop networks, see [16,22].

For leader election in *arbitrary networks* results are much less complete. The best bounds on the time of deterministic algorithms without collision detection are $\Omega(n \log n)$ (by applying techniques from [7]) and $O(n \log^3 n)$ [6], respectively. To the best of our knowledge, no results are published for deterministic leader election with collision detection, but the lower bound $\Omega(n)$ and the upper bound $O(n \log n)$ are folklore (see, respectively, Proposition 1 and the procedure $\texttt{L\_Elect}(X,k)$ taken in case $k = n+1$).

2 Preliminaries

Throughout the paper, $[m]$ denotes the set $\{1,\ldots,m\}$, for any positive integer m. A linear upper bound on the number of nodes in the graph is denoted by n, and all labels of nodes are integers from the interval $[N]$, where $N = O(n^{\gamma})$, for a constant $\gamma > 1$. Without loss of generality we assume that N is a power of 2. All logarithms are to the base of 2. We denote $\ell = \log N + 1$ and assume that labels of nodes are binary strings of length ℓ. The notation polylog(m) stands for $O(\log^a m)$, for some constant $a > 1$.

The following folklore result gives a lower bound on the time of deterministic leader election. It holds even for less restrictive models than that of radio communication, e.g., for the message passing model and even if each station obtains parameters n and N as a part of the input.

Proposition 1. *Every deterministic algorithm solving the leader election problem in arbitrary networks requires time $\Omega(n)$ on some n node networks.*

It turns out that in networks without collision detection a stronger lower bound can be proved. An application of combinatorial tools developed in [7] to the leader election problem gives the following lower bound.

Proposition 2. *Every deterministic algorithm solving the leader election problem in arbitrary n-node radio networks* without collision detection *requires time* $\Omega(n \log n)$ *on some network.*

We say that a (combinatorial) data structure is *explicitly constructed* if there is an algorithm computing this structure in time polynomial in the size of the structure. We will use the following combinatorial notions.

Let $k \leq m$. A family $\mathcal{F}$ of subsets of $[m]$ is (m,k)*-strongly selective*, if for every non-empty subset Z of $[m]$ of size at most k, and for every element $z \in Z$, there exists a set $F \in \mathcal{F}$ such that $Z \cap F = \{z\}$. Notice that strongly selective families are equivalent to the well studied superimposed codes (cf. [17]) and have applications in many domains of computer science, ranging from pattern matching to circuit complexity.

Proposition 3 ([11]). *For any integers* $k \leq m$ *there exists an explicitly constructed* (m,k)*-strongly selective family of size* $O(k^2 \log m)$.

Strongly-selective families can be used by nodes to learn their neighborhood in a distributed way. This process, however, is not very fast if neighborhoods are large, due to a lower bound $\Omega(\min\{m,k^2\} \log_k m)$ on the size of such families [7]. In our algorithms only nodes with relatively small degrees learn their neighborhoods using strongly-selective families of size $O(n)$, however this requires a distributed procedure of checking in time $O(n)$ whether a node has a desirably small, or rather a large degree. In order to make such a local classification of nodes, we introduce a new combinatorial structure called a fuzzy-separating family. It will be an important tool in our distributed clustering method. A family $\mathcal{F}$ of subsets of $[m]$ is (m,a,b,c)*-fuzzy-separating*, for integers $a < b$ and c, if it has the following properties:

- for every non-empty subset Z of $[m]$ of size at most a, there are more than c sets disjoint from Z in $\mathcal{F}$;
- for every non-empty subset Z of $[m]$ of size at least b, there are at most c sets disjoint from Z in $\mathcal{F}$.

The name *fuzzy separating* comes from the fact that we can distinguish subsets Z of size at most a from those of size at least b by looking at the number of sets in $\mathcal{F}$ disjoint from Z, but the situation of subsets of intermediate size (between a and b) is not determined.

Lemma 1. *For sufficiently large n and N, there exists an explicitly constructed* $(N, n^{1/4}, \frac{n^{1/2}}{\log N}, c)$*-fuzzy-separating family of size* $O(n^{1/2} \text{ polylog}(N))$*, for some* $c = O(n^{1/2} \text{ polylog}(N))$.

We use the following terminology, introduced in [5]. A node v acting as a receiver in a given round *hears signal* μ, if at least one of its in-neighbors acts as a transmitter, i.e., if v hears a message or if there is a collision at v in this round. Otherwise (if no in-neighbor of v acts as a transmitter), v hears *silence*. A *contact message* is a fixed one-bit signal. A *k-neighborhood* of a node is the set of all nodes at distance at most k from it. A 1-neighborhood is simply called neighborhood.

Due to lack of space, the proofs of Proposition 2 and Lemma 1, as well as the correctness and complexity analysis of the algorithms, are omitted.

3 The Algorithm

We present a deterministic leader election algorithm working in time $O(n)$ on graphs with at most n nodes. We proceed in four steps. We first describe three procedures used in the description of the algorithm. Then we present an auxiliary leader election algorithm working under the additional assumptions that each node knows its neighborhood and knows the linear upper bound n on the number of nodes. (If the nodes know their neighborhood then leader election is much simpler then without this information.) Later we describe our main algorithm working without the first assumption (and using the auxiliary algorithm as an ingredient). Finally, we show how the second assumption (about knowledge of n) can be removed.

In the algorithm description we use the well-known concept of *multiplexing* of procedures. By multiplexing we mean that the execution of a procedure, described as a sequence of consecutive steps, will be interleaved with the execution of other procedures needed to complete the task. Disjoint *time threads* (which are sets of rounds) are reserved to participating procedures (e.g., odd and even rounds for two procedures, and in general, each step of x threads is run every x rounds). Multiplexing serves mainly to avoid collisions between transmissions from different procedures.

We will use two types of entities issued by nodes: *tokens* and *agents*. Tokens are labeled with the label of the issuing node. During the algorithm they are "shown" to other nodes. Each node keeps in its memory the maximum of its own label and of all labels of tokens it has seen to date; smaller labels are forgotten. At the end, all nodes elect as the leader the node with the label corresponding to this maximum (which is updated many times by nodes during the algorithm execution). The crucial difficulty of the algorithm is how to circulate these tokens. One way is well known in leader election algorithms: tokens follow a prescribed route from node to neighbor (a DFS route in our case). For this type of token circulation we use *agents*; messages that carry tokens and serve as vehicles following a prescribed DFS route in some part of the graph. Unfortunately, not in all parts such a DFS route is known (recall that, in our main algorithm, nodes do not know their neighborhood), and hence we also have to use *remote* transmissions of tokens, bit by bit, using collision detection. Choosing the right way of communicating labels in each part of the graph and combining these two ways of communication, avoiding the danger of unwanted interferences, is the main novelty of our algorithm and its main technical difficulty.

To overcome this difficulty we introduce several new algorithmic ideas and tools. The first is a fast classification of nodes according to their degree. This classification into nodes of small and of large degrees is "fuzzy" (nodes of intermediate degrees may be classified either way), but is sufficient for our purposes and can be done fast. To this end we use small fuzzy separating families constructed in the previous section. Then we partition the entire graph into clusters, in a distributed way, according to degrees of nodes and to the sizes of connected components spanned by nodes of small and of large degrees. This partitioning is crucial, as token circulation is done differently in each type of clusters. Next, we elect *representatives*, one in each cluster. Among those, in turn, we elect $O(\log N)$ *local leaders* which include the (yet unknown) node with the largest label. The final leader election is performed among these local leaders: their tokens are shown to all nodes using several ways of circulation, depending on the type of cluster. In order to keep the total time linear, it is crucial that the number of local leaders be small; indeed, the time of this last phase has a polylogarithmic overhead with respect to the number of contenders. The danger of unwanted interferences forces us to use different time threads for internal communication in each type of clusters and for inter-cluster communication.

3.1 Basic Procedures

In the description of the algorithm we will use the following procedures.

procedure `standard_DFS`
This procedure operates under the assumption that all nodes of the graph know their immediate neighborhood. The procedure is initiated at some node of the graph which issues an agent carrying a token. A time limit for the duration of the procedure is determined from the outset and implemented by a counter carried by the agent. The token is passed between neighbors in a DFS manner by this agent, always to the yet unvisited neighbor with lowest label. Token transfer is done in one round by sending the agent by the node currently hosting it: the agent is a message containing the token, the counter and labels of the sending and of the receiving node. The procedure is either carried out for the specified amount of time without visiting all nodes, and then the agent comes back to the initiating node, using the DFS path in the opposite direction, or the procedure is repeated cyclicly for the specified amount of time. In the first case we say that DFS was unsuccessful, in the second case that it was successful.

procedure `blind_DFS`
This procedure operates *without* the assumption that all nodes of the graph know their neighborhood. The only difference with respect to `standard_DFS` is the choice of the node to which the agent goes next. Since nodes do not know their neighbors, the yet unvisited neighbor with smallest label is discovered using collision detection, in a binary search fashion. This discovery takes $2\ell = O(\log N) = O(\log n)$ rounds and proceeds as follows. In the first step, the node hosting the agent asks all yet unvisited neighbors to transmit their labels. If it heard silence, the agent is sent back to its DFS parent. If it heard noise, it asks all yet unvisited neighbors with labels in the interval $[1, N/2]$ to transmit their labels. Then, depending on whether it heard noise, silence, or a message, it asks all yet unvisited neighbors with labels in the interval $[1, N/4]$ (resp. $[N/2+1, 3N/4]$) to transmit their labels, or selects the yet unvisited neighbor with smallest label and stops (if a message from some node is heard). After $2\ell = O(\log N) = O(\log n)$ rounds, the yet unvisited neighbor with smallest label is discovered and the agent is sent to it in the next round. In total, $2\ell + 1$ rounds are needed to go to the next node in the DFS route. As before, blind DFS can be successful or not.

procedure `L_Elect`(X,k)
This procedure uses two parameters: the set X of participating nodes in the graph and the depth k. The goal is to select at least one and at most $2n/k$ local leaders among nodes in X. Each local leader is the node from X which has the largest label among all nodes from X at distance at most k from it. The procedure proceeds in ℓ stages, each using k rounds. Every time a node calls procedure `L_Elect`(X,k), it must know two things: whether it belongs to set X or not, and what is the value of the parameter k. This condition will be satisfied in all applications of procedure `L_Elect` in our leader election algorithms.

At the beginning of the procedure all nodes in X have status `qualified`, and the goal is that at the end of the stage only local leaders keep this status. In the first round of stage i, for $1 \leq i \leq \ell$, each qualified node with the ith bit of its label equal to 1 transmits the contact message. In round $j > 1$ of stage $i \leq \ell$, where $1 < j \leq k$, every node of the graph that heard signal μ in the previous round transmits the contact message. At the end of stage i, the following update is done: if a qualified node has not transmitted in the first round of the stage and it heard signal μ during this stage, it becomes *non-qualified.* A qualified node becomes a local leader at the end of the last stage of the procedure.

3.2 The Auxiliary Algorithm

We first present the auxiliary algorithm `Known_Neighb_LE` that elects a leader under the additional assumption that each node knows labels of all its neighbors. It also gets parameters n and N as a part of the input. The algorithm consists of two parts. In the first part $O(\log n)$ local leaders are elected using procedure `L_Elect`. In the second part a unique final leader is elected among local leaders by passing tokens initiated by all local leaders in a DFS manner and eliminating tokens initiated at local leaders with smaller labels. The finally elected leader is the node with the largest label among all nodes. The novelty of our approach is in stopping the leader election process started in the first part, some time before finishing it (which results in selecting slightly more local leaders instead of one leader, but in a shorter time), and finishing the process in the second part, using a different approach based on remote token elimination (which is another new technique introduced in this paper).

Algorithm `Known_Neighb_LE`

Part 1. Election of local leaders
Call procedure `L_Elect`(X,k) for X equal to the set of all nodes in the graph and $k = \frac{2n}{\log N}$. The output is the nonempty set Y of at most $\log N$ local leaders: nodes in Y get the status *local leader* and all other nodes in the graph get the status *non local leader*.

Part 2. Remote token elimination
This part is organized in two time threads. In the first time thread each local leader runs standard DFS, stopping after the issued agent comes back to it having visited all nodes. The circulating agent issued by each local leader carries its token (the label of the issuing node) and the sequence of labels of all visited nodes.[1] Each action of passing the

[1] This is only a technical assumption and can be avoided by introducing a counter and confirmation bits at visited nodes; thus the size of agents (and all other messages in our algorithms) can be reduced to $O(\log N)$.

agent from node u currently hosting it to the next node v in the DFS route is carried out in blocks consisting of 3 rounds. In the first round of the block *u proposes* the agent to v, in the second round *v confirms* that it can receive it, in the third round *u sends* the agent to v (if it got confirmation). The second time thread is reserved for possible *conflict resolution* designed to eliminate all agents but one, if more than one agent is proposed to node v. This conflict resolution lasts 2ℓ rounds and is performed in the second time thread, while the execution in the first time thread is halted (no action is performed until resolving the conflict). Thus execution in the second time thread is carried out in consecutive blocks of 2ℓ rounds. Below we give the details of the execution in both time threads.

Part 2 lasts exactly $7n$ rounds. In the beginning of part 2 each node becomes non *locked* and starts executing thread 1 (it is idle in thread 2 until it becomes *locked*).
Time thread 1: It is executed only by non *locked* nodes. A node that becomes non *locked* waits until the beginning of the next block of thread 1 and then starts its run. Recall that one block in which an agent is passed to the next node on the DFS route, or its host becomes locked, consists of three rounds.
Round 1: If a node u holding an agent is not *locked* then it proposes the agent to the next node v on its DFS route. The message has format *propose*(u,v) and contains the agent. If the DFS route is finished (at the node issuing the agent) the node does nothing.
Round 2: A node that receives a message *propose*(u,v) in the first round sends back a confirmation of the format *confirm*(v,u), and if it hears a collision it sends the contact message (to "jam" the channel).
Round 3:

- If a node u that proposed the agent to v in the first round receives the confirmation from v in the second round, it "passes" the agent to v: the agent is erased at u and v is now the node hosting the agent; otherwise (a collision, the contact message, or silence is heard) it sends an *alert* (the contact message) and becomes *locked*;
- if a node v that sent the confirmation to node u in the second round receives an alert or hears a collision in the third round, it does not accept the proposed agent (the agent is not "passed" from u to v); otherwise (if it hears silence) it accepts the agent (the agent is "passed" from u to v;
- if the token carried by the agent accepted by v is smaller than the largest token ever received by node v or smaller than its own label then the newly accepted agent is erased at v.

Time thread 2: The execution in this time thread is performed only by *locked* nodes. A locked node (hosting an agent) initializes status to *candidate*. It waits until the end of the current block of 2ℓ rounds in time thread 2 and performs the following procedure, similar to procedure `L_Elect`, lasting the entire next block of 2ℓ rounds in this time thread. At the end of the executed procedure the node becomes non locked.

procedure `Conf_Res`

- in round $2j+1$, where $0 \leq j < \ell$, if the node is a *candidate* and additionally the jth bit of the label of the token carried by the agent hosted by this node is 1 then the node transmits the contact message, otherwise it acts as a receiver;

- in round $2j+2$, where $0 \leq j < \ell$, if the node heard signal μ in round $2j+1$ then it transmits the contact message, otherwise it acts as a receiver;
- if signal μ has been heard in round $2j+1$ or $2j+2$ by a *candidate* node that did not transmit in round $2j+1$, where $0 \leq j < \ell$, then it changes its status to *non-candidate*;
- after 2ℓ rounds, if a node is *non-candidate* then the agent hosted by the node is erased.

3.3 The Algorithm without Local Knowledge

We now present algorithm `Ma_LE`, which does not assume that nodes know their neighbors, however it still assumes a known upper bound n on the number of nodes in the (unknown) network. (This assumption will in turn be removed at the end of this section.) While it is possible to elect a sublinear number of local leaders as before, the remote token elimination part must be different from that in algorithm `Known_Neighb_LE`, as it is impossible to use Standard DFS. In order to overcome this difficulty, we first add a new preprocessing part (Part 0) in which the entire graph is divided, in a distributed way, into clusters spanned by nodes of small and of large degrees, and further an additional size criterion is applied to classify clusters with nodes of large degrees. We call this newly introduced clustering the *fuzzy-degree clustering*, as it mainly depends on node degrees and the main combinatorial tool used to compute it are fuzzy separating families. In the first type of clusters, whose nodes have small degrees, it is possible to learn the neighborhood (using strongly selective families), and there we can build a DFS route visiting all the nodes of the cluster, which permits to eliminate tokens during the remote token elimination part, similarly as in algorithm `Known_Neighb_LE`. In the second type of clusters, which are of relatively small size, it may be too costly to learn neighborhoods, but still a DFS route can be built using the `blind_DFS` procedure. In the third type of clusters, which are large and contain nodes of large degrees, a slightly different method must be used: tokens are eliminated using "waves of noise". In each cluster a *representative* is selected. Then local leaders are chosen, using procedure `L_Elect`, from among representatives (Part 1). Next a more complex remote token elimination part (Part 2), working in parallel in previously prepared clusters and taking care of the difficulties mentioned above, permits to eliminate all local leaders except one. As before, the final leader is the node with the largest label.

Algorithm `Ma_LE`

Part 0. Computing fuzzy-degree clustering

Stage 1. Let $\mathcal{F} = \{F_1, \ldots F_k\}$ be a $(N, n^{1/4}, n^{1/2}/\log N, c)$-fuzzy separating family of size $\alpha = O(n^{1/2} \operatorname{polylog}(N))$, with $c = O(n^{1/2} \operatorname{polylog}(N))$, given by Lemma 1. In the ith round of Stage 1, for $1 \leq i \leq \alpha$, nodes with labels in set F_i transmit the contact message. By the definition of a fuzzy separating family, we get the following classification of nodes: nodes that heard silence during at most c rounds - we call such nodes *large*, and nodes that heard silence during more than c rounds - we call such nodes *small.* It can be proved that nodes of degree at least $n^{1/2}/\log N$ are large and nodes of degree smaller than $n^{1/4}$ are small. Note that nodes of intermediate degree may be classified either as small or as large, due to the "fuzziness" of the separating family. This, however, will not affect our considerations.

Stage 2. Let $\mathcal{S} = \{S_1, \ldots S_\beta\}$ be a $(N, n^{1/2}/\log N)$-strongly selective family of size $\beta = O((n^{1/2}/\log N)^2 \log N) = O(n/\log N)$. In the ith round of Stage 2, for $1 \leq i \leq \beta$, nodes with labels in set S_i transmit their label. It can be easily proved that upon completion of this stage all small nodes have heard the message of each of their neighbors, and since small nodes know that they are small, each such node knows that it heard from all of its neighbors.

Define *white clusters* to be connected components of the graph spanned by small nodes and *red clusters* to be connected components of the graph spanned by large nodes. Notice that red clusters have diameter at most $3n^{3/4}$.

Stage 3. This stage is executed in two time threads: thread 1 for small nodes and thread 2 for large nodes.

Time thread 1 (small nodes): Call algorithm `Known_Neighb_LE` in all white clusters in parallel, to elect a leader of each white cluster. (Recall that nodes in white clusters know all their neighbors.) Call the computed leader of each white cluster its *representative*.

Time thread 2 (large nodes): Call procedure `L_Elect`(X,k) in all red clusters in parallel, where X is the set of large nodes and $k = 3n^{3/4}$. Since red clusters have diameter at most $3n^{3/4}$, this procedure elects a single node in each red cluster. Call this node the *representative* of the cluster.

Thus, upon completion of Stage 3, a unique representative is elected in each cluster.

Stage 4. Each representative of a red cluster runs in parallel `blind_DFS` for n rounds. If this DFS was successful (all nodes of the cluster were visited) the red cluster is re-colored gray; if it was unsuccessful, the red cluster is re-colored black. Re-coloring is done in such a way that if `blind_DFS` was successful, another run of `blind_DFS` is initiated for n rounds by the representative whose agent carries the re-coloring message. If a node does not receive such an agent within the n rounds dedicated to this process, it re-colors itself black; otherwise it re-colors itself gray. At the end of the stage, the token issued by the representative comes back to it. Note that gray clusters are those clusters of large nodes that have sufficiently few nodes for `blind_DFS` to visit all of them within time n, and black clusters are all other clusters of large nodes.

This concludes Part 0 of the algorithm. At the end of this part there are three types of clusters in the graph together with their representatives:
(i) white, consisting of small nodes, in which a complete DFS route is established,
(ii) gray, consisting of large nodes, in which a complete DFS route is established, and
(iii) black, consisting of large nodes, in which DFS failed.

Part 1. Election of local leaders

Call procedure `L_Elect`(X,k), for X equal to the set of all cluster representatives and $k = 2n/\log N$. The output is the nonempty set Y of at most $\log N$ local leaders: nodes in Y have status *local leader* and all other nodes in the graph have status *non local leader*.

Part 2. Remote token elimination

This part is devoted to eliminating all local leaders except one, which becomes the leader. As in the auxiliary algorithm `Known_Neighb_LE`, each local leader issues a token that has its label. Since local leaders are chosen among cluster representatives, there is at most one local leader per cluster. Unlike in the auxiliary algorithm, tokens are not always carried by agents. Instead, they are carried by an agent inside gray or white clusters, propagated using the procedure `L_Elect` within black clusters, and transfered between clusters using the procedure `Conf_Res`. In each white or gray cluster there is a single agent that carries the largest token met in the cluster and this agent moves along a DFS route fixed for the cluster by its representative. As usual, nodes forget all tokens except the largest one they have seen to date.

Part 2 is organized in four time threads: one thread for actions in each type of clusters (white, gray and black), and one thread for inter-cluster communication. We call these threads white, gray, black and inter-cluster, respectively. Each thread consists of $9n$ rounds. Only nodes with the color of a time thread participate in this thread, while others are idle. During the inter-cluster time thread all nodes are active. Recall that clusters of the same color are never neighboring (by an edge), hence in time threads dedicated to communication within clusters of the same color (either white or gray or black), nodes is one cluster do not interfere with nodes in other clusters of the same color (i.e., their transmissions do not cause collisions in nodes in other clusters of the same color). On the other hand, interferences between nodes of different colors, which may be neighboring by an edge, are avoided by the definition of "color" time threads.

Time thread white/gray:

In white and gray clusters each representative sends an agent which travels cyclicly along the DFS route (prepared in Part 0). The agent carries some token which may change during its travel. In each round one step is done by the agent along the DFS route. If the representative is a local leader, the agent starts carrying the token with the representative's label, otherwise it starts carrying a default token 0. Whenever the agent visits a node that remembers a larger token (sent from a neighboring cluster), the currently carried token is destroyed and the larger token is further carried by the agent along the DFS route. Visiting nodes that remember smaller tokens than the one carried by the agent do not affect the carried token.

Time thread black:

In black clusters, where there is no prepared route to circulate tokens, smaller tokens are eliminated remotely. This is done using procedure $\texttt{L\_Elect}(X,k)$, where X is the set of all black nodes and $k = 3n^{3/4}$, which is an upper bound on the diameter of a black cluster. Each node uses the largest of all labels it has seen as its name when running procedure $\texttt{L\_Elect}(X,k)$. Let X' be the set of nodes that remain qualified at the end of the procedure. Note that since k is the upper bound on the diameter of the black cluster, all nodes in X' have seen the same largest label. Next the procedure $\texttt{L\_Elect}(X',k)$ is run for set X' and $k = 3n^{3/4}$, and, similarly as before, nodes in X' use the largest of all labels they have seen as their name in this procedure. During the execution of the procedure, all black nodes not in X' additionally extract a label from the execution of the procedure. More precisely, a node writes 1 in position i of the binary representation if it hears signal μ in the ith stage of the procedure, otherwise it writes 0. All nodes of the cluster have now seen the same largest label. This block of $2k\ell$ rounds, consisting of procedures $\texttt{L\_Elect}(X,k)$ and $\texttt{L\_Elect}(X',k)$, is repeated $9n/(2k\ell)$ times in the black time thread.

Time thread inter-cluster:

Transferring tokens between clusters is done in the inter-cluster thread. It does not guarantee the transfer of every token, however the tokens which are located on the border of a cluster and are locally the largest ones will be successfully transfered to the neighboring clusters. Each node executes the following extended procedure $\texttt{Conf\_Res}$ in a cyclic way. One cycle lasts 3ℓ rounds. During the first 2ℓ rounds the procedure

`Conf_Res` is run. At the beginning of this run, the node fixes its current token. At the end of the procedure, each node has status either candidate or non-candidate. In the remaining ℓ rounds each candidate node *keeps transmitting control messages* according to the label of its token (i.e., transmits only in rounds i such that there is 1 in position i). A node receiving signal μ in round i of the last ℓ rounds writes 1 in position i of its new token's label, otherwise it puts 0. At the end it compares the label of its new token with the ones arriving during the execution of the cycle and remembers only the maximal one.

Theorem 1. *Algorithm* `Ma_LE` *elects a leader in any ad hoc radio network with at most n nodes in time $O(n)$, assuming that all nodes know parameters n and N.*

We finally extend the `Ma_LE` algorithm, removing the assumption that a linear upper bound n on the number of nodes is known to stations a priori. We apply a standard doubling technique in which algorithm `Ma_LE` is run under the assumptions that the number of participating nodes is *at most* 2^i, for consecutive integers $i > 0$, until the leader is successfully elected. We show that this happens not later than when the estimate 2^i becomes larger than n. Additionally, we implement a distributed procedure for checking whether this process should be stopped after the current run of algorithm `Ma_LE`, or if it should be continued for subsequent powers of 2 as estimates for the number of nodes.

Theorem 2. *There exists an algorithm electing a leader in any ad-hoc n-node radio network in time $O(n)$, for any unknown integer $n > 0$.*

References

1. Alon, N., Bar-Noy, A., Linial, N., Peleg, D.: A lower bound for radio broadcast. J. of Computer and System Sciences 43, 290–298 (1991)
2. Bar-Yehuda, R., Goldreich, O., Itai, A.: On the time complexity of broadcast in radio networks: an exponential gap between determinism and randomization. J. of Computer and System Sciences 45, 104–126 (1992)
3. Capetanakis, J.: Tree algorithms for packet broadcast channels. IEEE Transactions on Information Theory 25, 505–515 (1979)
4. Chlamtac, I., Kutten, S.: On broadcasting in radio networks - problem analysis and protocol design. IEEE Trans. on Communications 33, 1240–1246 (1985)
5. Chlebus, B., Gąsieniec, L., Gibbons, A., Pelc, A., Rytter, W.: Deterministic broadcasting in unknown radio networks. Distributed Computing 15, 27–38 (2002)
6. Chrobak, M., Gasieniec, L., Rytter, W.: Fast broadcasting and gossiping in radio networks. In: Proc. 41st Symp. on Foundations of Computer Science (FOCS 2000), pp. 575–581 (2000)
7. Clementi, A.E.F., Monti, A., Silvestri, R.: Distributed broadcast in radio networks of unknown topology. Theor. Comput. Sci. 302, 337–364 (2003)
8. Czumaj, A., Rytter, W.: Broadcasting algorithms in radio networks with unknown topology. In: Proc. 44th Symp. on Foundations of Computer Science (FOCS 2003), pp. 492–501 (2003)
9. De Marco, G.: Distributed broadcast in unknown radio networks. In: Proc. 19th ACM-SIAM Symp. on Discrete Algorithms (SODA 2008), pp. 208–217 (2008)
10. Dessmark, A., Pelc, A.: Broadcasting in geometric radio networks. Journal of Discrete Algorithms 5, 187–201 (2007)
11. Erdos, P., Frankl, P., Furedi, P.: Families of finite sets in which no set is covered by the union of r others. Israel J. of Math. 51, 79–89 (1985)

12. Gąsieniec, L., Pelc, A., Peleg, D.: The wakeup problem in synchronous broadcast systems. SIAM J. on Discrete Mathematics 14, 207–222 (2001)
13. Greenberg, A.G., Winograd, S.: A lower bound on the time needed in the worst case to resolve conflicts deterministically in multiple access channels. J. ACM 32, 589–596 (1985)
14. Hayes, J.F.: An adaptive technique for local distribution. IEEE Transactions on Communications 26, 1178–1186 (1978)
15. Indyk, P.: Explicit constructions of selectors and related combinatorial structures, with applications. In: Proc. 13th ACM-SIAM Symp. on Discrete Algorithms (SODA 2002), pp. 697–704 (2002)
16. Jurdzinski, T., Kutylowski, M., Zatopianski, J.: Efficient algorithms for leader election in radio networks. In: Proc. 21st ACM Symp. on Principles of Distr. Comp (PODC 2002), pp. 51–57 (2002)
17. Kautz, W.H., Singleton, R.R.C.: Nonrandom binary superimposed codes. IEEE Trans. on Inf. Theory 10, 363–377 (1964)
18. Kowalski, D., Pelc, A.: Time of deterministic broadcasting in radio networks with local knowledge. SIAM J. on Computing 33, 870–891 (2004)
19. Kowalski, D., Pelc, A.: Broadcasting in undirected ad hoc radio networks. Distributed Computing 18, 43–57 (2005)
20. Kushilevitz, E., Mansour, Y.: An $\Omega(D \log(N/D))$ lower bound for broadcast in radio networks. SIAM J. on Computing 27, 702–712 (1998)
21. Lynch, N.A.: Distributed Algorithms. Morgan Kaufmann Publ., Inc., San Francisco (1996)
22. Nakano, K., Olariu, S.: Uniform leader election protocols for radio networks. IEEE Transactions on Parallel Distributed Systems 13, 516–526 (2002)
23. Ta-Shma, A., Umans, C., Zuckerman, D.: Loss-less condensers, unbalanced expanders, and extractors. In: Proc. 33rd ACM Symposium on Theory of Computing (STOC 2001), pp. 143–152 (2001)
24. Tsybakov, B.S., Mikhailov, V.A.: Free synchronous packet access in a broadcast channel with feedback. Prob. Inf. Transmission 14, 259–280 (1978)
25. Willard, D.E.: Log-logarithmic selection resolution protocols in a multiple access channel. SIAM J. on Computing 15, 468–477 (1986)

Secure Function Collection with Sublinear Storage

Maged H. Ibrahim[1,⋆], Aggelos Kiayias[2,3], Moti Yung[4], and Hong-Sheng Zhou[3,⋆⋆]

[1] Faculty of Engineering, Helwan University, 1 Sherif St., Helwan, Cairo, Egypt
mhii72@hotmail.com.

[2] Dept. of Informatics and Telecommunications, U. of Athens, Greece

[3] University of Connecticut, Computer Science & Engineering, Storrs, CT, USA
{aggelos,hszhou}@cse.uconn.edu.

[4] Google Inc. and Computer Science, Columbia University, New York, NY, USA
moti@cs.columbia.edu.

Abstract. Consider a center possessing a trusted (tamper proof) device that wishes to securely compute a public function over private inputs that are contributed by some network nodes. In network scenarios that support direct communication of nodes with the center, the computation can be done by the nodes encrypting their inputs under the device's public key and sending the ciphertexts to the center which, in turn, feeds them to the trusted device that computes the function. In many modern networking scenarios, however, the center and the contributing nodes are not directly connected/connectable, in which case the discovery and collection of inputs can only be performed by an agent (or agents) released to the network by the center. This introduces a new set of issues for secure computation. In this work we consider an agent that is released, sweeps the network once and then returns to its origin. The direct solution, in this case, is for the agent to possess a certified public key of the trusted device and for the nodes to contribute their inputs as ciphertexts under this key; once the agent collects all inputs it reconnects with the center for function computation. The above single-sweep simple collection requires the agent to store a linear number of ciphertexts. The goal of this work is to formalize and solve the above problem for a general set of functions by an agent that employs sub-linear storage while maintaining input privacy (an important technical requirement akin of that of "Private Information Retrieval" protocols).

1 Introduction

Secure multi-party computations is a general paradigm introduced by Goldreich, Micali and Wigderson [9]. It models a group of network nodes wishing to compute a public function on private inputs (or a private function on a universal

[⋆] Research partly supported by Egyptian Culture and Education Bureau (ECEB) Washington-DC.

[⋆⋆] Research partly supported by NSF awards 0447808, 0831304, 0831306.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 534–545, 2009.

circuit), producing the correct result, while a faulty subset of the nodes cannot learn from the computation more than what can be deduced from the output availability. In a model allowing nodes to also deviate from the protocol, the computation also requires that the faulty subset cannot hurt the computation. This paradigm is central in cryptography and secure network protocols, and has produced numerous variations. For example, recently, it was advocated to develop solutions that take into account specific communication environments for improved efficiency [14].

In this work we consider a setting for multi-party computations which is motivated by two modern phenomena: The first is modern network environments and network processing settings where network nodes are not always connected or directly reachable by a global routing infrastructure (ad hoc networks, sensor networks), or where applications at a network node are activated only when being approached by an agent (e.g., web crawling). The second motivation is the availability of trusted computing platforms representing a trusted tamper resistant computational device. Indeed, building cryptographic systems based on trusted and tamper proof implementations and cryptographically fortifying such implementations is an area of increasing recent interest [13,12,8].

The problem we consider is the setting where a server (called the "center") M in possession of a tamper-proof cryptographic device (that offers elementary public-key signing and decryption functionality) wishes to compute a function described by a program P on a set of m inputs residing in network units $\mathcal{U} = \{U_1, ..., U_m\}$, to which the center has no direct interaction with. The task of collecting the inputs is given to an agent A that visits the units. The computation is required to maintain input security and to have certain robustness properties as well. The setting, which can be called "agent-oriented secure function evaluation" gives rise to new problems in the area of secure computing based on various constraints and limitations that can be put on the parties by varying the problem's parameters. In this work we consider the case of an agent that is limited to returning to the center once after sweeping the network nodes, in which case the interesting challenge is to employ an agent with sub-linear number of ciphertexts.

We formalize the above problem which we call "secure collection of a function" (SCF). We want to achieve the following security and integrity properties:

- *Unit Privacy against Malicious Agent:* Even if the agent is malicious it cannot violate the privacy of the units; furthermore,
- *Unit Privacy against Curious Center:* While the center is trusted in our framework (since it is the interested party) we still wish to maintain the privacy of the units against in the honest-but-curious sense. We note that this can be very useful also in the setting where the trusted device employed by the center simulated by a quorum of entities (employing threshold cryptography techniques shared among trusted bases).
- *Integrity in the presence of Malicious Units:* If any number of units turn malicious they cannot collude to disrupt or spoil the computation (except by choosing their private inputs arbitrarily).

- *Computation Privacy in the presence of Malicious Units:* If any number of units turn malicious and are given the contents of the agent they cannot learn the state of the computation (beyond what can be deduced from their own contributions and the size of the state's encoding).

Based on the above, from a security point of view, the agent is oblivious to the contributions of the units and the state of the computation. Its content only reveal some encoding of state. From a robustness point of view, we note that changing the state of the agent can be detected by keeping local copies at neighbors (such extensions are left for future work). Further, in the worse case, this amounts to a total denial of service (namely, elimination of the agent), but it will not violate the privacy of the units. Mitigating the denial of service resulting by agent capturing can only be coped with by releasing a multitude of agents and we do not address these aspects at present (such variations are also suggested future work).

Our contributions and results. Here we present the concrete results we achieve. As mentioned earlier, a sweeping agent can collect all inputs in ciphertext form, a solution that requires it to carry a linear number of ciphertexts. Our goal then is to reduce this storage requirement for a large set of functions. Given our setting, we choose to implement the branching program model for representing and computing our function f. We show that this model can lead to working solutions within the constraints on agent's storage as well as those on the computational complexity imposed by the SCF setting. We employ a Paillier type encryption [19] assuming the decisional composite residuosity (DCR) assumption.

To solve our problem, we develop a new cryptographic primitive that efficiently computes a function f in the branching program model. We call it "1-out-of-2 private table evaluation" (PTE_2^1). In this primitive, the agent A has input two tables ($\mathcal{T}^{(0)}$ and $\mathcal{T}^{(1)}$). It delivers an encrypted version of the two tables to the unit. The unit, in turn, makes a selection according to her private input bit b, and delivers a re-encryption of her choice back to A.

Private table evaluation enables the agent to walk along a branching program layer by layer in an oblivious fashion without learning in cleartext the actual state of the computation. At the end of the protocol (after visiting all units) the agent returns an encryption of the leaf representing the output of the program to the center who can then employ the trusted device to decrypt and obtain the result. During the protocol, the agent stores only an encryption of the current table. A nice property of this primitive is that the units need not have any public keys and they employ the key of the trusted device. We give an efficient implementation of the PTE_2^1 that yields an SCF protocol achieving a complexity (both in terms of communication as well as in the size of the active read/write storage of the agent) in the order of $\mathcal{O}(k \cdot w_{max})$ where k is the security parameter of the underlying homomorphic public key encryption scheme and w_{max} is the maximum width of the branching program. For a large set of branching programs this value is independent of the number of units m. For example for NC^1 programs we can get memory linear in the security parameter k due to Barrington's theorem [1].

Another appealing property of the PTE_2^1 is that, with the help of well known and suitable proofs of knowledge and with agent signing commitments to input upon its first visit we can design efficient solutions against malicious behavior of the units without hurting the complexity of the honest-but-curious protocol. The total run-time of the computation is proportional to the size of the branching program.

We show how to perform general function evaluation in this model. We then show how to implement SCF for concrete problems that are of practical importance in the setting of network sweeping: (i) polling-to-partition procedures, that can simulate a wide array of distributed decision-making and polling operations, and (ii) pattern and string matching procedures typical in search problems. Finally, we note that to further hide the circuit structure from units, the agent can perform dummy PTE_2^1 operations, in a random walk, mixing it with real computations.

Related Work. Secure multiparty computation was initiated and studied by Goldreich, Micali and Wigderson [9] in a fully connected (broadcast links). This was followed by information theoretic protocols assuming (added) private links [2,4,20]. Various models with partial connectivity were then considered in [6,14,7]. In our setting we take the approach to restrict even further the connectivity (by not assuming a complete routing infrastructure as in ad hoc scenarios). We further employ a trusted device (also employed in recent work on employing trusted hardware in [13,10]). We next note that our problem bears some relation to private information retrieval [5,16,3] where a client wishes to extract a record from a database in sublinear communication. The distinction in our setting is that the role of the database is distributed to a set of units and the role of the communication channel is played by the agent that interacts with the units.

The use of the branching program model of computation for reducing the communication complexity in secure computations between two curious parties was put forth by Naor and Nissim in [18]. Note that the units are not supposed to coordinate with each other, so their technique cannot be directly adapted for our setting. Their model was also utilized in the context of sublinear communication oblivious transfer in [17] and for general two party computation in [11].

Paper Organization. In section 2 we present the formulation of the problem and the security model. Next, in section 3 we show our basic secure collection of a function protocol construction using private table evaluation as a building block. An explicit private table evaluation protocol is then presented in section 4. Due to lack of space, we leave in the full version the examples of our framework, the zero-knowledge proof protocol for enforcing security against malicious units as well as proofs of our claims.

2 Problem and Model

We first introduce the general formulation of our problem.

Definition 1. *The Collecting a Function (CF) problem: A* center *M wishes to compute a function $f : X^m \rightarrow Z$ on inputs $x_1, \ldots, x_m$ that are distributed to a set of units $\mathcal{U} = \{U_1, ..., U_m\}$. The units communicate with the center indirectly through black-box interaction with an agent A.*

A communication pattern Π for a CF problem is a string P from $\{U_1, \ldots, U_m\}^*$ that specifies the order with which the agent interacts with the units. Note that the agent must also interact two times with the center, one time when the agent is dispatched to collect the information (initialization) and another time when the agent delivers the results to the center.

A secure CF (SCF) protocol for a function $f : X^m \rightarrow Z$ and a communication pattern P is a triple $\langle \mathsf{Init}, \mathsf{Final}, (\pi_A, \pi_U)\rangle$ such that Init *receives a description of f and a security parameter and produces a public/secret-key pair (pk, sk); (π_A, π_U) is a two-party protocol where in the j-th round, ($j \in \{1, \ldots, |P|\}$), the agent running π_A on input (s_{j-1}, pk) and a description of f interacts with the unit U_t, where $t \in \{1, \ldots, m\}$ and U_t is the j-th symbol of P, running protocol π_U on input (s'_{j-1}, x_t); the two parties terminate returning private outputs s_j and s'_j respectively. Note that $s_0 := \epsilon$, and $s'_0 := pk$.* Final *receives the string $s_{|P|}$ and sk and returns a value in the range of f.*

An SCF protocol is correct if for all $x_1, \ldots, x_m \in X$ it holds that Final *in the computation as described above returns the value $f(x_1, \ldots, x_m)$ always.*

Security Properties for SCF protocols. Next we introduce the security and integrity properties for SCF protocols.

- **Unit Privacy against a Malicious Agent.** For any $i = 1, \ldots, m$, and for any PPT adversary Adv playing the role of the agent A and all units $\{U_1, \ldots, U_m\} \setminus \{U_i\}$, there exists an expected polynomial-time simulator Sim, such that for any $x_1, \ldots, x_m \in X$, the output of Sim is computationally indistinguishable to the view of the adversary Adv that includes all protocol transcripts and private tapes of corrupted parties. The input of Sim is equal to the input of the (corrupted) agent A and the (corrupted) units $\{x_1, \ldots, x_m\} \setminus \{x_i\}$.
- **Unit Privacy against a Honest-but-Curious Center.** There exists an expected polynomial-time simulator Sim such that for any $x_1, \ldots, x_m \in X$, the output of Sim is identical to the distribution of the internal state of the agent at the end of all interactions with the units. The input of Sim is equal to the value $f(x_1, \ldots, x_m)$.
- **Computation Integrity in the presence of Malicious Units.** There is a deterministic implicit input function IN, such that for any PPT adversary Adv, playing the role of all units $\{U_1, \ldots, U_m\}$, it holds that for any $x_1, \ldots, x_m \in X$, if the agent terminates successfully then the output of the center M equals $f(x_1, \ldots, x_m)$ where $(x_1, \ldots, x_m) = \mathsf{IN}(\tau)$ where τ is the total communication transcript of all the interactions between the agent and the units. Note that IN is not required to be polynomial-time.
- **Computation Privacy in the presence of Malicious Units.** For any PPT adversary Adv playing the role of all units $\{U_1, \ldots, U_m\}$ that is

capable of corrupting the agent at a certain point of the system execution and receiving its content, there is an expected polynomial-time simulator Sim, such that for any $x_1, \ldots, x_m \in X$, the output of Sim is computationally indistinguishable to the view of the adversary. The input of Sim is equal to $x_1, \ldots, x_m$ as well as a description of the function f.

Efficiency Parameters of SCF protocols. For an SCF protocol $\langle \mathsf{Init}, \mathsf{Final}, (\pi_A, \pi_U) \rangle$ we are primarily interested in the following efficiency measures:

- **Agent Storage.** Is the maximum number of storage bits required by the agent quantified over all possible CF computations and coin tosses.
- **Agent-Unit Communication.** Is the maximum number of bits communicated between the unit and the agent quantified over all possible SCF protocol computations and coin tosses.

Naturally, the time complexity of agent, units and the center is also of interest.

3 Designing SCF Protocols from Branching Programs

We first describe the notion of layered branching programs for modeling multiparty protocols. The definition below is a straightforward generalization of the notion of two party branching programs used for modeling protocols in the communication complexity model as given in [15]. We define how m parties can compute a function $f : X^m \to Z$ based on their inputs and broadcast communication.

Definition 2 (Multiparty computation in the Branching Program model (BPM)). *A (layered) branching program P for the function $f : X^m \to Z$ and parties $U_1, \ldots, U_m$ is a layered directed graph $G = (V = (L_0, ..., L_c), E)$ of depth c where $|L_0| = 1$. Each edge in E connects layer L_ℓ to layer $L_{\ell+1}$ where $\ell \in \{0, \ldots, c-1\}$ and is labeled by an element of X so that each vertex has exactly $|X|$ outgoing edges all labeled distinctly. Further, each layer is labeled by a party $U \in \{U_1, \ldots, U_m\}$ and each node in L_c is labeled with an element $z \in Z$.*

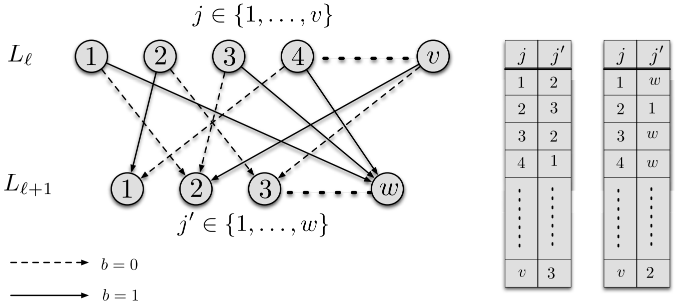
The output of the program P on input $(x_1, ..., x_m)$ where $x_i \in X$ is the label of the leaf reached by starting at the single node at the L_0 level and traversing the graph following the appropriately labeled edges at each level based on the input of the party that labels the layer. We say that the branching program P computes $f : X^m \to Z$ if its value on input $(x_1, ..., x_m)$ equals $f(x_1, ..., x_m)$. The cost of the program is c, its width is $\max |L_\ell|$.

A branching program P can capture multiparty computation of a function f in the following way: the party U_i that labels the top layer L_0 that contains a single node v_0 computes the next node following the edge labeled according to its input and broadcasts the index of the next node to the other parties. Next, the party at level L_1 continues recursively based on the communication transcript and the computation proceeds recursively until all parties terminate at the leaf level. The overall communication complexity equals c broadcasts of at most $\log_2 \max\{|L_\ell|\}$ bits.

In our setting the units do not interact with one another but rather the broadcast channel can be simulated by interacting with the agent. In particular:

- Given the branching program P of cost c for the function $f : X^m \rightarrow Z$, we derive a communication pattern $\varPi \in \{U_1, \ldots, U_m\}^c$ based on the labels of the layers of P.
- The agent maintains a state that corresponds to a node of the branching program P. Initially this node is set to be single node v_0 of the top layer L_0.
- The agent, while at a state corresponding to the v-th vertex at level L_ℓ for $\ell \in \{0, \ldots, c-1\}$, visits the next unit U_i following the order of the communication pattern $\varPi$ and executes a protocol with U_i that enables the agent to compute the index of the vertex in level $L_{\ell+1}$ in the layered branching program P that is determined by the outgoing edge of v labeled by x_i where x_i is the input of the unit U_i.
- The computation proceeds recursively until the agent reaches the last layer of the branching program P that reveals the value $f(x_1, \ldots, x_m)$.

In the remaining of the section for simplicity we will focus on the special case that $X = \{0, 1\}$. Extending our results to the general case is straightforward. We next consider how one can add privacy to the above CF computation strategy. We first observe that each level of the branching program can be represented by a set of tables $\mathcal{T}^{(0)}, \mathcal{T}^{(1)}$. We depict this in figure 1.



j	j'
1	2
2	3
3	2
4	1
$\vdots$	$\vdots$
v	3

j	j'
1	w
2	1
3	w
4	w
$\vdots$	$\vdots$
v	2

Fig. 1. An arbitrary level in a fragment of branching program (on the left) and the corresponding tables (on the right). The choice of a table depends on U's input b.

As described above, at each layer of the computation, the agent and the unit will utilize a protocol that we call *1-out-of-2 private table evaluation* (PTE). This is a two-party protocol that relies on a suitable public-key encryption scheme $\langle \mathcal{G}, \mathcal{E}, \mathcal{D} \rangle$. The agent will be given its input j in encrypted form as $\mathcal{E}_{pk}(j)$ and in the course of the interaction with the unit it expects to update it to the encryption $\mathcal{E}_{pk}(\mathcal{T}^{(b)}[j])$ where b is the unit's input.

The two tables $\mathcal{T}^{(0)}, \mathcal{T}^{(1)}$ are selected from $\{1, \ldots, w\}^v$ where $v, w \in \mathbb{N}$ are parameters of the protocol. For greater generality in our protocol description we will also allow the underlying encryption scheme used to be parameterized and use the notation $\mathcal{E}^v_{pk}(m)$ for any $v \in \mathbb{N}$. Pictorially we have the following:

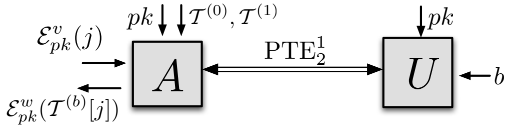

$\mathcal{E}^v_{pk}(j)$
$\mathcal{E}^w_{pk}(\mathcal{T}^{(b)}[j])$
pk
$\mathcal{T}^{(0)}, \mathcal{T}^{(1)}$
A
PTE^1_2
pk
U
b

Fig. 2. 1-out-of-2 private table evaluation

Definition 3 (1-out-of-2 private table evaluation (PTE^1_2)**).** *The protocol is based on parameters $v, w \in \mathbb{N}$ and is executed by agent A and unit U based on a public key encryption $\langle \mathcal{G}, \mathcal{E}, \mathcal{D} \rangle$. First an external party generates a key pair $(pk, sk) \leftarrow \mathcal{G}(1^\lambda)$, and delivers pk to both agent A and unit U. Second A interacts with U; here A has input $\mathcal{E}^v_{pk}(j)$ where $j \in \{1, \ldots, v\}$, and $\mathcal{T}^{(0)}, \mathcal{T}^{(1)} \in \{1, \ldots, w\}^v$, and U has the input $b \in X$. Finally only the agent outputs $\mathcal{E}^w_{pk}(j')$ where $j' = \mathcal{T}^{(b)}[j] \in \{1, \ldots, w\}$.*

We say that a PTE^1_2 protocol is secure, if the following properties are satisfied. In all cases below it is assumed that the input of both parties includes the public-key pk that is honestly generated and neither party is in possession of the secret-key.

Integrity. Informally, we require that the unit can only contribute in terms of selecting one of the two tables that the agent possesses and is not capable – even if acting maliciously – to influence the agent to derive an encryption of any value other than an encryption of j'. More formally, we require that even if the unit is malicious it holds that the output of the agent is distributed uniformly either over all ciphertexts of the form $\mathcal{E}_{pk}(\mathcal{T}^{(0)}[j])$ or all ciphertexts of the form $\mathcal{E}_{pk}(\mathcal{T}^{(1)}[j])$.

Unit Privacy. Informally, we require that the input of the unit remains hidden from the agent. More formally, for any PPT adversary Adv corrupting the agent A, there exists an expected polynomial-time simulator Sim, such that for any $x \in X$ the output of Sim is computationally indistinguishable to the view of the adversary Adv that includes protocol transcripts and private tapes of the corrupted agent. The input of Sim equals the corrupted agent's input.

Agent Privacy. Informally, we require that the two input tables $\mathcal{T}^{(0)}, \mathcal{T}^{(1)}$ of the agent remain hidden from the unit. More formally, for any $x \in X$, and any PPT adversary Adv corrupting the unit U, there exists an expected polynomial-time simulator Sim, such the output of Sim is computationally indistinguishable to the view of the adversary Adv that includes protocol transcripts between the unit U and the agent A.

Our Design Strategy. Based on an implementation of PTE^1_2 we can derive a SCF protocol for any function $f : X^m \to Z$ as follows. Let P be a branching program for computing f with communication pattern Π. We assume each unit U_i holds an input $b_i \in X = \{0, 1\}$. Our SCF system is illustrated in figure 3 and operates as follows:

- In the initialization process Init, given a security parameter k a key pair $(pk, sk) \leftarrow \mathcal{G}(1^k)$. The public key pk is supplied to the agent A and all units, and the secret key sk is only known by the center. M also gives A the branching program P for function f.
- According to the structure of P, A invokes the PTE_2^1 with each unit $U \in \{U_1, \ldots, U_m\}$. In particular at each layer of P (except the last) the agent defines two tables $\mathcal{T}^{(b)}[\cdot] \in \{1, \ldots, w\}^v$ where v is the width of the current layer and w is the width of the next layer as shown in figure 1.
- Using sk, Final decrypts the ciphertext to receive the decryption of the final index that also reveals the output of the computation.

Connecting the above to our broad modeling as given in the introduction the implementation of Init, Final is assumed internal to the center's trusted device.

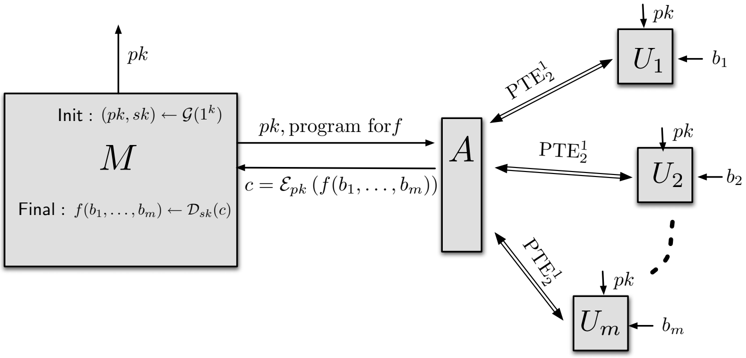

pk
Init : $(pk, sk) \leftarrow \mathcal{G}(1^k)$
M
Final : $f(b_1, \ldots, b_m) \leftarrow \mathcal{D}_{sk}(c)$
pk, program for f
$c = \mathcal{E}_{pk}(f(b_1, \ldots, b_m))$
A
PTE_2^1
PTE_2^1
PTE_2^1
pk
U_1
b_1
pk
U_2
b_2
pk
U_m
b_m

Fig. 3. The SCF protocol based on private-table evaluation

Theorem 1. *The SCF protocol defined above satisfies (i) unit privacy against a malicious agent under the assumption that the given PTE satisfies unit privacy (ii) unit privacy against an honest-but-curious center unconditionally (iii) computation integrity in the presence of malicious units under the assumption that the given PTE satisfies integrity, (iv) computation privacy in the presence of malicious units under the assumption that the given PTE satisfies agent privacy.*

4 Implementing Private Table Evaluation Efficiently

Our implementation of PTE_2^1 is based on an IND-CPA public-key encryption scheme with homomorphic property. Here we use the Paillier cryptosystem [19]; we note that the construction could be based on other homomorphic encryptions. Next we give a brief review of Paillier system $\langle \mathcal{G}, \mathcal{E}, \mathcal{D} \rangle$.

- **Key Generation**, $\mathcal{G}(1^k)$. On input a security parameter k, let p, q be random k-bit primes such that $p, q > 2$, $p \neq q$, and $\gcd(pq, (p-1)(q-1)) = 1$; let $N = pq$, $d = \mathrm{lcm}(p-1, q-1)$, $K = d^{-1} \bmod N$, and $h = (1+N)$; output a key pair (pk, sk), where $sk = \langle p, q \rangle$ and $pk = \langle N, h \rangle$.

- **Encryption**, $\mathcal{E}_{pk}(m)$. Define a plaintext space $\mathcal{M}_{pk} := \mathbb{Z}_N$ and a ciphertext space $\mathcal{C}_{pk} := \mathbb{Z}^*_{N^2}$. Upon receiving $m \in \mathcal{M}_{pk}$, randomly choose $r \in_R \mathbb{Z}^*_N$, and output the ciphertext $c = h^m r^N \bmod N^2$. We denote the ciphertext as $\mathcal{E}_{pk}(m)$ or $\mathcal{E}_{pk}(m; r)$.
- **Decryption**, $\mathcal{D}_{sk}(c)$. Upon receiving $c \in \mathcal{C}_{pk}$, compute $m = \frac{(c^{dK} \bmod N^2)-1}{N} \bmod N$. Note that d, K can be obtained from sk as in key generation.

The Paillier cryptosystem is IND-CPA secure [19] under the Decisional Composite Residuosity (DCR) assumption. It also enjoys an additive homomorphic property. For binary operators "+" over $\mathcal{M}_{pk}$ and "$\cdot$" over $\mathcal{C}_{pk}$, we have $\mathcal{E}_{pk}(m_1; r_1) \cdot \mathcal{E}_{pk}(m_2; r_2) = \mathcal{E}_{pk}(m_1 + m_2; r_1 r_2)$.

Observe that based on the homomorphic property, we can randomize a valid ciphertext c into another one c' while preserving the underlying plaintext m intact. In particular, c' can be obtained by multiplying c with a ciphertext for 0, i.e., $c' = c \cdot \mathcal{E}_{pk}(0; r_0) = \mathcal{E}_{pk}(m; r_m) \cdot \mathcal{E}_{pk}(0; r_0) = \mathcal{E}_{pk}(m; r_m r_0)$, where $r_m, r_0 \in_R \mathbb{Z}^*_N$. Notice that for any c it holds that c' is a random variable over the subset of $\mathcal{C}_{pk}$ that is of size $|\mathbb{Z}^*_N|$ and corresponds to all possible encryptions of the plaintext in c.

The parameterization we will use will be as follows: $\mathcal{E}^v_{pk}(m) = \langle \mathcal{E}_{pk}(m), \mathcal{E}_{pk}(m^2), \ldots, \mathcal{E}_{pk}(m^{v-1}) \rangle$, i.e., for any $v > 0$ it holds that $\mathcal{E}^v_{pk}(m)$ is a vector of $v - 1$ ciphertexts. Note that if $v = 1$ it holds that $\mathcal{E}^1_{pk}(m) = \langle \mathcal{E}(1) \rangle$ (independently of m).

4.1 Building Block: A PTE Construction

We present next our construction for private table evaluation that is utilizing Paillier encryption as defined in the previous section.

1. Generate $(pk, sk) \leftarrow \mathcal{G}(1^k)$, deliver pk to agent A and unit U. Then A and U operate as follows.
2. Upon receiving two tables $\mathcal{T}^{(0)}, \mathcal{T}^{(1)} \in \{1, \ldots, w\}^v$, and a vector of ciphertexts $\mathcal{E}^v_{pk}(j) = \langle C_1, \ldots, C_{v-1} \rangle$ where recall that $C_i = \mathcal{E}_{pk}(j^i)$ for $i = 1, \ldots, v-1$, $j \in \{1, \ldots, v\}$, the agent A chooses $2(w-1)$ polynomials with degree $v-1$ as: $g^{(0)}_1(x), \ldots, g^{(0)}_{w-1}(x), g^{(1)}_1(x), \ldots, g^{(1)}_{w-1}(x)$ over $\mathbb{Z}^*_N$, such that $g^{(b)}_\ell(j) = (\mathcal{T}^{(b)}[j])^\ell \bmod N$, for $j = 1, \ldots, v$, where $\ell \in \{1, \ldots, w-1\}$ and $b \in \{0, 1\}$. Notice that the polynomial $g^{(b)}_\ell(x)$ is computed as $g^{(b)}_\ell(x) = \sum_n \lambda^{(j)}_n (\mathcal{T}^{(b)}(n))^\ell$ where the $\lambda^{(j)}_n$'s are the Lagrange coefficients: for any polynomial p of degree less than v they satisfy $p(x) = \sum_{n=0}^{v-1} \lambda^{(x)}_n p(n)$. Let $a^{(b,\ell)}_0, \ldots, a^{(b,\ell)}_{v-1}$ be the coefficients of polynomial $g^{(b)}_\ell(x)$.
   Using the above set of polynomials and its input ciphertexts $\langle C_1, \ldots, C_{v-1} \rangle$ the agent computes $C^{(b)}_\ell = \mathcal{E}_{pk}(g^{(b)}_\ell(j)) = \prod_{i=0}^{v-1} (C_i)^{a^{(b,\ell)}_i}$, where $C_0 = \mathcal{E}_{pk}(1)$ for $\ell = 1, \ldots, w-1$. We denote $C^{(b)} = \langle C^{(b)}_1, \ldots, C^{(b)}_{w-1} \rangle$, for $b = 0, 1$. Observe that $C^{(b)} = \mathcal{E}^w_{pk}(\mathcal{T}^{(b)}[j])$. The agent sends $\langle C^{(0)}, C^{(1)} \rangle$ to unit U.
3. On input $b \in \{0, 1\}$, after receiving $\langle C^{(0)}, C^{(1)} \rangle$ from A, the unit randomizes $C^{(b)}$ into C' where $C' = \langle C'_1, \ldots, C'_{w-1} \rangle$ and $C'_\ell = C^{(b)}_\ell \cdot \mathcal{E}_{pk}(0; r_\ell)$ and $r_\ell \in_R \mathbb{Z}^*_N$ for $\ell = 1, \ldots, w-1$, and returns C' to agent A.

4. Finally agent A outputs C'.

Below we prove the security of the above protocol for honest units. In the next session we show how to *efficiently* handle malicious units. We show:

Theorem 2. *The PTE protocol defined above satisfies (i) integrity for honest units unconditionally, (ii) unit privacy under DCR assumption, (iii) agent privacy under DCR assumption.*

4.2 PTE against Malicious Units

To defend against a malicious unit, we need to enforce the unit to commit to its input b, i.e, $E = h^b t^N \bmod N^2$ where $t \in_R \mathbb{Z}_N^*$, and the commitment E is further to be signed by the agent, $\sigma = \mathtt{Sign}_{sk}(E)$ and deposited to the unit. Further the unit needs to prove that the vector of ciphertexts C' is randomization of $C^{(b)}$ and such b is the one committed in E, that means the unit should show a proof that it knows the witness of the following language

$$\mathcal{L} = \left\{ \begin{array}{l} \langle C_1^{(0)}, \ldots, C_{w-1}^{(0)}, C_1^{(1)}, \ldots, C_{w-1}^{(1)}, C_1', \ldots, C_{w-1}', E \rangle | \\ \qquad \exists b, r_1, \ldots, r_{w-1}, \beta : \\ \qquad\quad C_\ell'/C_\ell^{(0)} = (C_\ell^{(1)}/C_\ell^{(0)})^b \cdot r_\ell^N \bmod N^2 \text{ for } \ell = 1, \ldots, w-1, \\ \qquad\quad E = h^b \beta^N \bmod N^2 \end{array} \right\}$$

Note that the unit should save $\langle E, \sigma \rangle$ and in the case that the agent visits the same unit again, $\langle E, \sigma \rangle$ will be handed to the agent and the ZK proof will be based on this recorded E and newly produced vector of ciphertexts. The proof can be done by standard techniques, and the details can be found in the full version.

Theorem 3. *The PTE protocol defined above satisfies (i) integrity under the* `Sign` *unforgeability, (ii) unit privacy under DCR assumption, (iii) agent privacy under DCR assumption.*

Corollary 1. *The SCF protocol of section 3 with the PTE of the present section satisfies (i) unit privacy against a malicious agent under DCR assumption (ii) unit privacy against a semi-honest center unconditionally (iii) computation integrity in the presence of malicious units under* `Sign` *unforgeability, (iv) computation privacy in the presence of malicious units under DCR assumption.*

References

1. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. J. Comput. Syst. Sci. 38(1), 150–164 (1989)
2. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: STOC, pp. 1–10 (1988)

3. Cachin, C., Micali, S., Stadler, M.: Computationally private information retrieval with polylogarithmic communication. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 402–414. Springer, Heidelberg (1999)
4. Chaum, D., Crépeau, C., Damgård, I.: Multiparty unconditionally secure protocols (extended abstract). In: STOC, pp. 11–19 (1988)
5. Chor, B., Kushilevitz, E., Goldreich, O., Sudan, M.: Private information retrieval. J. ACM 45(6), 965–981 (1998)
6. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. J. ACM 40(1), 17–47 (1993)
7. Garay, J.A., Ostrovsky, R.: Almost-everywhere secure computation. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 307–323. Springer, Heidelberg (2008)
8. Gennaro, R., Lysyanskaya, A., Malkin, T.G., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
9. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or A completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
10. Hazay, C., Lindell, Y.: Constructions of truly practical secure protocols using standardsmartcards. In: ACM CCS, pp. 491–500 (2008)
11. Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 575–594. Springer, Heidelberg (2007)
12. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
13. Katz, J.: Universally composable multi-party computation using tamper-proof hardware. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 115–128. Springer, Heidelberg (2007)
14. Katz, J., Koo, C.-Y.: Round-efficient secure computation in point-to-point networks. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 311–328. Springer, Heidelberg (2007)
15. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)
16. Kushilevitz, E., Ostrovsky, R.: Replication is not needed: Single database, computationally-private information retrieval. In: FOCS, pp. 364–373 (1997)
17. Lipmaa, H.: An oblivious transfer protocol with log-squared communication. In: Zhou, J., López, J., Deng, R.H., Bao, F. (eds.) ISC 2005. LNCS, vol. 3650, pp. 314–328. Springer, Heidelberg (2005)
18. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: STOC, pp. 590–599 (2001)
19. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999)
20. Rabin, T., Ben-Or, M.: Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In: STOC, pp. 73–85. ACM Press, New York (1989)

Worst-Case Efficiency Analysis of Queueing Disciplines[⋆]

Damon Mosk-Aoyama[⋆⋆] and Tim Roughgarden[⋆⋆⋆]

Department of Computer Science, Stanford University, 353 Serra Mall, Stanford,
CA 94305
damonma@cs.stanford.edu, tim@cs.stanford.edu

1 Introduction

Consider n users vying for shares of a divisible good. Every user i wants as much of the good as possible but has diminishing returns, meaning that its *utility* $U_i(x_i)$ for $x_i \geq 0$ units of the good is a nonnegative, nondecreasing, continuously differentiable concave function of x_i. The good can be produced in any amount, but producing $X = \sum_{i=1}^n x_i$ units of it incurs a cost $C(X)$ for a given nondecreasing and convex function C that satisfies $C(0) = 0$. Cost might represent monetary cost, but other interesting interpretations are also possible. For example, x_i could represent the amount of traffic (measured in packets, say) that user i injects into a queue in a given time window, and $C(X)$ could denote aggregate delay ($X \cdot c(X)$, where $c(X)$ is the average per-unit delay). An altruistic designer who knows the utility functions of the users and who can dictate the allocation $x = (x_1, \ldots, x_n)$ can easily choose the allocation that maximizes the *welfare* $W(x) = \sum_{i=1}^n U_i(x_i) - C(X)$, where $X = \sum_{i=1}^n x_i$, since this is a simple convex optimization problem.

But what if users are autonomous and choose the quantities x_i to maximize their own objectives? The most natural way to proceed is *equilibrium analysis*, where we model each user as maximizing its own payoff function and consider equilibrium allocations — those from which no user can unilaterally change its quantity to increase its payoff. We can then study the welfare achieved by autonomous and self-optimizing users via the *price of anarchy (POA)* — the worst (i.e., smallest) ratio between the welfare of an equilibrium (the outcome of selfish behavior) and the maximum-possible welfare (the ideal for an altruistic designer). The POA is a standard measure of inefficiency in game-theoretic systems, with a value near 1 indicating that selfish behavior is essentially benign.

Defining user payoffs requires a fundamental modeling decision: how does the joint cost $C(X)$ of producing an allocation translate to negative incentives for

[⋆] The results in Section 3 of this work also appear in Chapter 5 of the first author's PhD thesis [4].

[⋆⋆] This work was done while the author was a PhD student at Stanford University, and supported in part by the ONR Young Investigator Award of the second author.

[⋆⋆⋆] Supported in part by NSF CAREER Award CCF-0448664, an ONR Young Investigator Award, an AFOSR MURI grant, and an Alfred P. Sloan Fellowship.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 546–557, 2009.

users? This choice is formalized by a *cost-sharing method* $\xi : \mathbf{R}^n_+ \to \mathbf{R}^n_+$, which distributes the joint cost to the users: $\sum_i \xi_i(x) = C(X)$ for every allocation x with $X = \sum_i x_i$. For example, a natural cost-sharing method is *average-cost pricing*, defined by $\xi_i^{\text{FIFO}}(x) = \frac{x_i}{X} \cdot C(X)$; we also call this the "FIFO method." In the queue example above, average-cost pricing naturally arises from the FIFO (first-in, first-out) queue service discipline with random packet arrivals. The other cost-sharing method that has been extensively studied in the present context is *serial cost-sharing*, which we define in Section 2 and also call the "Fair Share method," after Shenker [11]. Given a cost-sharing method ξ, we define the payoff of user i in allocation x as $P_i(x) = U_i(x_i) - \xi_i(x)$; equilibria and the POA are then defined as outlined above. (Thus, we assume utility functions are in the same units as the cost function.)

From a design perspective, an obvious question is: *which cost-sharing method yields the best welfare guarantee (i.e., POA closest to 1)?* This question, as stated, is not well defined: the best cost-sharing method depends on the players' utilities, and it is not reasonable to assume that these are known a priori to the designer. We therefore study the *worst-case POA* of cost-sharing methods, with the "worst" quantifier ranging over all possible utility function profiles $U_1, \ldots, U_n$ for a fixed number of users and a fixed cost function. Our research agenda is twofold: (1) For fundamental cost-sharing methods, precisely determine the worst-case POA in as many settings as possible; and (2) Identify the optimal cost-sharing method for a given environment — the one with the maximum-possible worst-case POA.

Our Results. Solving problems (1) and (2) in their full generality appears intractable, and our goal here is to provide precise answers for important special cases. Our first main result is for quadratic cost functions (of the form $C(X) = aX^2 + bX$). For a class of cost-sharing methods that strictly generalizes the FIFO and Fair Share methods, we give an exact formula for the worst-case POA of every method in the class for every number n of users. We give a single analysis that applies to all methods in the class. Our analysis identifies restricted linear structure in the equilibrium conditions for such methods and uses it to identify worst-case utility profiles. The precision of our analysis permits identification of the optimal method in this class for every number n of users. For example, our analysis shows that, in the limit as $n \to \infty$, the Fair Share method has the optimal worst-case POA among methods in the class.

General cost functions produce nonlinear equilibrium conditions and are much more difficult to analyze. For the Fair Share method and the case of $n = 2$ users, however, we show how to determine the worst-case POA with respect to general cost functions. This result is based on a novel and unexpected "reduction" to nonatomic selfish routing games.

Related Work. The work closest to ours is Moulin [5], who studies our exact model. Moulin [5] proved our first result for quadratic cost functions in the important special cases of the FIFO and Fair Share methods, using different (somewhat ad hoc) computations for each case. Here, we give a single analysis generalizing these two results of his, which also applies to a broader class of cost-sharing methods. In our opinion, it is surprising that the seemingly very different FIFO and Fair Share methods (see Section 2) can be simultaneously analyzed with a common proof. Our result for general cost functions and two players generalizes a different result in Moulin [5], who gave tight bounds in the two-player case for monomial cost functions (for both the FIFO and Fair Share methods). The connection to selfish routing games is new to this paper, and it allows us to analyze general (non-monomial) cost functions. Moulin [5] also gave a number of results for the incremental cost-sharing method, which generally charges users more than the total production cost (i.e., is not "budget-balanced") and therefore falls outside our purview. In subsequent work, Moulin [6] used budget-balanced cost-sharing methods with negative cost shares — subsidies, which are not permitted in this paper — to obtain much stronger positive results. For example, for every quadratic cost function and number of users, there is such a cost-sharing method that only induces games with POA equal to 1 [6].

A number of different but related models have been studied before. Closest to the present work is Johari and Tsitsiklis [2,3], who studied a variant of our model with inelastic supply — i.e., there is a fixed amount of the divisible good but no production cost — and identified the allocation mechanism with the best worst-case POA among those in a broad class. We mention also Shenker [11], who studied the Fair Share method in a queueing context but without any quantitative efficiency guarantees; Moulin and Shenker [7], who compared the FIFO and Fair Share methods from an axiomatic perspective; and Christodoulou et al. [1], who were the first to study (in a different model) how to design protocols to optimize the worst-case POA.

2 Fundamental Cost-Sharing Methods

The FIFO method was defined in Section 1. The Fair Share method is an alternative designed to insulate users that request smaller quantities from the large requests. For example, with two players and quantities $x_1 \leq x_2$, the method assigns a cost share of $C(2x_1)/2$ to the first player (its fair share, if we pretend that the second player shares its size), and the balance $C(x_1+x_2)-C(2x_1)/2$ to the second player. In general, all users split the cost that would ensue if all users were the same size as the smallest one; and the remaining cost is recursively allocated to the $n-1$ largest users.

Precisely, using $[n]$ to mean $\{1, 2, \ldots, n\}$: for a vector $x \in \mathbf{R}^n$, a permutation $\pi : [n] \rightarrow [n]$ is an *ordering* of x if the vector $z \in \mathbf{R}^n$ such that $z_{\pi(i)} = x_i$ satisfies $z_1 \leq z_2 \leq \cdots \leq z_n$. The vector z is the *ordered version* of x. There are multiple orderings of a vector x when it has some equal components, but all the orderings give rise to the same ordered version z.

Definition 1 (The Fair Share Method [7,11]). *For any cost function C, number of users n, and vector $x \in \mathbf{R}^n_+$, let π be an ordering of x with ordered* *version z. For $k \in [n]$, let $s_k = \sum_{\ell=1}^{k-1} z_\ell + (n-k+1)z_k$. Then the cost share of user $i \in [n]$ is*

$$\xi_i^{FS}(x) = \frac{C(s_{\pi(i)})}{n-\pi(i)+1} - \sum_{k=1}^{\pi(i)-1} \frac{C(s_k)}{(n-k+1)(n-k)}.$$

A simple way to interpolate between the FIFO and Fair Share methods is via the following θ-*combinations* for $\theta \in [0,1]$:

$$\xi_i(x) = \theta\xi_i^{\mathrm{FS}}(x) + (1-\theta)\xi_i^{\mathrm{FIFO}}(x). \qquad (1)$$

The FIFO and Fair Share methods correspond to the values $\theta = 0$ and $\theta = 1$, respectively. A θ-combination can be implemented in a system with one FIFO queue and one Fair Share queue. Each arriving packet is placed into the Fair Share queue with probability θ (and otherwise the FIFO queue). Departing packets are chosen from a queue with these same probabilities. We emphasize that while θ-combinations are defined as a linear combination of the FIFO and Fair Share methods, the equilibria and POA with respect to such methods are *not* linear in θ — indeed, even for a quadratic cost function, the worst-case POA is a fairly complex function of θ (Theorem 1).

3 Quadratic Cost Functions

This section considers quadratic cost functions. This assumption is restrictive, but we will be rewarded with an exact characterization of the worst-case price of anarchy for every θ-combination and number n of users. For simplicity, we assume that $C(X) = X^2$ throughout this section; scaling by a constant changes nothing, and adding a linear term (with a nonnegative coefficient) only improves the POA.

3.1 Equilibrium Properties

We now state some basic properties of equilibria with respect to a θ-combination and the cost function $C(X) = X^2$. The proofs of these preliminary results are not trivial, but they are not the main point of this paper and we refer the interested reader to [4, Chapter 5] for the technical details.

For a given θ-combination, with a quadratic cost function, there is a linear relationship between an allocation vector x and the corresponding marginal costs $\xi_i'(x_i; x_{-i})$. (Here the derivative is w.r.t. x_i and x_{-i} denotes the other users' quantities — the vector x with the ith component removed.) For example, Figure 1 demonstrates this linear relationship in the special case of the FIFO and Fair Share methods, when $n = 4$. The next proposition formalizes the relationship for all of the cost-sharing methods that we study.

$$B^0 = \begin{bmatrix} 2 & 1 & 1 & 1 \\ 1 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 1 & 1 & 2 \end{bmatrix} \qquad B^1 = \begin{bmatrix} 8 & 0 & 0 & 0 \\ 2 & 6 & 0 & 0 \\ 2 & 2 & 4 & 0 \\ 2 & 2 & 2 & 2 \end{bmatrix}$$

Fig. 1. The matrix B defined in Proposition 1 for the FIFO method (B^0) and Fair Share method (B^1), when $n = 4$. Assuming that the users have been sorted in nondecreasing order of x_i, the columns correspond to the quantities x_i, and the rows to the induced marginal costs $\xi_i'(x_i; x_{-i})$.

Proposition 1. *For the cost function $C(y) = y^2$, any number of users n, any user $i \in [n]$, and any $\theta \in [0, 1]$, let ξ_i denote the cost share of user i under the θ-combination. Define an $n \times n$ matrix B by*

$$B_{k\ell} = \begin{cases} 2(1 + \theta(n - k)), & \text{if } k = \ell; \\ 1 + \theta, & \text{if } k > \ell; \\ 1 - \theta, & \text{if } k < \ell \end{cases}$$

for any $k, \ell \in [n]$. For any vector $x \in \mathbf{R}^n_+$, let π be any ordering of x, and let z be the ordered version of x.

(a) The vector $p \in \mathbf{R}^n$ with $p_{\pi(i)} = \xi_i'(x_i; x_{-i})$ for all $i \in [n]$ is given by $p = Bz$.
(b) If $z_{k_1} < z_{k_2}$ then $p_{k_1} < p_{k_2}$; if $z_{k_1} \leq z_{k_2}$ then $p_{k_1} \leq p_{k_2}$.

Part (a) of Proposition 1 asserts that the matrix B correctly maps allocations to marginal costs. Part (b) asserts that marginal costs must be increasing in the quantities x_i. Proposition 1 will be useful in our POA analysis and also enables us to establish existence and uniqueness of equilibria.

Proposition 2. *For every θ-combination, quadratic cost function, and utility function profile, the induced game has a unique equilibrium allocation vector.*

Proposition 2 also holds for much more general convex cost functions. It can be proved by modifying Rosen's existence and uniqueness theorems for convex games [8]. (Modifications are needed because the Fair Share method is not continuously differentiable at allocation vectors with two equal components.)

3.2 Tight Bounds on the POA

We now show how to determine the worst-case price of anarchy of every θ-combination under a quadratic cost function, over all utility function profiles. We first note that linear functions — of the form $U_i(x_i) = a_i x_i$ for $a_i \geq 0$ — induce games with as large a POA as any other type of (nonnegative and concave) utility function.

Lemma 1 (Linearization Lemma [2,5]). *For every θ-combination, number n of users, and convex cost function, the worst-case POA (over all utility function profiles) is determined by linear utility function profiles.*

The proof of Lemma 1 simply shows that linearizing utility functions at the equilibrium point only worsens the POA, and then shifting the resulting nonnegative affine functions to be linear again only worsens the POA.

For the rest of this section, we assume that all utility functions are linear with $0 < a_1 \leq a_2 \leq \cdots \leq a_n$. For such a profile, an optimal solution allocates only to the nth user. A simple calculation shows that the optimal amount to give this user is $a_n/2$, leading to a welfare of $a_n(a_n/2) - (a_n/2)^2 = a_n^2/4$.

The next lemma studies the welfare of an equilibrium allocation x^* and is central to our analysis. It states that the requested quantities in x^* are in the same order as the a_i values, determines a remarkable formula for the welfare of the system under x^*, and develops a constraint that relates the x_i^* values to a_n.

Lemma 2. *For the cost function $C(y) = y^2$, any number of users n, any $\theta \in [0, 1]$, and any $a \in \mathbf{R}_+^n$ such that $0 < a_1 \leq a_2 \leq \cdots \leq a_n$, let x^* be the equilibrium allocation under the θ-combination.*

(a) The requested quantities in x^* are in the order $x_1^* \leq x_2^* \leq \cdots \leq x_n^*$.
(b) The welfare of the system under x^* is

$$W(x^*) = \sum_{i=1}^{n} (2\theta(n-i)+1)\,(x_i^*)^2\,. \tag{2}$$

(c) The components of x^* satisfy the equation

$$(1+\theta)\sum_{i=1}^{n-1} x_i^* + 2x_n^* = a_n. \tag{3}$$

Proof. Simple computations show that, for every i, the function $\xi_i(y; x_{-i})$ is convex and differentiable in y for every fixed x_{-i}. It follows that an allocation vector x is an equilibrium, with each user i choosing an optimal quantity given x_{-i}, if and only if

$$\begin{aligned} x_i^* > 0 &\Rightarrow a_i = \xi_i'(x_i^*; x_{-i}^*); \\ x_i^* = 0 &\Rightarrow a_i \leq \xi_i'(0; x_{-i}^*). \end{aligned} \tag{4}$$

To prove (a), let x^* be an equilibrium and suppose for contradiction that there are two users i_1 and i_2 such that $i_1 < i_2$ and $x_{i_1}^* > x_{i_2}^*$. Because $x_{i_2}^* \geq 0$, we have $x_{i_1}^* > 0$, and so the equilibrium conditions in (4) imply that $a_{i_1} = \xi_{i_1}'(x_{i_1}^*; x_{-i_1}^*)$ and $a_{i_2} \leq \xi_{i_2}'(x_{i_2}^*; x_{-i_2}^*)$. Since $a_{i_1} \leq a_{i_2}$, $\xi_{i_1}'(x_{i_1}^*; x_{-i_1}^*) \leq \xi_{i_2}'(x_{i_2}^*; x_{-i_2}^*)$. On the other hand, since $x_{i_1}^* > x_{i_2}^*$, Proposition 1 implies that $\xi_{i_1}'(x_{i_1}^*; x_{-i_1}^*) > \xi_{i_2}'(x_{i_2}^*; x_{-i_2}^*)$, contradicting this inequality.

Given that $x_1^* \leq x_2^* \leq \cdots \leq x_n^*$, we can apply Proposition 1 with the ordering π of x^* being the identity permutation to rewrite the equilibrium conditions in (4) as

$$\begin{aligned} x_i^* > 0 \Rightarrow a_i &= \sum_{j=1}^{n} B_{ij} x_j^*; \\ x_i^* = 0 \Rightarrow a_i &\le \sum_{j=1}^{n} B_{ij} x_j^*, \end{aligned} \tag{5}$$

where B is the $n \times n$ matrix defined in Proposition 1. By definition, the welfare of x^* is

$$W(x^*) = \sum_{i=1}^{n} a_i x_i^* - \left(\sum_{i=1}^{n} x_i^* \right)^2 . \tag{6}$$

The equilibrium conditions in (5) imply that total utility at equilibrium can be written as a quadratic form:

$$\sum_{i=1}^{n} a_i x_i^* = \sum_{i=1}^{n} x_i^* \sum_{j=1}^{n} B_{ij} x_j^* = (x^*)^T B x^* = (x^*)^T \left(\frac{1}{2} \left(B + B^T \right) \right) x^* .$$

Since a quadratic cost function can be similarly expressed as

$$C(x) = \left(\sum_i x_i \right)^2 = x^T E x,$$

where E is the all-ones $n \times n$ matrix, *equilibrium welfare can be expressed as a quadratic form*:

$$\begin{aligned} W(x^*) &= (x^*)^T \left(\frac{1}{2} \left(B + B^T \right) \right) x^* - (x^*)^T E x^* \\ &= (x^*)^T \left(\frac{1}{2} \left(B + B^T \right) - E \right) x^* . \end{aligned} \tag{7}$$

Let D denote the symmetric matrix $\frac{1}{2}(B + B^T) - E$. By the definition of B, the diagonal entries of D are $D_{ii} = B_{ii} - 1 = 2\theta(n - i) + 1$ for all $i \in [n]$. For any $i, j \in [n]$ such that $i \neq j$, we have $D_{ij} = D_{ji} = (1/2)(1 + \theta + 1 - \theta) - 1 = 0$. *Thus D is a diagonal matrix*, and the equation involving the quadratic form $(x^*)^T D x^*$ in (7) simplifies to

$$W(x^*) = \sum_{i=1}^{n} D_{ii} \left(x_i^* \right)^2 ,$$

which yields the expression in (2) upon substitution of the D_{ii} values.

Finally, since $a_n > 0$, the equilibrium condition in (5) implies that $x_n^* > 0$ with

$$a_n = \sum_{i=1}^{n} B_{ni} x_i^* .$$

By substituting the B_{ni} values, we obtain the equation in (3). □

Scaling a vector of coefficients a by λ increases both the optimal and equilibrium welfares by a factor of λ^2 (for the latter, this follows from the linear equilibrium conditions and Lemma 2(b)). Since the POA is the ratio of these, and since the optimal welfare depends only on a_n, we can restrict our search for the worst-case utility function profile to the set $\mathcal{A} = \{a \in \mathbf{R}_+^n \mid 0 < a_1 \leq a_2 \leq \cdots \leq a_n = 1\}$ and focus on minimizing the equilibrium welfare W_a over $a \in \mathcal{A}$. Our second key lemma computes this minimum precisely.

Lemma 3. *Fix the cost function $C(y) = y^2$, any number of users n, and any $\theta \in [0, 1]$. Then $\inf_{a \in \mathcal{A}} W_a = 1/4\Gamma_\theta(n)$, where W_a is the welfare of the (unique) equilibrium for the utility profile a, and*

$$\Gamma_\theta(n) = 1 + \frac{(1+\theta)^2}{4} \sum_{i=1}^{n-1} \frac{1}{2\theta i + 1}. \tag{8}$$

Proof. By Lemma 2, a lower bound on the minimum-possible equilibrium welfare is provided by the value of the convex program

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} (2\theta(n-i)+1) x_i^2 \\ \text{subject to} \quad & (1+\theta) \sum_{i=1}^{n-1} x_i + 2x_n = 1. \end{aligned} \tag{9}$$

We introduce a Lagrange multiplier λ for the constraint $1 - (1+\theta)\sum_{i=1}^{n-1} x_i - 2x_n = 0$. Then the Karush-Kuhn-Tucker (KKT) optimality conditions for the program in (9) are

$$2(2\theta(n-i)+1)x_i - \lambda(1+\theta) = 0, \quad \forall i \in [n-1];$$
$$2x_n - 2\lambda = 0.$$

Solving the KKT conditions for the x_i values yields

$$\begin{aligned} x_i &= \left(\frac{1+\theta}{2\theta(n-i)+1}\right)\frac{\lambda}{2}, \quad \forall i \in [n-1]; \\ x_n &= \lambda. \end{aligned}$$

Substituting these values into the equality constraint in (9), we obtain

$$\begin{aligned} 1 &= \lambda \left(\frac{(1+\theta)^2}{2} \sum_{i=1}^{n-1} \frac{1}{2\theta(n-i)+1} + 2 \right) \\ &= 2\lambda \left(\frac{(1+\theta)^2}{4} \sum_{i=1}^{n-1} \frac{1}{2\theta i + 1} + 1 \right) \\ &= 2\lambda \Gamma_\theta(n), \end{aligned}$$

and thus $\lambda = 1/2\Gamma_\theta(n)$.

The value of the objective function in (9) for this vector x is

$$\begin{aligned}\sum_{i=1}^{n-1}(2\theta(n-i)+1)\left(\frac{1+\theta}{2\theta(n-i)+1}\right)^2\left(\frac{\lambda}{2}\right)^2+\lambda^2 &= \lambda^2\left(\frac{(1+\theta)^2}{4}\sum_{i=1}^{n-1}\frac{1}{2\theta(n-i)+1}+1\right)\\ &= \lambda^2\Gamma_\theta(n)\\ &= \frac{1}{4\Gamma_\theta(n)}.\end{aligned}$$

This quantity lower bounds the minimum-possible equilibrium welfare (for vectors $a \in \mathcal{A}$).

To obtain a matching upper bound, consider the vector $x \in \mathbf{R}_+^n$ obtained by solving the KKT conditions and imposing the equality constraint in (9). The components of x are

$$\begin{aligned} x_i &= \left(\frac{1+\theta}{2\theta(n-i)+1}\right)\frac{1}{4\Gamma_\theta(n)}, \quad \forall i \in [n-1];\\ x_n &= \frac{1}{2\Gamma_\theta(n)},\end{aligned}$$

and so $0 \le x_1 \le x_2 \le \cdots \le x_n$. Define a vector a as $a = Bx$, where B is the matrix defined in the statement of Proposition 1 for the θ-combination. Proposition 1 implies that $a_1 \le a_2 \le \cdots \le a_n$. Moreover, because x satisfies the equality constraint in (9), we have

$$a_n = \sum_{i=1}^{n-1} B_{ni}x_i + B_{nn}x_n = (1+\theta)\sum_{i=1}^{n-1} x_i + 2x_n = 1.$$

Finally, the vector x satisfies the equilibrium conditions in (5), so it is the equilibrium of the game with utility functions defined by a. Since the welfare of x is $W_a = 1/4\Gamma_\theta(n)$, the proof is complete. □

Recalling that the optimal welfare is $1/4$ for every vector $a \in \mathcal{A}$, we have our main result for quadratic cost functions.

Theorem 1. *For the cost function $C(y) = y^2$, any number of users n, and any $\theta \in [0,1]$, the price of anarchy of the θ-combination is $1/\Gamma_\theta(n)$.*

The formula in Theorem 1 for the two special cases $\theta = 0$ and $\theta = 1$ was established in Moulin [5] using two different proofs. Obviously, the formula in (8) can be used to identify the optimal θ-combination for every number n of users. When $n = 2$, the FIFO method is the best θ-combination (with worst-case POA 4/5) while the Fair Share method is the worst (with worst-case POA 3/4). When $n = 3$, the Fair Share method remains the worst (with worst-case POA 15/23). The FIFO method is slightly better, with worst-case POA 2/3. Taking $\theta \approx .262648$ yields a superior θ-combination, with worst-case POA $\approx .687$. For all $n \ge 4$, the Fair Share method outperforms the FIFO method but other θ-combinations are still better (see Table 1). In the limit as $n \to \infty$, for every

Table 1. Illustration of Theorem 1. Comparison, for different n, of the exact worst-case POA of the FIFO method, the Fair Share method, and the optimal θ-combination.

n	POA of FIFO	POA of Fair Share	Optimal θ	Optimal POA
2	.8	.75	0	.8
3	$\approx .667$	$\approx .652$	$\approx .262648$	$\approx .687$
4	$\approx .571$	$\approx .595$	$\approx .375361$	$\approx .623$
5	.5	$\approx .559$	$\approx .442921$	$\approx .581$
10	$\approx .308$	$\approx .469$	$\approx .588111$	$\approx .481$
20	$\approx .174$	$\approx .403$	$\approx .677465$	$\approx .410$
40	$\approx .093$	$\approx .354$	$\approx .737$	$\approx .358$

$\theta \in (0, 1]$, $\Gamma_\theta(n)$ scales as $(1+\theta)^2 \ln n/(8\theta)$. (For $\theta = 0$, the FIFO method scales as $4/(n+3)$.) Since $\theta/(1+\theta)^2$ is increasing in the interval $\theta \in (0, 1]$, the Fair Share method has the best asymptotic worst-case POA, which scales as $2/(\ln n)$ for large n.

4 General Cost Functions

This section considers general (non-quadratic) cost functions. Analyzing the case of many players appears intractable, so we settle for a solution to two-player games induced by the Fair Share method. We begin with a simple but useful lemma, which holds even with many users.

Lemma 4. *If all users have linear utility functions and the cost function is strictly convex, then the total quantity allocated in the Fair Share equilibrium equals that in the optimal allocation.*

Proof. (Sketch.) Suppose user n has the largest utility function coefficient a_n. One optimal solution allocates only to user n, and the optimal amount to allocate is the unique point X at which $a_n = C'(X)$.

Consider an equilibrium under Fair Share. Analogous to condition (5) in Lemma 2 and the bottom row of the matrix B^1 in Figure 1, at equilibrium we must have $a_n = \xi'_n(x^*_n; x^*_{-n}) = C'(X)$. Thus the quantity allocated at x^* (across all users) equals that in the optimal solution (to user n only). □

Our approach is to show an explicit connection, interesting in its own right, between games with two users with linear utility functions and nonatomic selfish routing games (e.g. [10]). Recall *Pigou's example*, a basic selfish routing network: X units of traffic, comprising a large number of infinitesimal autonomous users, choose between two parallel links connecting one vertex to another. One link has some per-unit cost function $c(x_1)$, and the other has constant per-unit cost $c(X)$. The first link is a dominant strategy, so in the only equilibrium all traffic takes it and the aggregate cost is $X \cdot c(X)$. An optimal outcome, by definition, splits the traffic x_1 and $x_2 = X - x_1$ between the two links to minimize $x_1 \cdot c(x_1) + x_2 \cdot c(X)$. Much is known about the ratio between the equilibrium and optimal costs in

Pigou's example (and much more general selfish routing networks), as a function of c. This ratio is called the *Pigou bound for c* and is denoted $\alpha(c)$. For example, the Pigou bound for all affine functions is at most $4/3$, with equality achieved when $c(x) = ax$ for some $a > 0$; and for per-unit cost functions c that are polynomials with nonnegative coefficients and degree at most p, the largest Pigou bound grows like $\approx p/\ln p$ [9].

The connection between Pigou's example and the queueing games studied in this paper is most vivid for the total user utility — so for our penultimate result, we ignore the cost term in the welfare objective function.

Theorem 2. *For every differentiable convex cost function C, the worst-case fraction of the optimal total utility achieved by the Fair Share equilibrium allocation with two players is exactly*

$$\frac{1}{2}\left(1 + \frac{1}{\alpha(C')}\right),$$

where $\alpha(C')$ is the Pigou bound for C'.

Proof. (Sketch.) We prove the theorem constructively, by exhibiting a worst-possible example for the Fair Share equilibrium allocation. Fix a choice of $X \geq 0$; we later optimize adversarially over X. Give the second user the utility function $U_2(x_2) = C'(X) \cdot x_2$. For any $x_1 \in [0, X/2]$, choosing the coefficient $a_1 = C'(2x_1) \leq C'(X)$ for the first user's utility function ensures that $x_1^* = x_1$ at the Fair Share equilibrium x^*. For a given choice of X and $x_1 \in [0, X/2]$, the total user utility obtained by Fair Share is then $x_1 \cdot C'(2x_1) + (X - x_1) \cdot C'(X)$. By a change of variable, this payoff is minimized at $y^*/2$, where y^* is the optimal amount of traffic to route on the non-constant link of Pigou's example when there are X units of traffic and the non-constant per-unit cost function c is C'. The resulting total user utility is

$$\begin{aligned}\frac{y^*}{2}C'(y^*) + \left(X - \frac{y^*}{2}\right) \cdot C'(X) &= \frac{1}{2}\left(y^* C'(y^*) + (X - y^*) \cdot C'(X)\right) + \frac{1}{2}\left(X \cdot C'(X)\right) \\ &\geq X \cdot C'(X)\left(\frac{1}{2} + \frac{1}{2\alpha(C')}\right),\end{aligned}$$

where the inequality follows from the definition of the Pigou bound $\alpha(C')$ for C'. The parameter X can be chosen so that the inequality is arbitrarily close to an equality. The total user utility obtained in the optimal solution is $X \cdot C'(X)$, and the Fair Share equilibrium allocation obtains only a $\frac{1}{2}(1 + 1/\alpha(C'))$ fraction of this. Reversing the steps in the argument above shows that no worse example is possible. □

To extend Theorem 2 to a bound on the POA for the welfare objective, we need to re-introduce the cost terms (for both the optimal and equilibrium allocations). This can be approached in a number of ways. Since we can assume utility functions are linear (Lemma 1), Lemma 4 shows that both allocations will suffer the same cost. A crude way to proceed is to define

$$\gamma(c) = \sup_{X \geq 0} \frac{\int_0^X c(x)dx}{X \cdot c(X)}; \tag{10}$$

for example, if $c(x) = x^d$, then $\gamma(c) = 1/(d+1)$. Theorem 2 then yields the following corollary for the POA.

Corollary 1. *For every differentiable convex cost function C, the worst-case POA of Fair Share with two players is at least*

$$\frac{1}{1-\gamma(C')} \cdot \left(\frac{1}{2} \left(1 + \frac{1}{\alpha(C')} \right) - \gamma(C') \right),$$

where $\alpha(C')$ is the Pigou bound for C' and $\gamma(C')$ is defined as in (10).

For example, for the marginal cost function $C'(x) = x^d$, plugging in the known upper bound on the Pigou value [9] together with $\gamma(C') = 1/(d+1)$ immediately gives a lower bound of $1 - \frac{1}{2}(d+1)^{-1/d}$, on the worst-case POA, recovering a result of Moulin [5]. Other natural types of cost functions can be treated in a similar way.

References

1. Christodoulou, G., Koutsoupias, E., Nanavati, A.: Coordination mechanisms. In: D´Aaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 345–357. Springer, Heidelberg (2004)
2. Johari, R., Tsitsiklis, J.N.: Efficiency loss in a network resource allocation game. Mathematics of Operation Research 29(3), 407–435 (2004)
3. Johari, R., Tsitsiklis, J.N.: Efficiency of scalar-parameterized mechanisms. Operations Research (to appear, 2009)
4. Mosk-Aoyama, D.: Convergence to and Quality of Equilibria in Distributed Systems. PhD thesis, Stanford University (2008)
5. Moulin, H.: The price of anarchy of serial, average and incremental cost sharing. Economic Theory 36(3), 379–405 (2008)
6. Moulin, H.: An efficient and almost budget balanced cost sharing method. Games and Economic Behavior (in press, 2009)
7. Moulin, H., Shenker, S.J.: Serial cost sharing. Econometrica 60(5), 1009–1037 (1992)
8. Rosen, J.B.: Existence and uniqueness of equilibrium points for concave n-person games. Econometrica 33(3), 520–534 (1965)
9. Roughgarden, T.: The price of anarchy is independent of the network topology. Journal of Computer and System Sciences 67(2), 341–364 (2003)
10. Roughgarden, T.: Selfish Routing and the Price of Anarchy. MIT Press, Cambridge (2005)
11. Shenker, S.J.: Making greed work in networks: A game-theoretic analysis of switch service disciplines. IEEE/ACM Transactions on Networking 3(6), 819–831 (1995)

On Observing Dynamic Prioritised Actions in SOC*

Rosario Pugliese[1], Francesco Tiezzi[1], and Nobuko Yoshida[2]

[1] Dipartimento di Sistemi e Informatica, Università degli Studi di Firenze
[2] Department of Computing, Imperial College London

Abstract. We study the impact on observational semantics for SOC of priority mechanisms which combine dynamic priority with local pre-emption. We define manageable notions of strong and weak labelled bisimilarities for COWS, a process calculus for SOC, and provide alternative characterisations in terms of open barbed bisimilarities. These semantics show that COWS's priority mechanisms partially recover the capability to observe receive actions (that could not be observed in a purely asynchronous setting) and that high priority primitives for termination impose specific conditions on the bisimilarities.

1 Introduction

Service-oriented computing (SOC) is an emergent paradigm for distributed computing that aims to build networks of interoperable applications through the use of platform-independent, reusable software components called *services*. Service definitions are used as templates for creating service instances that supply application functionalities to either end-user applications or other instances. Being inherently loosely coupled, SOC systems do not provide intrinsic mechanisms to identify service instances for delivering messages and to link together actions executed as part of the same client-service long-running interaction. Therefore, emerging standards like WS-BPEL and WS-CDL advocate the use of *correlation data* within exchanged messages that the interacting partners can retrieve by means of a *pattern matching* mechanism.

Recently, many process calculi have been expressly designed to model SOC scenarios, so that service definitions and service instances are represented as reactive processes running concurrently. In this setting, priority mechanisms which allow some actions to take precedence over others can be very fruitful. E.g., when a message arrives, the problem arises of rightly handling race conditions among those service instances and the corresponding service definition which are able to receive the message. This can be modelled by exploiting a parallel composition operator that gives precedence to actions with greater priority. Receive activities are then assigned priority values which depend on the messages available so that, in presence of concurrent matching receives, only a receive using a more defined pattern (i.e. having greater priority) can proceed. This way, service instances take precedence over the corresponding service definition when both can process the same message, thus preventing creation of wrong new instances.

Notably, receives would have *dynamically* assigned priority values since these values depend on the matching ability of their argument pattern. Indeed, while computation

* This work has been supported by the EU project SENSORIA, IST-2 005-016004, and EPSRC GR/T03215, GR/T03208 and EP/F003757.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 558–570, 2009.

proceeds, some of the variables used in the argument pattern of a receive can be assigned values, because of execution of syntactically preceding receives or of concurrent threads sharing these variables. This restricts the set of messages matching the pattern while increases the priority of the receive. Furthermore, pre-emption is *local* since receives having a more defined pattern have a higher execution priority with respect to only the other receives matching the same message.

There are other situations where local pre-emption is needed. For example, when a fault arises in a scope, (some of) the remaining activities of the enclosing scope should be terminated before starting the execution of the relative fault handler. This can be modelled by exploiting the same parallel operator as before together with actions for forcing immediate termination of concurrent activities which take the greatest priority. The same mechanism can also be used for exception and compensation handling.

However, apart from COWS [10,13], priority mechanisms that combine dynamic priority with local pre-emption have not been studied yet in the literature [4]. COWS (*Calculus for Orchestration of Web Services*) is an extension of asynchronous π-calculus [7,1] equipped with the priority mechanisms sketched above and with other distinctive features of SOC systems inspired by WS-BPEL, such as shared variables among parallel threads, and communication based on correlation and pattern matching.

This paper studies the impact of COWS's priority mechanisms on observational semantics for SOC. We first define strong and weak labelled bisimilarities for a fragment of COWS without primitives for termination, and prove that they are *sound* and *complete* with respect to contextual barbed ones (Section 3). Due to the locality of received endpoints, the labelled bisimilarities involves a family of relations indexed by sets of names, similar to the quasi-open bisimilarity for π-calculus [12]. The obtained semantics inhabits between asynchrony and synchrony because, with respect to a purely asynchronous setting, the priority mechanism permits partially recovering the capability to observe receive actions. We then extend our investigation and results to COWS (Section 4). We get that the primitives with greatest priority causing termination require specific conditions on the labelled bisimilarities for these to be congruences. Hence, the resulting observations are more fine-grained than the previous ones. Our semantic theories are usable to check interchangeability of services and conformance against service specifications as demonstrated through a practical example in Sections 2 and 4.

2 A 'Morra game' Scenario

We start providing some insights into COWS's main features in a step-by-step fashion by means of an example service described at two different levels of abstraction.

The service allows its clients to play the well-known game Morra, where two players, named "odds" and "evens", throw out a single hand, each showing zero to five fingers. If the sum of fingers shown by both players is an even number then the "evens" player wins; otherwise the "odds" player is the winner. The service collects the two throws (i.e. two integers), calculates the winner and sends the result back to the two players. A high-level specification of the service in COWS is:

$$
\begin{aligned}
* \, [x_{id}, x_p, x_{num}, y_p, y_{num}] \, (\, & \mathit{odds} \bullet \mathit{throw}?\langle x_{id}, x_p, x_{num} \rangle \mid \mathit{evens} \bullet \mathit{throw}?\langle x_{id}, y_p, y_{num} \rangle \\
& \mid x_p \bullet \mathit{res}!\langle x_{id}, \mathit{win}(x_{num}, y_{num}, 1) \rangle \mid y_p \bullet \mathit{res}!\langle x_{id}, \mathit{win}(x_{num}, y_{num}, 0) \rangle \,)
\end{aligned} \tag{1}
$$

The replication operator $*\,\_$, that spawns in parallel as many copies of its argument term as necessary, supports creation of multiple instances to serve several matches simultaneously. The delimitation operator $[\_]$ declares the scope of variables x_{id}, x_p, y_p, x_{num} and y_{num}. Two distinct endpoints, i.e. pairs $odds \bullet throw$ and $evens \bullet throw$, are used by the service to receive throws from the players. When sending their throws, the players are required to provide a match identifier, stored in x_{id}, and the partner names, stored in x_p and y_p, that they will use to receive the result. To avoid interferences between matches played simultaneously, match-ids could be made unique by using delimitation.

Players throws arrive randomly, thus any interaction with the service starts with one of the two *receive* activities $odds \bullet throw?\langle x_{id}, x_p, x_{num}\rangle$ or $evens \bullet throw?\langle x_{id}, y_p, y_{num}\rangle$, that are *correlated* by means of the *shared* variable x_{id}, and terminates with the two *invoke* activities $x_p \bullet res!\langle x_{id,} win(x_{num}, y_{num}, 1)\rangle$ and $y_p \bullet res!\langle x_{id,} win(x_{num}, y_{num}, 0)\rangle$, used to reply with the result. We assume that $win(x, y, z)$ is a total function which, if x and y are integers between 0 and 5, returns w in case $(x + y)\, mod\ 2$ is equal to z, and l if $(x + y)\, mod\ 2$ is different from z; otherwise, err is returned.

A communication takes place when the arguments of a receive and of a concurrent invoke along the same endpoint match and causes replacement of the variables arguments of the receive with the corresponding values arguments of the invoke (within the scope of variables declarations). When operation *throw* is invoked, if a service instance with the same match-id already exists, then the invocation is received by the instance, otherwise a new instance is activated. This is done through the *dynamic prioritised mechanism* of COWS, i.e. assigning the receives by instances (having a more defined pattern) a greater priority than the receives by a service definition.

Thus, for example, after an interaction with the following client

$$[z]\,(\,evens \bullet throw!\langle first, cbB, 1\rangle \;\mid\; cbB \bullet res?\langle first, z\rangle\,.\,\langle\text{rest of client B}\rangle\,)$$

service definition (1) runs in parallel with the instance identified by the match-id *first*

$$\begin{aligned}[x_p, x_{num}]\ (\,& odds \bullet throw?\langle first, x_p, x_{num}\rangle \\ & \mid\ x_p \bullet res!\langle first, win(x_{num}, 1, 1)\rangle \ \mid\ cbB \bullet res!\langle first, win(x_{num}, 1, 0)\rangle\,)\end{aligned}$$

Now, if another client performs the invocation $odds \bullet throw!\langle first, cbA, 2\rangle$, it will be processed by the already existing instance because, w.r.t. this invocation, the receive $odds \bullet throw?\langle first, x_p, x_{num}\rangle$ has greater priority than the receive $odds \bullet throw?\langle x_{id}, x_p, x_{num}\rangle$ occurring in (1) (that has a less defined argument pattern).

For a lower level implementation, we wish to maximise the abilities of different services, while preserving the observable behaviour of the whole service w.r.t. the high-level specification. The main service is now composed of three entities as follows:

$$[req2f, req5f, resp2f, resp5f]\,(\,*\,M \;\mid\; *\,2F \;\mid\; *\,5F\,) \tag{2}$$

The delimitation operator is used here to declare that *req2f*, *req5f*, *resp2f* and *resp5f* are private operation names known to the three components *M*, *2F* and *5F*, and only to them. The three subservices are defined as follows:

$$\begin{aligned}M \triangleq\ & [x_{id}, x_p, x_{num}, y_p, y_{num}] \\ & (\,odds \bullet throw?\langle x_{id}, x_p, x_{num}\rangle \ \mid\ evens \bullet throw?\langle x_{id}, y_p, y_{num}\rangle \\ & \ \mid\ [k]\,(\,m \bullet req2f!\langle x_{id}, x_{num}, y_{num}\rangle \ \mid\ m \bullet req5f!\langle x_{id}, x_{num}, y_{num}\rangle \\ & \qquad \mid\ [x_o, x_e]\, m \bullet resp2f?\langle x_{id}, x_o, x_e\rangle.\,(\,\mathbf{kill}(k) \mid \{\!|x_p \bullet res!\langle x_{id}, x_o\rangle \mid y_p \bullet res!\langle x_{id}, x_e\rangle|\!\}\,) \\ & \qquad \mid\ [x_o, x_e]\, m \bullet resp5f?\langle x_{id}, x_o, x_e\rangle.\,(\,\mathbf{kill}(k) \mid \{\!|x_p \bullet res!\langle x_{id}, x_o\rangle \mid y_p \bullet res!\langle x_{id}, x_e\rangle|\!\}\,)\,)\end{aligned}$$

Table 1. μCOWS syntax

$s ::= u \bullet u'!\bar{\epsilon} \;\mid\; g \;\mid\; s \mid s \;\mid\; [u]\, s \;\mid\; *s$	(invoke, guard, parallel, delimitation, replication)
$g ::= \mathbf{0} \;\mid\; p \bullet o?\bar{w}.s \;\mid\; g + g$	(nil, receive, choice)

$$\begin{array}{l} \mathit{2F} \triangleq [x] \\ \quad (\, m \bullet \mathit{req2f}?\langle x, 1, 1\rangle.\, m \bullet \mathit{resp2f}!\langle x, l, w\rangle \\ \quad\; + m \bullet \mathit{req2f}?\langle x, 1, 2\rangle.\, m \bullet \mathit{resp2f}!\langle x, w, l\rangle \\ \quad\; + m \bullet \mathit{req2f}?\langle x, 2, 1\rangle.\, m \bullet \mathit{resp2f}!\langle x, w, l\rangle \\ \quad\; + m \bullet \mathit{req2f}?\langle x, 2, 2\rangle.\, m \bullet \mathit{resp2f}!\langle x, l, w\rangle\,) \end{array}$$

$$\begin{array}{l} \mathit{5F} \triangleq [x, y, z] \\ \quad (\, m \bullet \mathit{req5f}?\langle x, y, z\rangle.\, m \bullet \mathit{resp5f}!\langle x, \mathit{err}, \mathit{err}\rangle \\ \quad\; + m \bullet \mathit{req5f}?\langle x, 0, 0\rangle.\, m \bullet \mathit{resp5f}!\langle x, l, w\rangle \\ \quad\; + m \bullet \mathit{req5f}?\langle x, 0, 1\rangle.\, m \bullet \mathit{resp5f}!\langle x, w, l\rangle \\ \quad\; + \ldots + m \bullet \mathit{req5f}?\langle x, 5, 5\rangle.\, m \bullet \mathit{resp5f}!\langle x, l, w\rangle) \end{array}$$

Service M is publicly invocable and can interact with players as well as with the 'internal' services *2F* and *5F*. These latter two services, instead, can only be invoked by M and have the task of calculating the winner of a match. In particular, *2F* performs a quick computation of simple matches where both players hold out either one or two fingers, while *5F* performs a slower computation of standard 5-fingers matches (that exactly corresponds to the computation modelled by the function $\mathit{win}(\_)$). After the two initial receives, for e.g. performance and fault tolerance purposes, M invokes services *2F* and *5F* concurrently. Communication between M and the other two subservices exploits the match identifier (stored in x) as a correlation datum. When one of *2F* and *5F* replies, M immediately stops the other computation. This is done by executing the *kill* activity $\mathbf{kill}(k)$, that forces termination of all unprotected parallel terms inside the enclosing $[k]$, which stops the killing effect. Kill activities take the greatest priority w.r.t. the other parallel activities included within the enclosing scope. However, critical activities can be protected from the effect of a forced termination by using the *protection* operator $\{\!|\_|\!\}$; this is indeed the case of the response $x_p \bullet \mathit{res}!\langle x_{id}, x_o\rangle$ in our example. Finally, M forwards the responses to the players and terminates.

Services *2F* and *5F* use the *choice* operator $\_ + \_$ to offer alternative behaviours: one of them can be selected by executing an invoke matching the receive leading the behaviour. If the throws are not integers between 0 and 5, *2F* does not reply, while *5F* returns the string *err*. Indeed, the receive $m \bullet \mathit{req5f}?\langle x, y, z\rangle$ is assigned less priority than the other receive activities, i.e. it is only executed when none of the other receives matches the two throws, thus avoiding to return *err* in case of admissible throws.

3 μCOWS : The Protection- and Kill-Free Fragment of COWS

μCOWS's syntax is presented in Table 1. We use two countable and disjoint sets: the set of *values* (ranged over by v, v', ...) and the set of 'write once' *variables* (ranged over by $x, y, \ldots$). The set of values includes the set of *names* (ranged over by n, m, p, o, ...) mainly used to represent partners and operations. We also use a set of *expressions* (ranged over by ϵ) which contains, at least, values and variables. Partner names and operation names can be combined to designate *endpoints*, written $p \bullet o$, and can be communicated, but dynamically received names can only be used for service invocation.

We use w to range over values and variables and u to range over names and variables. $\bar{\cdot}$ stands for tuples, e.g. $\bar{x}$ means $\langle x_1, \ldots, x_n\rangle$ (with $n \geq 0$) where the x_is are pairwise distinct. We write $a, \bar{b}$ to denote the tuple obtained by concatenating the element

Table 2. Matching rules

$$\mathcal{M}(x, v) = \{x \mapsto v\} \qquad \mathcal{M}(v, v) = \emptyset \qquad \mathcal{M}(\langle\rangle, \langle\rangle) = \emptyset \qquad \frac{\mathcal{M}(w_1, v_1) = \sigma_1 \qquad \mathcal{M}(\bar{w}_2, \bar{v}_2) = \sigma_2}{\mathcal{M}((w_1, \bar{w}_2), (v_1, \bar{v}_2)) = \sigma_1 \uplus \sigma_2}$$

a to the tuple $\bar{b}$. All notations shall extend to tuples component-wise. n ranges over communication endpoints that do not contain variables (e.g. $p \bullet o$), while u ranges over communication endpoints that may contain variables (e.g. $u \bullet u'$). When convenient, we shall regard a tuple or an endpoint simply as a set, writing e.g. $x \in \bar{y}$ to mean that x is an element of $\bar{y}$. We will omit trailing occurrences of $\mathbf{0}$ and write $[u_1, \ldots, u_n]\, s$ in place of $[u_1] \ldots [u_n]\, s$. We will write $I \triangleq s$ to assign a name I to the term s.

We assume that monadic operators bind more tightly than parallel composition, and prefixing more tightly than choice. The only *binding* construct is delimitation: $[u]\, s$ binds u in the scope s. The occurrence of a name/variable is *free* if it is not under the scope of a delimitation for it. $\mathrm{fu}(t)$ denotes the set of free names/variables in t.

The operational semantics of μCOWS is defined only for *closed* terms, i.e. terms without free variables, and is given in terms of a structural congruence and of a labelled transition relation. The *structural congruence*, written $\equiv$, is defined as the least congruence relation induced by a given set of equational laws. We explicitly show here the laws for replication and delimitation

$$*\,\mathbf{0} \equiv \mathbf{0} \quad *\,s \equiv s \,|\, *\,s \quad [u]\,\mathbf{0} \equiv \mathbf{0} \quad [u_1]\,[u_2]\,s \equiv [u_2]\,[u_1]\,s \quad s_1 \mid [u]\,s_2 \equiv [u]\,(s_1 \mid s_2) \text{ if } u \notin \mathrm{fu}(s_1)$$

while omit the (standard) laws for the other operators stating that parallel composition and guarded choice are commutative, associative and have $\mathbf{0}$ as identity element. All the presented laws are straightforward. In particular, the last law permits to extend the scope of names (as in π-calculus) and variables, thus enabling possible communication.

To define the labelled transition relation, we use three auxiliary functions. Firstly, we use the function $[\![\_]\!]$ for evaluating *closed* expressions (i.e. expressions without variables): it takes a closed expression and returns a value. It is not explicitly defined since the exact syntax of expressions is deliberately not specified. Secondly, we use the partial function $\mathcal{M}(\_, \_)$ for performing *pattern-matching* on semi-structured data and, thus, determining if a receive and an invoke over the same endpoint can synchronise. The (straightforward) rules defining $\mathcal{M}(\_, \_)$ are shown in Table 2. When tuples $\bar{w}$ and $\bar{v}$ match, $\mathcal{M}(\bar{w}, \bar{v})$ returns a substitution for the variables in $\bar{w}$; otherwise, it is undefined. *Substitutions* (ranged over by σ) are functions mapping variables to values and are written as collections of pairs of the form $x \mapsto v$. Application of σ to s, written $s \cdot \sigma$, has the effect of replacing every free occurrence of x in s with v, for each $x \mapsto v \in \sigma$, by possibly using α-conversion for avoiding v to be captured by name delimitations within s. We use $\emptyset$ to denote the empty substitution, $\mid \sigma \mid$ to denote the number of pairs in σ, and $\sigma_1 \uplus \sigma_2$ to denote the union of σ_1 and σ_2 when they have disjoint domains. Finally, in Table 3, we inductively define a predicate for checking communication conflicts: $\mathrm{noConf}(s, \mathrm{n}, \bar{v}, \ell)$ holds true if s cannot immediately perform a receive over the endpoint n matching $\bar{v}$ and generating a substitution σ with lesser pairs than ℓ.

The *labelled transition relation*, written $\xrightarrow{\alpha}$, is the least relation over terms induced by the rules in Table 4, where label α is generated by the following grammar:

Table 3. There are not conflicting receives along n matching $\bar{v}$

$$\mathrm{noConf}(\mathrm{u}!\bar{\epsilon},\mathrm{n},\bar{v},\ell) = \mathrm{noConf}(\mathbf{0},\mathrm{n},\bar{v},\ell) = \mathbf{true} \qquad \mathrm{noConf}(*\,s,\mathrm{n},\bar{v},\ell) = \mathrm{noConf}(s,\mathrm{n},\bar{v},\ell)$$

$$\mathrm{noConf}(\mathrm{n}'?\bar{w}.s,\mathrm{n},\bar{v},\ell) = \begin{cases} \mathbf{false} & \text{if } \mathrm{n}' = \mathrm{n} \wedge |\mathcal{M}(\bar{w},\bar{v})| < \ell \\ \mathbf{true} & \text{otherwise} \end{cases}$$

$$\mathrm{noConf}(s \mid s',\mathrm{n},\bar{v},\ell) = \mathrm{noConf}(s + s',\mathrm{n},\bar{v},\ell) = \mathrm{noConf}(s,\mathrm{n},\bar{v},\ell) \wedge \mathrm{noConf}(s',\mathrm{n},\bar{v},\ell)$$

$$\mathrm{noConf}([u]\,s,\mathrm{n},\bar{v},\ell) = \begin{cases} \mathrm{noConf}(s,\mathrm{n},\bar{v},\ell) & \text{if } u \notin \mathrm{n} \\ \mathbf{true} & \text{otherwise} \end{cases}$$

Table 4. μCOWS operational semantics

$$\frac{[\![\bar{\epsilon}]\!] = \bar{v}}{\mathrm{n}!\bar{\epsilon} \xrightarrow{\mathrm{n}\lhd\bar{v}} \mathbf{0}}\ \textit{(inv)} \qquad \mathrm{n}?\bar{w}.s \xrightarrow{\mathrm{n}\rhd\bar{w}} s\ \textit{(rec)} \qquad \frac{g \xrightarrow{\alpha} s}{g + g' \xrightarrow{\alpha} s}\ \textit{(choice)}$$

$$\frac{s \xrightarrow{\mathrm{n}\lhd[\bar{m}]\,\bar{v}} s' \qquad n \in \bar{v} \qquad n \notin (\mathrm{n} \cup \bar{m})}{[n]\,s \xrightarrow{\mathrm{n}\lhd[n,\bar{m}]\,\bar{v}} s'}\ \textit{(open}_{\textit{inv}}\textit{)} \qquad \frac{s \xrightarrow{\sigma\uplus\{x\mapsto v\}} s'}{[x]\,s \xrightarrow{\sigma} s'\cdot\{x \mapsto v\}}\ \textit{(del}_{\textit{com}}\textit{)}$$

$$\frac{s \xrightarrow{\mathrm{n}\rhd[\bar{y}]\,\bar{w}} s' \qquad x \in \bar{w} \qquad x \notin \bar{y}}{[x]\,s \xrightarrow{\mathrm{n}\rhd[x,\bar{y}]\,\bar{w}} s'}\ \textit{(open}_{\textit{rec}}\textit{)} \qquad \frac{s \xrightarrow{\mathrm{n}\,\sigma\uplus\{x\mapsto v\}\,\ell\,\bar{v}} s'}{[x]\,s \xrightarrow{\mathrm{n}\,\sigma\,\ell\,\bar{v}} s'\cdot\{x \mapsto v\}}\ \textit{(del}_{\textit{com}2}\textit{)}$$

$$\frac{s_1 \xrightarrow{\mathrm{n}\rhd\bar{v}} s_1' \qquad s_2 \xrightarrow{\mathrm{n}\lhd\bar{v}} s_2'}{s_1 \mid s_2 \xrightarrow{\emptyset} s_1' \mid s_2'}\ \textit{(match)} \qquad \frac{s \xrightarrow{\alpha} s' \qquad u \notin (\mathrm{u}(\alpha) \cup \mathrm{ce}(\alpha))}{[u]\,s \xrightarrow{\alpha} [u]\,s'}\ \textit{(del)}$$

$$\frac{s_1 \xrightarrow{\mathrm{n}\rhd\bar{w}} s_1' \qquad s_2 \xrightarrow{\mathrm{n}\lhd\bar{v}} s_2' \qquad \mathcal{M}(\bar{w},\bar{v}) = \sigma \qquad |\sigma| \geqslant 1 \qquad \mathrm{noConf}(s_1 \mid s_2,\mathrm{n},\bar{v},|\sigma|)}{s_1 \mid s_2 \xrightarrow{\mathrm{n}\,\sigma\,|\sigma|\,\bar{v}} s_1' \mid s_2'}\ \textit{(com)}$$

$$\frac{s_1 \xrightarrow{\mathrm{n}\,\sigma\,\ell\,\bar{v}} s_1' \qquad \mathrm{noConf}(s_2,\mathrm{n},\bar{v},\ell)}{s_1 \mid s_2 \xrightarrow{\mathrm{n}\,\sigma\,\ell\,\bar{v}} s_1' \mid s_2}\ \textit{(par}_{\textit{com}}\textit{)} \qquad \frac{s_1 \xrightarrow{\alpha} s_1' \qquad \alpha \neq \mathrm{n}\,\sigma\,\ell\,\bar{v}}{s_1 \mid s_2 \xrightarrow{\alpha} s_1' \mid s_2}\ \textit{(par)}$$

$$\frac{s \xrightarrow{\mathrm{n}\,\sigma\,\ell\,\bar{v}} s' \qquad n \in \mathrm{n}}{[n]\,s \xrightarrow{\sigma} [n]\,s'}\ \textit{(private)} \qquad \frac{s \equiv s_1 \qquad s_1 \xrightarrow{\alpha} s_2 \qquad s_2 \equiv s'}{s \xrightarrow{\alpha} s'}\ \textit{(str)}$$

$$\alpha \ ::= \ \mathrm{n}\lhd[\bar{n}]\,\bar{v} \ \mid \ \mathrm{n}\rhd[\bar{x}]\,\bar{w} \ \mid \ \sigma \ \mid \ \mathrm{n}\,\sigma\,\ell\,\bar{v}$$

Labels $\mathrm{n}\lhd[\bar{n}]\,\bar{v}$ and $\mathrm{n}\rhd[\bar{x}]\,\bar{w}$ denote execution of invoke and receive activities over the endpoint n, resp., while labels σ and $\mathrm{n}\,\sigma\,\ell\,\bar{v}$ denote execution of a communication with generated substitution σ to be still applied. Thus, $\emptyset$ and $\mathrm{n}\,\emptyset\,\ell\,\bar{v}$ denote *computational steps* corresponding to taking place of communication without pending substitutions. In the sequel, we will write $\mathrm{n}\lhd\bar{v}$ (resp. $\mathrm{n}\rhd\bar{w}$) instead of $\mathrm{n}\lhd[\,]\,\bar{v}$ (resp. $\mathrm{n}\rhd[\,]\,\bar{w}$) and use $\mathrm{u}(\alpha)$ to denote the set of names and variables occurring in α, where $\mathrm{u}(\mathrm{n}\,\sigma\,\ell\,\bar{v}) = \mathrm{u}(\sigma)$, $\mathrm{u}(\{x \mapsto v\}) = \{x\} \cup \mathrm{fu}(v)$ and $\mathrm{u}(\sigma_1 \uplus \sigma_2) = \mathrm{u}(\sigma_1) \cup \mathrm{u}(\sigma_2)$. We use $\mathrm{ce}(\alpha)$ to denote the names composing the endpoint if α denotes execution of a communication, i.e. $\mathrm{ce}(\alpha)$ is $\emptyset$ except for $\alpha = \mathrm{n}\,\sigma\,\ell\,\bar{v}$ for which we let $\mathrm{ce}(\mathrm{n}\,\sigma\,\ell\,\bar{v}) = \mathrm{n}$. Finally, we use $\mathrm{bu}(\alpha)$ to denote the set of names/variables that occur bound in α; i.e. $\mathrm{bu}(\alpha)$ is $\emptyset$ except for $\alpha = \mathrm{n}\lhd[\bar{n}]\,\bar{v}$ and $\alpha = \mathrm{n}\rhd[\bar{x}]\,\bar{w}$ for which we let $\mathrm{bu}(\mathrm{n}\lhd[\bar{n}]\,\bar{v}) = \bar{n}$ and $\mathrm{bu}(\mathrm{n}\rhd[\bar{x}]\,\bar{w}) = \bar{x}$.

We comment on salient points of rules in Table 4. An invocation can proceed only if the expressions in the argument can be evaluated *(inv)*. This means, for example, that if it contains a variable x it is stuck until x is not replaced by a value because of execution of a receive assigning a value to x. A receive activity offers an invocable operation along a given partner name *(rec)*, and the execution of a receive permits to take a decision between alternative behaviours *(choice)*. Bound invocations, that transmit private names, can be generated by *(open$_{inv}$)*, while delimited receive activities can proceed by *(open$_{rec}$)*.

Communication can take place when two parallel terms perform matching receive and invoke activities *(com)*. Communication generates a substitution that is recorded in the transition label (for subsequent application), rather than a silent transition as in most process calculi. In particular, two different kinds of communication label can be generated: σ and $\mathtt{n}\,\sigma\,\ell\,\bar{v}$. The latter label, produced by *(com)*, carries information about the communication which has taken place (i.e. the endpoint, the transmitted values, the generated substitution and its length) used to check the presence of conflicting receives in parallel components. Indeed, if more then one matching is possible, the receive that needs fewer substitutions is selected to progress (*(com)* and *(par$_{com}$)*). This mechanism permits to correlate different service communications thus implicitly creating a long-running interaction and can be exploited to model the precedence of a service instance over the corresponding service specification when both can process the same request. However, the check for the presence of a conflict is not needed when either the performed receive has the highest priority (i.e. the substitution has length 0) or the communication takes place along a private endpoint. In the former case, label $\emptyset$ is immediately generated by *(match)*. In the latter case, when the delimitation of a name belonging to the endpoint of a communication label is encountered (i.e. the communication is identified as private), the transition label $\mathtt{n}\,\sigma\,\ell\,\bar{v}$ is turned into σ *(private)*.

When the delimitation of a variable x argument of a receive involved in a communication is encountered, i.e. the whole scope of the variable is determined, the delimitation is removed and the substitution for x is applied to the term (*(del$_{com}$)* and *(del$_{com2}$)*); thus, x disappears from the term and cannot be reassigned a value. $[u]\ s$ behaves like s *(del)*, except when the transition label α contains u. Execution of parallel terms is interleaved *(par)*, but when a communication subject to conflict check is performed. Indeed, it must ensure that the receive activity with greater priority progresses (*(com)* and *(par$_{com}$)*).

Now, we want to define a co-inductive notion of *bisimulation* for the calculus. Since communication is asynchronous, an obvious starting point is considering as observable only the output capabilities of terms, as done by the labelled bisimulation introduced for asynchronous π-calculus in [1]. The intuition is that an asynchronous observer cannot directly observe the receipt of data that it has sent. Moreover, to enable compositional reasoning, we want our bisimulation to be a *congruence*, namely to be preserved by all μCOWS (closed) contexts $\mathbb{C}$ that are generated by the following grammar:

$$\mathbb{C} ::= [\![\cdot]\!] \;\mid\; \mathbb{G} \;\mid\; \mathbb{C} \mid s \;\mid\; s \mid \mathbb{C} \;\mid\; [u]\mathbb{C} \;\mid\; *\,\mathbb{C} \qquad \mathbb{G} ::= \mathtt{n}?\bar{w}.\mathbb{C} \;\mid\; \mathbb{G}+g \;\mid\; g+\mathbb{G}$$

such that, once the hole is filled with a closed term s, $\mathbb{C}[\![s]\!]$ is a μCOWS closed term.

In [13], we show that for μCOWS^m, a fragment of μCOWS that dispenses with priority in parallel composition, a notion of bisimulation inspired to [1] enjoys the equality

$$[x]\,(\,\emptyset + \mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle\,) \;=\; \emptyset \tag{3}$$

where, for the sake of presentation, we exploit the context $\emptyset + \llbracket\cdot\rrbracket \triangleq [\mathtt{m}]\,(\mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle + \llbracket\cdot\rrbracket)$ and the term $\emptyset \triangleq [\mathtt{m}]\,(\mathtt{m}!\langle\rangle \mid \mathtt{m}?\langle\rangle)$[1]. Intuitively, the equality means that a term that emits the data it has received behaves as a term that simply performs an unobservable action and is an analogous of the input absorption law, i.e. $a(b).\,\bar{a}b + \tau = \tau$, characterising strong bisimilarity for asynchronous π-calculus [1].

In μCOWS, instead, the context $\mathbb{C} \triangleq [y, z]\,\mathtt{n}?\langle y, z\rangle.\,\mathtt{m}!\langle\rangle \mid \mathtt{n}!\langle v', v\rangle \mid \llbracket\cdot\rrbracket$ can tell the two terms above apart. In fact, we have $\mathbb{C}\llbracket\emptyset\rrbracket \xrightarrow{\mathtt{n}\,\emptyset\,2\,\langle v', v\rangle} \mathtt{m}!\langle\rangle \mid \emptyset$, where the term $(\mathtt{m}!\langle\rangle \mid \emptyset)$ can perform the invoke $\mathtt{m}!\langle\rangle$. Instead, the other term cannot properly reply because the receive $\mathtt{n}?\langle x, v\rangle$ has higher priority than $\mathtt{n}?\langle y, z\rangle$ when synchronising with the invocation $\mathtt{n}!\langle v', v\rangle$. Thus, $\mathbb{C}\llbracket[x]\,(\,\emptyset + \mathtt{n}?\langle x, v\rangle.\,\mathtt{n}!\langle x, v\rangle\,)\rrbracket$ can only evolve to terms that cannot immediately perform the activity $\mathtt{m}!\langle\rangle$. This means that μCOWS^m's notion of bisimulation is not a congruence for μCOWS. This is due to the fact that receive activities that exercise a priority (i.e. receives whose arguments contain some values) can be detected by an interacting observer (as shown by the above example). Hence, for a suitable notion of labelled bisimilarity for μCOWS, equation (3) does not hold. Now, consider the term $[x, x']\,(\,\emptyset + \mathtt{n}?\langle x, x'\rangle.\,\mathtt{n}!\langle x, x'\rangle\,)$. Since $\mathtt{n}?\langle x, x'\rangle$ does not exercise any priority on parallel terms, the context $\mathbb{C}$ cannot tell the term above and $\emptyset$ apart. Similarly, we have that $\emptyset + \mathtt{n}?\langle\rangle.\,\mathtt{n}!\langle\rangle$ and $\emptyset$ cannot be distinguished by $\mathbb{D} \triangleq \mathtt{n}?\langle\rangle.\,\mathtt{m}!\langle\rangle \mid \mathtt{n}!\langle\rangle \mid \llbracket\cdot\rrbracket$. Therefore, such pairs of terms should be considered as bisimilar.

Now, consider the terms $s_1 \triangleq [\mathtt{n}]\,(\,\mathtt{m}!\langle\mathtt{n}\rangle \mid \mathtt{n}!\langle\rangle\,)$ and $s_2 \triangleq [\mathtt{n}]\,\mathtt{m}!\langle\mathtt{n}\rangle$. Although the former also contains the subterm $\mathtt{n}!\langle\rangle$, they can both perform only the invocation along the endpoint $\mathtt{m}$. In fact, $\mathtt{n}!\langle\rangle$ is blocked since initially it is in the scope of $[\mathtt{n}]$ and afterwards no interacting partner can ever be able to receive along $\mathtt{n}$ (contexts of the form $\llbracket\cdot\rrbracket \mid \mathtt{m}?\langle x\rangle.\,x?\langle\rangle.\,\mathbf{0}$ are not allowed because of the syntactic constraint on the 'localisation' of names). Therefore, s_1 and s_2 should be considered as bisimilar. Instead, the natural asynchronous labelled bisimilarity derived from [1] would tell them apart and, hence, need to be weakened. Hence, we define a labelled bisimulation as a family of relations indexed with sets of names corresponding to the names that cannot be used by contexts (to test) for reception since they are dynamically exported private names.

Definition 1. *A* names-indexed family $\mathcal{F}$ of relations *is a set of symmetric binary relations $\mathcal{R}_{\mathcal{N}}$ on* μCOWS *closed terms, one for each set of names $\mathcal{N}$, i.e. $\mathcal{F} = \{\mathcal{R}_{\mathcal{N}}\}_{\mathcal{N}}$.*

Definition 2 (Labelled bisimilarity). *A names-indexed family of relations $\{\mathcal{R}_{\mathcal{N}}\}_{\mathcal{N}}$ is a* labelled bisimulation *if, whenever $s_1 \mathcal{R}_{\mathcal{N}} s_2$ and $s_1 \xrightarrow{\alpha} s_1'$, where* $\mathrm{bu}(\alpha)$ *are fresh, then:*

1. *if $\alpha = \mathtt{n} \triangleright [\bar{x}]\,\bar{w}$ then one of the following holds:*
   (a) $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\triangleright[\bar{x}]\,\bar{w}} s_2'$ *and*
   $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\bar{x}|) \,:\, s_1'\cdot\sigma\, \mathcal{R}_{\mathcal{N}}\, s_2'\cdot\sigma$
   (b) $|\bar{x}| = |\bar{w}|$ *and* $\exists\, s_2' \,:\, s_2 \xrightarrow{\emptyset} s_2'$ *and*
   $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma$ *and* $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\bar{x}|) \,:\, s_1'\cdot\sigma\, \mathcal{R}_{\mathcal{N}}\, (s_2' \mid \mathtt{n}!\bar{v})$
2. *if $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$ where $\ell = |\bar{v}|$ then one of the following holds:*
   (a) $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_2'$ *and* $s_1'\, \mathcal{R}_{\mathcal{N}}\, s_2'$ (b) $\exists\, s_2' \,:\, s_2 \xrightarrow{\emptyset} s_2'$ *and* $s_1'\, \mathcal{R}_{\mathcal{N}}\, s_2'$

[1] $\emptyset$ plays a role similar to τ in π-calculus, namely $\emptyset$ is both a label and a term such that $\emptyset \xrightarrow{\emptyset} \mathbf{0}$.

3. *if* $\alpha = \mathtt{n} \lhd [\bar{n}]\,\bar{v}$ *where* $\mathtt{n} \notin \mathcal{N}$ *then* $\exists\, s_2' \,:\, s_2 \xrightarrow{\mathtt{n}\lhd[\bar{n}]\,\bar{v}} s_2'$ *and* $s_1' \,\mathcal{R}_{\mathcal{N}\cup\bar{n}}\, s_2'$
4. *if* $\alpha = \emptyset$ *or* $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$, *where* $\ell \neq |\bar{v}|$, *then* $\exists\, s_2' \,:\, s_2 \xrightarrow{\alpha} s_2'$ *and* $s_1' \,\mathcal{R}_{\mathcal{N}}\, s_2'$

Two closed terms s_1 *and* s_2 *are* $\mathcal{N}$-bisimilar*, written* $s_1 \sim_\mu^{\mathcal{N}} s_2$*, if* $s_1 \mathcal{R}_{\mathcal{N}} s_2$ *for some* $\mathcal{R}_{\mathcal{N}}$ *in a labelled bisimulation. They are* labelled bisimilar*, written* $s_1 \sim_\mu s_2$*, if they are* $\emptyset$*-bisimilar.* $\sim_\mu^{\mathcal{N}}$ *is called* $\mathcal{N}$*-bisimilarity, while* $\sim_\mu$ *is called labelled bisimilarity.*

The resulting definition somewhat recalls that of quasi-open bisimilarity for π-calculus [12]. Clause 1 deals with both observable and unobservable receives. In fact, all receives can be simulated in a normal way (clause 1.(a)); additionally, receives such that $|\bar{x}|=|\bar{w}|$, i.e. $\bar{w}$ contains only variables or is the empty tuple (since $\bar{x} \subseteq \bar{w}$ and $\bar{w}\backslash\bar{x}$ does not contain variables), can be simulated by a computational step leading to a term that, when composed with the invoke activity consumed by the receive, stands in the appropriate relation (clause 1.(b)). Execution of receives whose argument contains variables leads to open terms, which the operational semantics is not defined for. Since the freed variables are placeholders for values to be received, clause 1 requires the two continuations to be related for any matching tuple of values that can be effectively received (i.e. that do not give rise to communication conflicts). Clause 2 permits replying also with an $\emptyset$-transition to communications involving an unobservable receive (ℓ=$|\bar{v}|$ implies that the argument of the receive is a, possible empty, tuple of variables). Clause 3, and the use of names-indexed families of relations, handles the fact that dynamically exported private names cannot be used by a receiver within the endpoint of a receive (whose syntax does not allow to use variables). With abuse of notation, $\mathtt{n} \notin \mathcal{N}$ in clause 3, with $\mathtt{n} = p \bullet o$, stands for $p \notin \mathcal{N} \wedge o \notin \mathcal{N}$. Thus, invocations along endpoints using either of the names in $\mathcal{N}$ are unobservable, hence these endpoints cannot be used to tell the executing terms apart. Finally, clause 4 deals with computational steps. Notably, actions σ and $\mathtt{n}\,\sigma\,\ell\,\bar{v}$, with $\sigma \neq \emptyset$, are not taken into account, since they cannot be performed by closed terms (see rules *(com)*, *(del$_{com}$)* and *(del$_{com2}$)*).

Theorem 1. $\sim_\mu$ *is a congruence for* μCOWS *closed terms.*

As a further evidence of the reasonableness of our notion of bisimilarity, we provide now an alternative characterization in terms of *(open) barbed bisimilarity* along the line of [8,12]. To this aim, first we identify an appropriate *basic observable*, namely a predicate that points out the interaction capabilities of a term. Since communication is asynchronous, again we consider as observable only the output capabilities of terms.

Definition 3 (Observable for μCOWS). *Let* s *be a* μCOWS *closed term. Predicate* $s \downarrow_{\mathtt{n}}$ *holds true if there exist* s'*,* $\bar{n}$ *and* $\bar{v}$ *such that* $s \xrightarrow{\mathtt{n}\lhd[\bar{n}]\,\bar{v}} s'$*.*

Definition 4 (Open barbed bisimilarity). *A symmetric binary relation* $\mathcal{R}$ *on* μCOWS *closed terms is an* open barbed bisimulation *if whenever* $s_1 \mathcal{R} s_2$ *the following holds:*

(Barb preservation) *if* $s_1 \downarrow_{\mathtt{n}}$ *then* $s_2 \downarrow_{\mathtt{n}}$*;*

(Computation closure) *if* $s_1 \xrightarrow{\emptyset} s_1'$ *(resp.* $s_1 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_1'$*) then there exists* s_2' *such that* $s_2 \xrightarrow{\emptyset} s_2'$ *(resp.* $s_2 \xrightarrow{\mathtt{n}\,\emptyset\,\ell\,\bar{v}} s_2'$ *or* ℓ=$|\bar{v}| \wedge s_2 \xrightarrow{\emptyset} s_2'$*) and* $s_1' \mathcal{R} s_2'$*;*

(Context closure) $\mathbb{C}[\![s_1]\!]\,\mathcal{R}\,\mathbb{C}[\![s_2]\!]$, *for every closed context* $\mathbb{C}$.

Two closed terms s_1 and s_2 are open barbed bisimilar, *written $s_1 \simeq_\mu s_2$, if $s_1 \mathcal{R} s_2$ for some open barbed bisimulation $\mathcal{R}$. $\simeq_\mu$ is called open barbed bisimilarity.*

Barbed bisimilarity has the advantage of an intuitive meaning, since it is induced by a simple notion of observable and is defined by means of computation and context closure. Differently from $\sim_\mu$, the definition of $\simeq_\mu$ suffers from universal quantification over all possible language contexts, which makes the reasoning on terms very hard.

Our main results state that labelled bisimilarity is *sound* and *complete* with respect to open barbed bisimilarity. Due to the intuitiveness of $\simeq_\mu$, this result makes us confident that the notion of labelled bisimularity is sufficiently reasonable.

Theorem 2. *$\sim_\mu$ and $\simeq_\mu$ coincide.*

Our semantic theories extend in a standard way to the weak case so that results of congruence and coincidence still hold. Due to space limitations, the exact definitions are relegated to [13]. Here, we conclude with an example inspired to the law $!(a(b).\,\bar{a}b) = \mathbf{0}$ that holds for weak bisimilarity in asynchronous π-calculus [1]. In fact, the analogous of equality (3) for the weak case is $*\,[x,x']\,\mathtt{n}?\langle x,x'\rangle.\,\mathtt{n}!\langle x,x'\rangle \approx_\mu \mathbf{0}$. To prove validity, the most significant case is simulating the transition

$$*\,[x,x']\,\mathtt{n}?\langle x,x'\rangle.\,\mathtt{n}!\langle x,x'\rangle \xrightarrow{\mathtt{n}\,\rhd\,[x,x']\,\langle x,x'\rangle} *\,[x,x']\,(\mathtt{n}?\langle x,x'\rangle.\,\mathtt{n}!\langle x,x'\rangle) \mid \mathtt{n}!\langle x,x'\rangle$$

The term on the right replies with an empty transition and it is easy to show that, for all v and v', $(*\,[x,x']\,(\mathtt{n}?\langle x,x'\rangle.\,\mathtt{n}!\langle x,x'\rangle) \mid \mathtt{n}!\langle v,v'\rangle)$ and $(\mathbf{0} \mid \mathtt{n}!\langle v,v'\rangle)$ are weak bisimilar.

4 COWS

COWS is obtained by enriching μCOWS as follows:

$$s \;::=\; \ldots \quad\mid\quad \mathbf{kill}(k) \quad\mid\quad \{\!|s|\!\} \quad\mid\quad [k]\,s$$

Besides the sets of values and variables, we also use the set of *(killer) labels* (ranged over by $k, k', \ldots$). Notably, expressions do not include killer labels (that, hence, are *not* communicable). Delimitation now is a binder also for killer labels and, differently from the scope of names and variables, that of killer labels *cannot* be extended. A COWS term is *closed* if it does not contain free variables and killer labels.

Informally, execution of a *kill* activity $\mathbf{kill}(k)$ causes termination of all parallel terms inside the enclosing $[k]$, which stops the killing effect. Critical activities can be protected from the effect of a forced termination by using the *protection* operator $\{\!|s|\!\}$. E.g., $K \triangleq \mathtt{n}!\langle v\rangle \mid [k]\,(\,[x]\,\mathtt{n}?\langle x\rangle.s \mid \{\!|\mathtt{m}!\langle v'\rangle|\!\} \mid \mathbf{kill}(k)\,)$ can perform a *computational step* $\dagger$ by executing the kill activity and evolving to $(\mathtt{n}!\langle v\rangle \mid [k]\,(\,\mathit{halt}([x]\,\mathtt{n}?\langle x\rangle.s) \mid \mathit{halt}(\{\!|\mathtt{m}!\langle v'\rangle|\!\})\,)) \equiv (\mathtt{n}!\langle v\rangle \mid [k]\,\{\!|\mathtt{m}!\langle v'\rangle|\!\})$, where function $\mathit{halt}(\_)$, given a term s, returns the term obtained by only retaining the protected activities inside s. COWS's priority mechanism assigns greatest priority to kill activities so that they pre-empt all other activities inside the enclosing killer label's delimitation. For example, in the term K above communication along $\mathtt{n}$ and activity $\mathtt{m}!\langle v'\rangle$ are blocked until the kill activity has been performed. We refer to [13] for a complete account of COWS semantics.

When considering observational semantics for COWS we soon discover that $\sim_\mu$ is not preserved by those contexts forcing termination of the activities in the hole. For example, $\emptyset \sim_\mu \{\!|\emptyset|\!\}$ trivially holds. However, the COWS context $[k]\,(\mathbf{kill}(k) \mid \llbracket\cdot\rrbracket)$ can tell the two terms apart. Indeed, $[k]\,(\mathbf{kill}(k) \mid \emptyset) \sim_\mu [k]\,(\mathbf{kill}(k) \mid \{\!|\emptyset|\!\})$ does not hold since, after execution of the kill activity (that has highest priority), we would get $\mathbf{0} \sim_\mu \{\!|\emptyset|\!\}$ which is trivially false. Therefore, $\sim_\mu$ is not a congruence for COWS.

Open barbed bisimilarity, namely $\simeq$, is by definition closed under all contexts and its definition only needs to be tuned for considering also †-transitions in the 'Computation closure' (the exact definition is in [13]). Instead, labelled bisimilarity must explicitly take care of the effects of execution of kill activities and of occurrences of the protection operator. These differences w.r.t. to Definition 2 are highlighted with a gray background.

Definition 5 (Labelled bisimilarity). *A names-indexed family of relations* $\{\mathcal{R}_{\mathcal{N}}\}_{\mathcal{N}}$ *is a* labelled bisimulation *if* $s_1 \mathcal{R}_{\mathcal{N}} s_2$ *then* $halt(s_1)\,\mathcal{R}_{\mathcal{N}}\,halt(s_2)$ *and if* $s_1 \xrightarrow{\alpha} s'_1$, *where* $\mathrm{bu}(\alpha)$ *are fresh, then: we replace clauses 1.(b) and 4 in Definition 2 as follows (other clauses are identical).*

1. *(b)* $|\bar{x}|=|\bar{w}|$ *and* $\exists\, s'_2 : s_2 \xrightarrow{\emptyset} s'_2$ *and* $\forall\, \bar{v}$ *s.t.* $\mathcal{M}(\bar{x}, \bar{v}) = \sigma$ *and*
   $\mathrm{noConf}(s_2, \mathtt{n}, \bar{w}\cdot\sigma, |\bar{x}|) \,:\, s'_1\cdot\sigma\,\mathcal{R}_{\mathcal{N}}\,(s'_2 \mid \mathtt{n}!\bar{v})$ *or* $s'_1\cdot\sigma\,\mathcal{R}_{\mathcal{N}}\,(s'_2 \mid \{\!|\mathtt{n}!\bar{v}|\!\})$

4. *if* $\alpha = \emptyset$, $\alpha = \dagger$ *or* $\alpha = \mathtt{n}\,\emptyset\,\ell\,\bar{v}$, *where* $\ell \neq |\bar{v}|$, *then* $\exists\, s'_2 \,:\, s_2 \xrightarrow{\alpha} s'_2$ *and* $s'_1\,\mathcal{R}_{\mathcal{N}}\,s'_2$

Two closed terms s_1 *and* s_2 *are* $\mathcal{N}$-bisimilar, *written* $s_1 \sim^{\mathcal{N}} s_2$, *if* $s_1 \mathcal{R}_{\mathcal{N}} s_2$ *for some* $\mathcal{R}_{\mathcal{N}}$ *in a labelled bisimulation. They are* labelled bisimilar, *written* $s_1 \sim s_2$, *if they are* $\emptyset$*-bisimilar.* $\sim^{\mathcal{N}}$ *is called* $\mathcal{N}$*-bisimilarity, while* $\sim$ *is called labelled bisimilarity.*

halt-closure takes into account kill activities performed by contexts ($halt(s)$ gets the same effect as of plunging s within the context $[k]\,(\mathbf{kill}(k) \mid \llbracket\cdot\rrbracket)$), while clause 4 takes into account kill activities that are active within the considered terms. Clause 1.(b) considers that if a closed term s performs a transition labelled by $\mathtt{n} \lhd [\bar{m}]\,\bar{v}$, then s contains an invoke of the form $\mathtt{n}!\bar{\epsilon}$, with $\llbracket\bar{\epsilon}\rrbracket = \bar{v}$, which can be either protected or not.

Theorem 3. (1) $\sim$ *is a congruence for* COWS *closed terms; and* (2) $\sim$ *and* $\simeq$ *coincide.*

Extension to the weak case is standard (definitions of the bisimilarities are in [13]). Again, results of congruence and coincidence hold.

We finish studying the relationship between the specifications of the Morra service introduced in Section 2. We can prove that (1) $\not\approx$ (2). Indeed, (1) can perform two transitions labelled by $evens \bullet throw \rhd [x_{id}, y_p, y_{num}]\,\langle x_{id}, y_p, y_{num}\rangle$ and by $odds \bullet throw \rhd [x_p, x_{num}]\,\langle first, x_p, x_{num}\rangle$ and, because of application of substitutions $\{x_{id} \mapsto first, y_p \mapsto cbB, y_{num} \mapsto 1\}$ and $\{x_p \mapsto cbA, x_{num} \mapsto 2\}$, evolve to $(\,cbA \bullet res!\langle first, win(2,1,1)\rangle \mid cbB \bullet res!\langle first, win(2,1,0)\rangle\,)$. (2) can properly simulate the above transitions but it can only evolve to $\{\!|\,cbA \bullet res!\langle first, w\rangle \mid cbB \bullet res!\langle first, l\rangle\,|\!\}$. Of course, the latter term behaves differently from the former one in presence of kill activities. In fact, given the context $\mathbb{C} \triangleq [k']\,(\,[\mathtt{n}]\,(\mathtt{n}!\langle\rangle \mid \mathtt{n}?\langle\rangle.\,\mathbf{kill}(k')) \mid \llbracket\cdot\rrbracket\,)$, we have that $\mathbb{C}\llbracket(1)\rrbracket \not\approx \mathbb{C}\llbracket(2)\rrbracket$.

If we modify the last two invokes in the high-level specification (1) as follows:

$$\begin{aligned} * \,[x_{id}, x_p, x_{num}, y_p, y_{num}]\,(\, & odds \bullet throw?\langle x_{id}, x_p, x_{num}\rangle \mid evens \bullet throw?\langle x_{id}, y_p, y_{num}\rangle \\ & \mid \{\!|\, x_p \bullet res!\langle x_{id}, win(x_{num}, y_{num}, 1)\rangle \mid y_p \bullet res!\langle x_{id}, win(x_{num}, y_{num}, 0)\rangle \,|\!\}\,) \end{aligned} \tag{4}$$

the problem persists, because (4) after the first two transitions always provides a response, while (2) could fail to provide a response in presence of kill activities. Instead, a term bisimilar to (4) can be obtained by replacing M in (2) by the following term:

$$[x_{id}, x_p, x_{num}, y_p, y_{num}]\,(\, odds \bullet throw?\langle x_{id}, x_p, x_{num}\rangle \mid evens \bullet throw?\langle x_{id}, y_p, y_{num}\rangle \mid \{\!|\, [k]\,(\ldots)\,|\!\}\,)$$

5 Related Work

Many process calculi with priority have been proposed in the literature [4]. In previous proposals, dynamic priorities are basically used to model scheduling approaches and real-time aspects (see e.g. [2,5]) while in COWS they are used for coordination, as well as for orchestration, purposes. For example, in the service *5F* of Section 2 they enable implementing a sort of ‘default’ behaviour, that returns *err* when a throw is not admissible. To the best of our knowledge (see also [4]), the interplay between dynamic priorities and local pre-emption, and their impact on semantic theories of processes have never been explored before. A termination construct similar to COWS’s **kill** activity has been introduced in [6] in the setting of a distributed pi-calculus, where it is used to model the failure of a node. Instead, COWS’s **kill** activity (in conjunction with protection and delimitation) is more flexible since it permits terminating parallel activities in a more selective way. [9,3] use labelled bisimilarities to prove compliance between service implementations and specifications for process calculi based on an explicit notion of *session* rather than on correlation. COWS’s barbed bisimilarities follow the approach of open barbed bisimilarities [8,12] rather than that of barbed congruence [11], i.e. quantification over contexts occurs recursively inside the definition of bisimilarity. COWS’s priority mechanisms make some receive actions observable (which leads to a novel notion of observation that refines the purely asynchronous one [1,7]), and require specific conditions on the labelled bisimilarities for these to be congruences.

References

1. Amadio, R.M., Castellani, I., Sangiorgi, D.: On Bisimulations for the Asynchronous pi-Calculus. Theoretical Computer Science 195(2), 291–324 (1998)
2. Bhat, G., Cleaveland, R., Lüttgen, G.: A Practical Approach to Implementing Real-Time Semantics. Annals of Software Engineering 7, 127–155 (1999)
3. Bruni, R., Lanese, I., Melgratti, H., Tuosto, E.: Multiparty sessions in SOC. In: Lea, D., Zavattaro, G. (eds.) COORDINATION 2008. LNCS, vol. 5052, pp. 67–82. Springer, Heidelberg (2008)
4. Cleaveland, R., Lüttgen, G., Natarajan, V.: Priorities in process algebra. In: Handbook of Process Algebra, pp. 391–424 (2001)
5. Fecher, H.: A Real-Time Process Algebra with Open Intervals and Maximal Progress. Nordic Journal of Computing 8(3), 346–365 (2001)

6. Francalanza, A., Hennessy, M.: A theory for observational fault tolerance. Journal of Logic and Algebraic Programming 73(1-2), 22–50 (2007)
7. Honda, K., Tokoro, M.: An Object Calculus for Asynchronous Communication. In: America, P. (ed.) ECOOP 1991. LNCS, vol. 512, pp. 133–147. Springer, Heidelberg (1991)
8. Honda, K., Yoshida, N.: On Reduction-Based Process Semantics. Theoretical Computer Science 151(2), 437–486 (1995)
9. Lanese, I., et al.: Disciplining Orchestration and Conversation in Service-Oriented Computing. In: SEFM, pp. 305–314. IEEE, Los Alamitos (2007)
10. Lapadula, A., Pugliese, R., Tiezzi, F.: A calculus for orchestration of web services. In: De Nicola, R. (ed.) ESOP 2007. LNCS, vol. 4421, pp. 33–47. Springer, Heidelberg (2007)
11. Milner, R., Sangiorgi, D.: Barbed Bisimulation. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 685–695. Springer, Heidelberg (1992)
12. Sangiorgi, D., Walker, D.: On Barbed Equivalences in pi-Calculus. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 292–304. Springer, Heidelberg (2001)
13. Tiezzi, F.: Specification and Analysis of Service-Oriented Applications. PhD Thesis, Univ. Florence (2009); http://rap.dsi.unifi.it/cows/theses/tiezzi_phdthesis.pdf

A Distributed and Oblivious Heap*

Christian Scheideler and Stefan Schmid

Institut für Informatik
Technische Universität München, D-85748 Garching, Germany

Abstract. This paper shows how to build and maintain a distributed heap which we call SHELL. In contrast to standard heaps, our heap is oblivious in the sense that its structure only depends on the nodes currently in the network but not on the past. This allows for fast join and leave operations which is desirable in open distributed systems with high levels of churn and frequent faults. In fact, a node fault or departure can be fixed in SHELL in a constant number of communication rounds, which significantly improves the best previous bound for distributed heaps. SHELL has interesting applications. First, we describe a robust distributed information system which is resilient to Sybil attacks *of arbitrary scale*. Second, we show how to organize heterogeneous nodes of *arbitrary* non-uniform capabilities in an overlay network such that the paths between any two nodes do not include nodes of lower capacities. This property is useful, e.g., for streaming. All these features can be achieved without sacrificing scalability: our heap has a de Bruijn like topology with node degree $O(\log^2 n)$ and network diameter $O(\log n)$, n being the total number of nodes in the system.

1 Introduction

In recent years, peer-to-peer systems have received a lot of attention both inside and outside of the research community. Major problems for these systems are how to handle a large churn, adversarial behavior, or participants with highly varying availability and resources. This is particularly the case in open peer-to-peer systems, where any user may join and leave at will. In this paper, we argue that many of these challenges can be solved by organizing the nodes in a distributed heap called *SHELL*.[1] SHELL is *oblivious*, which implies that its structure only depends on the nodes currently in the system but not on the past. It has turned out that this is a crucial property for systems with frequent membership changes as recovery and maintenance is simpler and faster. In fact, in SHELL, a join operation can be handled in $O(\log n)$ time and a leave operation in constant time, which is much better than the $O(\log^2 n)$ runtime bound previously known for scalable distributed heaps [3].

* Partly supported by the DFG-Project SCHE 1592/1-1. For the full version see [15].

[1] The name SHELL is due to the fact that nodes are organized in different layers in our network, where nodes "higher" in the heap can be protected and operate independently of nodes "lower" in the heap.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 571–582, 2009.

SHELL has a number of interesting applications. As a first case study, we construct a fault-tolerant distributed information system called *i-SHELL* which is resilient to churn and Sybil attacks of *any* size. Sybil attacks are a particularly cumbersome problem in open distributed systems: a user may join the system with a large number of identities in order to, e.g., take over responsibility for an unfair amount of the resources in the system, control or isolate certain parts of the overlay network, or flood the system with futile traffic. The key idea of i-SHELL is that nodes only connect to older nodes in the system so that nodes that were already in the system when the Sybil attack takes place are unaffected by it.

As a second case study, we show that SHELL can also be used to organize nodes with arbitrary capacities in an efficient manner. For example, in a scenario where nodes have non-uniform Internet connections, our *h-SHELL* system guarantees that the paths between two nodes with fast Internet connections only include nodes which are also fast while keeping a low congestion. This is a vital property, e.g., for streaming.

1.1 Model and Definitions

In order to present our key ideas in a clean way, we will use a high-level model for the design and analysis of the SHELL system. We assume that time proceeds in rounds, and all messages generated in round i are delivered in round $i+1$ as long as no node sends and receives more than a polylogarithmic amount of information. In other words, we assume the standard synchronous message-passing model with the restriction that a node can only communicate with nodes that it has currently links to. We do not consider the amount of time needed for internal computation as that will be very small in our case. Each node v in the system is associated with a key $key(v) \in \mathbb{N}$. Our heap will organize the nodes according to these keys. We assume the existence of a symmetry breaker (e.g., unique IP addresses) which allows us to order nodes with the same key so that we can assume w.l.o.g. that all keys are distinct. The *order* n_v of a node v is defined as the number of nodes w in the system with $key(w) < key(v)$. Intuitively, n_v represents the number of nodes that are above v in the heap.

The problem to be solved for the SHELL system is to find efficient and scalable algorithms for the following operations:

1. *v.join()*: Node v joins the system.
2. *v.leave()*: Node v leaves the system.
3. *v.rekey(x)*: Node v's key changes to x.

By "scalability" we mean that these operations can also be executed efficiently when performed *concurrently*.

Scalability is an important feature of SHELL. Messages can be routed fast while the traffic is distributed evenly among nodes. We measure the congestion as follows.

Definition 1 (Congestion). *The* congestion *at a node v is the number of packets forwarded by v in a scenario where each of the n nodes in the system wants to send a message to a random node.*

One application of SHELL is a distributed information system resilient to Sybil attacks. Formally, we will study the following type of attack.

Definition 2 (Sybil Attack). *Starting with time t_0, an attacker joins the network with an arbitrary number of nodes.*

Our goal is to ensure that *all nodes that joined the network before t_0* are safe against that Sybil attack.

1.2 Our Contributions

The main contribution of this paper is the presentation of a scalable and robust overlay network called SHELL. SHELL is a distributed heap with join and leave operations with asymptotically optimal runtime. In contrast to other distributed (as well as many standard sequential) heaps (e.g., *PAGODA* [3]), SHELL is oblivious, which allows it to react much more rapidly to dynamic changes: nodes can join in time $O(\log n)$ and leave in time $O(1)$. Another highlight of SHELL is its robustness. For example, we are not aware of any other structure which allows us to build a distributed information system resilient to Sybil attacks of arbitrary scale. We also show that SHELL has interesting applications, e.g., it can deal very efficiently with arbitrary variations in the capacities of the nodes. In summary, our distributed heap has the following properties.

1. *Scalability*: Nodes have degree $O(\log^2 n)$ and the network diameter is $O(\log n)$, where n is the network size. Congestion is bounded by $O(\log n)$ on expectation and $O(\log^2 n)$ w.h.p.[2], which is on par with well-known peer-to-peer networks like Chord [19].
2. *Dynamics*: Nodes can be integrated in $O(\log n)$ time and removed in $O(1)$ time.
3. *Robustness*: SHELL can be used to build robust distributed information systems, e.g., a system which is resilient to arbitrarily large Sybil attacks.
4. *Heterogeneity*: SHELL can organize arbitrarily heterogeneous nodes in an efficient way (e.g., for streaming).

1.3 Related Work

A heap is a standard data structure that has been extensively studied in computer science (e.g., [4]). There are several types of concurrent heap implementations such as *skip queues* [16] or *funnel trees* [17]. Moreover, distributed heaps have been used in the context of garbage collection in distributed programs. However, none of these constructions can be used to design scalable distributed systems like those considered in this paper.

[2] By "with high probability" or "w.h.p." we mean a probability of at least $1 - 1/poly(n)$.

A prominent way to build scalable distributed systems are distributed hash tables (or DHTs). Since the seminal works by Plaxton et al. [12] on locality-preserving data management in distributed environments and by Karger et al. [8] on consistent hashing, many DHTs have been proposed and implemented, e.g., Chord [19], CAN [13], Pastry [14], Tapestry [21], or D2B [7]. While these systems are highly scalable, they often assume nodes to be homogeneous, and they are vulnerable to various attacks, especially Sybil attacks that, if large enough, can cause network partitions in these DHTs.

Sybil attacks [6] are an important challenge for open distributed systems. A prominent example is our email system in which tons of spam emails are created by Sybils to evade filtering. A solution to the Sybil attack problem in practice is to have new subscribers solve difficult cryptographic puzzles which limits the rate at which participants can join, or to perform Turing tests to prevent automated subscriptions and to ensure that a new user is indeed a human being. Most of these solutions are based on centralized entities. In fact, a well-known result by Douceur [6] claims that in purely decentralized environments, it is inherently difficult to handle Sybil attacks. Douceur finds that the only means to limit the generation of multiple identities is to have a trusted authority be responsible for generating them. Bazzi et al. propose a Sybil defense based on network coordinates [1,2] in order to differentiate between nodes. Other approaches are based on social networks [5,20] and game theory [9]. All of these approaches are aiming at detecting and/or limiting Sybil attacks. In our paper, we do not aim at preventing Sybil attacks. We assume that an attacker can indeed connect an unbounded number of nodes to the network (by controlling, e.g., a botnet). Nevertheless, SHELL remains efficient at any time for those nodes that have already been in the system before the attack.

As a second application, we demonstrate how SHELL can organize heterogeneous nodes such that stronger nodes can operate independently of weaker nodes. While many peer-to-peer systems assume that nodes have uniform capabilities, there have also been several proposals to construct heterogeneous systems (e.g., [11,18]). These are usually based on multi-tier architectures but can only handle a certain subset of the capacity distributions well. The system closest to ours is PAGODA [3] which also constructs a distributed heap. However, this architecture is not oblivious. The more rigid structure implies that the system is less dynamic and cannot adapt to bandwidth changes nearly as quickly as SHELL. In fact, a join and leave operation take $O(\log^2 n)$ time, and it appears that without major modifications it is not possible to lower the runtime of the operations to something comparable with SHELL.

2 The SHELL Heap

In this section, we present the SHELL heap. Due to space constraints, the proofs were left out and can be found in [15].

2.1 The SHELL Topology

The SHELL topology is based on a dynamic de Bruijn graph and builds upon the continuous-discrete approach introduced by Naor and Wieder [10]. In the classical d-dimensional de Bruijn graph, $\{0,1\}^d$ represents the set of nodes and two nodes $x, y \in \{0,1\}^d$ are connected by an edge if and only if there is a bit $b \in \{0,1\}$ so that $x = (x_1 \dots x_d)$ and $y = (b\, x_1 \dots x_{d-1})$ (i.e., y is the result of a right shift of the bits in x with the highest bit position taken by b) or $y = (x_2 \dots x_d b)$. When viewing every node $x \in \{0,1\}^d$ as a point $\sum_{i=1}^{d} x_i/2^i \in [0,1)$ and letting $d \to \infty$, then the node set of the de Bruijn graph is equal to $[0,1)$ and two points $x, y \in [0,1)$ are connected by an edge if and only if $x = y/2$, $x = (1+y)/2$ or $x = 2y \pmod 1$. This motivates the dynamic variant of the de Bruijn graph described in the following.

For any $i \in \mathbb{N}_0$, a *level i interval* (or simply *i-interval*) is an interval of size $1/2^i$ starting at an integer multiple of $1/2^i$ in $[0,1)$. The *buddy* of an i-interval I is the other half of the $(i-1)$-interval that contains I. We assume that every node in the system is assigned to some fixed (pseudo-)random point in $[0,1)$ (the node set of the continuous de Bruijn graph above) when it joins the system. We also call this point its *identity* or *id*. For now, suppose that every node v knows its order n_v. Later in this section we present a local control strategy that allows the nodes to obtain a good estimate on n_v. We want to maintain the following condition at any point in time for some fixed and sufficiently large constant $c > 1$.

Condition 21. *Each node v has* forward edges *to all nodes w with key$(w) <$ key(v) in the following three intervals:*

- *the $\ell_{v,0}$-interval containing v (v's* home interval*) and its buddy,*
- *the $\ell_{v,1}$-interval containing $v/2$ and the $\ell_{v,2}$-interval containing* $(1+v)/2$ *(v's* de Bruijn intervals*) and their buddies.*

v also has backward edges *to all nodes that have forward edges to it.*

The level $\ell_{v,0} \in \mathbb{N}_0$ of v is chosen as the largest value such that the $\ell_{v,0}$-interval containing v contains at least $c \log n_v$ nodes w with key$(w) <$ key(v) for some fixed and sufficiently large constant c. If there is no such $\ell_{v,0}$ (i.e., n_v is very small), then $\ell_{v,0}$ is set to 0. The same rule is used for the selection of the levels $\ell_{v,i}$, $i \in \{1,2\}$, using the points $(i+v-1)/2$ *instead of v.*

The conditions on $\ell_{v,i}$ suffice for our operations to work w.h.p. If we want guarantees, one would have to extend the definition of $\ell_{v,i}$ to lower bounds on the number of nodes in both halves of the $\ell_{v,i}$-interval as well as its buddy, but for simplicity we will not require that here.

The forward edges are related to the upward edges in a standard (min-)heap while the backward edges are related to the downward edges. However, instead of a tree-like structure, we have a de Bruijn-like structure among the nodes. Forward edges to the home interval of a node are called *home edges* and edges to the de Bruijn intervals *de Bruijn edges*. Our construction directly yields the following properties.

Fact 22 (Oblivious Structure). *The SHELL topology only depends on the current set of nodes and their keys but not on the past.*

Fact 23 (Forward Independence). *The forward edges of a node v only depend on nodes u with $key(u) < key(v)$.*

Recall that every node is given a (pseudo-)random point in $[0, 1)$. Then the following property also holds.

Lemma 1 (Level Quasi-Monotonicity). *For any pair of nodes v and w with $key(v) > key(w)$ it holds that $\ell_{v,i} \geq \ell_{w,j} - 1$ for any $i, j \in \{0, 1, 2\}$, w.h.p.*

In this lemma and the rest of the paper, "w.h.p." means with probability at least $1 - 1/poly(n_v)$. Next we bound the degree of the nodes. From the topological conditions we can immediately derive the following property.

Lemma 2. *Every node v has $\Theta(c \log n_v)$ many forward edges to every one of its intervals, w.h.p.*

For the backward edges, we have the following bound, where n is the total number of nodes in the system.

Lemma 3. *The maximal number of backward edges of a node is limited by $O(c \log^2 n)$ w.h.p.*

2.2 Routing

We now present a routing algorithm on top of the described topology. For any pair (u, v) of nodes, the operation *route*(u, v) routes a message from node u to node v. The routing operation consists of two phases: first, a *forward*(v) operation is invoked which routes a message from node u to some node w with $key(w) < key(u)$ whose home interval contains v. Subsequently, if necessary (i.e., if $w \neq v$), a *refine*(v) operation performs a descent or ascent along the levels until (the level of) node v is reached. In the following, we will show how to implement the *route*(u, v) operation in such a way that a message is only routed along nodes w for which it holds that $key(w) \leq \max\{key(u), key(v)\}$.

Forward(v). We first consider the *forward*(v) algorithm, where node u sends a message along forward edges to a node whose home interval includes node v. Let $(x_1, x_2, x_3, \ldots)$ be the binary representation of u and $(y_1, y_2, y_3, \ldots)$ be the binary representation of v (i.e., $u = \sum_{i \geq 1} x_i/2^i$). Focus on the first $k = \log n_u$ bits of these representations. Ideally, we would like to send the message along the points $z_0 = (x_1, x_2, x_3, \ldots)$, $z_1 = (y_k, x_1, x_2, x_3, \ldots)$, $z_2 = (y_{k-1}, y_k, x_1, x_2, x_3, \ldots)$, $\ldots, z_k = (y_1, \ldots, y_k, x_1, x_2, x_3, \ \ldots)$. We emulate that in SHELL by first sending the message from node u along a forward edge to a node u_1 with largest key whose home interval contains z_1. u can indeed identify such a node since $z_1 = z_0/2$ or $z_1 = (1 + z_0)/2$, i.e., z_1 is contained in one of u's de Bruijn intervals, say, I. Furthermore, u has $\Theta(c \log n_u)$ forward edges to each of the two halves of its intervals, w.h.p., and from Lemma 1 it follows that every node w that u has a forward edge to, $\ell_{w,0} \leq \ell_{u,i} + 1$, i.e., w's home interval has at least half the size of I. Hence, u has a forward connection to a node u_1 whose home interval contains z_1 w.h.p. From u_1, we forward the message along a forward edge to a node u_2 with largest key whose home interval contains z_2. Again, u_1 can identify such a node since $z_2 = z_1/2$ or $z_2 = (1 + z_1)/2$ and z_1 belongs to the home interval of u_1, which implies that z_2 belongs to one of the de Bruijn intervals of u_1. We continue that way until a node u_k is reached whose home interval contains z_k, as desired. Observe that according to Lemma 1, u_k contains v in its home interval as the first k bits of u_k and v match and $\ell_{u_k,0} < k$, w.h.p., so the forward operation can terminate at u_k. We summarize the central properties of the forward phase in three lemmas. The first lemma bounds the dilation.

Lemma 4. *For any starting node u and any destination v, forward(v) has a dilation of $\log n_u$, w.h.p.*

The next lemma is crucial for the routing to be scalable.

Lemma 5. *In the forward phase, a packet issued by a node u of order n_u will terminate at a node of order at least $n_u/2$ w.h.p.*

As a consequence, we can bound the congestion.

Lemma 6. *For a random routing problem, the congestion at any node v is $O(\log n_v)$ on expectation and $O(\log^2 n_v)$ w.h.p.*

Refine(v). Recall that once the *forward(v)* operation terminates, the packet has been sent to a node w that contains the location of v in its home interval. In a second *refining* phase *refine(v)*, the packet is forwarded to the level of v in order to deliver it to v. First, suppose that the packet reaches a node w with $key(w) > key(v)$. According to Condition 21, w has forward connections to all nodes x in its home interval with $key(x) < key(w)$. Hence, w has a forward edge to v and can therefore directly send the packet to v.

So suppose that the packet reached a node w with $key(w) < key(v)$. In this case, w may not be directly connected to v since there will only be a forward edge from v to w (and therefore a backward edge from w to v) if w is in v's home interval, which might be much smaller than w's home interval. Therefore, the packet has to be sent downwards level by level until node v is reached. Suppose that w is at level ℓ in its home interval. We distinguish between two cases.

Case 1: $n_v \leq (3/2)n_w$. Then v and w can be at most one level apart w.h.p.: Since the interval size of w can be at most $2(1+\delta)c\log n_w/n_w$ for some constant $\delta > 0$ (that can be made arbitrarily small depending on c) w.h.p., two levels downwards there can be at most $(1+\delta)^2 c\log n_w/2$ nodes left in a home interval of that level that w has forward edges to, w.h.p. Moreover, there can be at most an additional $(1+\delta)(n_w/2)(1+\delta)c\log n_w/(2n_w) = (1+\delta)^2 c\log n_w/4$ nodes that v has forward edges to, which implies that the level of v must be larger than $\ell + 2$ w.h.p. Thus, w is either in v's home interval or its buddy, which implies that v has a forward edge to w (resp. w has a backward edge to v), so w can deliver the packet directly to v.

Case 2: $n_v > (3/2)n_w$. Then there must be at least one node x with $key(w) \leq key(x) < key(v)$ that is in the ℓ+1-interval containing v (which might be w itself) w.h.p. Take the node with largest such key. This node must satisfy $n_x \leq (3/2)n_w$ w.h.p., which implies that it is at level ℓ or $\ell + 1$ by Case 1, so w has a backward edge to that node and therefore can send the packet to it. The forwarding of the packet from x is continued in the same way as for w so that it reaches node v in at most $\log n_v$ hops.

For the refine operation, the following lemma holds.

Lemma 7. *For any starting node w and any node v, the refine(v) operation has a dilation of* $O(\log n_v)$*. Furthermore, the congestion at any node u is at most* $O(c \log n_u)$ *w.h.p.*

2.3 Join and Leave

Open distributed systems need to incorporate mechanisms for nodes to join and leave. Through these membership changes, the size of the network can vary significantly over time. A highlight of SHELL is its flexibility which facilitates very fast joins and leaves.

Join. We first describe the join operation. Recall that each node v is assigned to a (pseudo-)random point in $[0, 1)$ when it joins the system. For the bootstrap problem, we assume that node v knows an arbitrary existing node u in the system which can be contacted initially. Then the following operations have to be performed for $x \in \{v, v/2, (1 + v)/2\}$:

1. *forward(x)*: Route v's join request along forward edges to a node w with $key(w) \leq key(u)$ whose home interval contains x.
2. *refine(x)*: Route v's join request along forward or backward edges to a node w' with *maximum key* $\leq key(v)$ that contains x in its home interval.
3. *integrate(x)*: Copy the forward edges that w' has in its home interval and buddy to v (and check Condition 21 for the involved nodes).

Here, we use a slight extension of the refine operation proposed for routing. If $key(v) > key(w)$, there is no change compared to the old refine operation. However, if $key(v) < key(w)$, then we have to send v's join request upwards along the levels till it reaches a node w' with maximum key $\leq key(v)$ that contains v in its home interval. This is necessary because w may not have a forward edge to w'.

Observe that a membership change can trigger other nodes to upgrade or downgrade. To capture these effects formally, we define the *update cost* as follows.

Definition 3 (Update Cost). *The total number of links which need to be changed after a given operation (e.g., a single membership change) is referred to as the* update cost *induced by the operation.*

Theorem 24. *A join operation terminates in time* $O(\log n)$ *w.h.p. The update cost of a join operation is bounded by* $O(c \log^2 n)$ *w.h.p.*

Leave. If a node v leaves in SHELL, it suffices for its neighbors to drop v from their neighbor list, which can be done in constant time. Some of the nodes may then have to upgrade to a higher level, which also takes constant time w.h.p. as it suffices for a node to learn about its 3-hop neighborhood w.h.p. (due to the use of buddy intervals). Thus, we have the following result.

Theorem 25. *The leave operation takes a constant number of communication rounds w.h.p. Moreover, the update cost induced at other nodes (cf Definition 3) is bounded by $O(c \log^2 n)$ w.h.p.*

2.4 Rekey

There are applications where node keys are not static and change over time. For instance, in the heterogeneous peer-to-peer system described in Section 3, the available bandwidth at a node can decrease or increase dynamically. Our distributed heap takes this into account and allows for frequent rekey operations. Observe that we can regard a rekey operation as a leave operation which is immediately followed by a join operation at the same location in the ID space but maybe on a different partition level. While a node can downgrade in constant time, decrease key operations require collecting additional contact information, which takes logarithmic time. From our analysis of join and leave operations, the following corollary results.

Corollary 26. *In SHELL, a node can perform a rekey operation in time $O(\log n)$, where n is the total number of nodes currently in the system. The update cost induced at other nodes (cf Definition 3) is at most $O(c \log^2 n)$.*

2.5 Estimation of the Order

So far, we have assumed that nodes know their order in order to determine their level. Of course, an exact computation takes time and may even be impossible in dynamic environments. In the following, we will show that sufficiently good approximations of the correct partition level i can be obtained by *sampling.*

In order to find the best home interval, adjacent intervals, and de Bruijn interval sizes, a node v counts the number $B(j)$ of nodes in a given j-level interval it observes. Ideally, the smallest j with the property that the home interval contains at least $c \log n_v$ nodes of lower keys defines the forward edges. We now prove that if decisions are made with respect to these $B(j)$, errors are small in the sense that the estimated level is not far from the ideal level i.

Concretely, at join time, nodes do binary search to determine the level i according to the following rule: if $j > B(j)/c - \log B(j)$ then level j is increased, and otherwise, if $j < B(j)/c - \log B(j)$, j is decreased, until (in the ideal case) a level i with $i = B(i)/c - \log B(i)$ is found (or the level i closest to that).

The following lemma shows that this process converges and that the search algorithm efficiently determines a level which is at most one level off the ideal level with high probability.

Theorem 27. *Let $\widehat{i}$ be the level chosen by the sampling method and let i be the ideal level. It holds that $|\widehat{i} - i| \leq 1$ w.h.p.*

3 Applications

A distributed and oblivious heap structure turns out to be very useful in various application domains. In the following, we sketch two applications. For more information, we refer the reader to the technical report [15]. We first describe a fault-tolerant information system called *i-SHELL* which is resilient to Sybil attacks of arbitrary scale. Second, we show how our heap can be used to build a heterogeneous peer-to-peer network called *h-SHELL*.

3.1 i-SHELL

In order to obtain a robust distributed information system, we order the nodes with respect to their *join times* (i-order). Concretely, $key(v)$ is equal to the time step when v joined the system. For the bootstrap problem, we assume the existence of a *network entry point* assigning time-stamps to the nodes in a verifiable way. Recall that for this choice of the keys the forward connections of a node only depend on older nodes. Moreover, whenever two nodes u and v want to exchange messages, our routing protocol makes sure that these messages are only forwarded along nodes w that are at least as old as u or v. This has the following nice properties:

Churn. Suppose that there are some nodes frequently joining and leaving the system. Then the more reliable nodes can decide to reject re-establishing forward edges to such a node each time the node is back up, forcing it to obtain a new join time stamp from the entry point so that it can connect back to the system. In this way, the unreliable nodes are forced to the bottom of the SHELL system so that communication between reliable nodes (higher up in SHELL) will not be affected by them.

Sybil Attacks. Suppose that at some time t_0 the adversary enters the system with a huge number of Sybil nodes. Then any two nodes u and v that joined the SHELL system before t_0 can still communicate without being affected by the Sybils. The Sybils may try to create a huge number of backward edges to the honest nodes, but an honest node can easily counter that by only keeping backward edges to the T oldest nodes, for some sufficiently large threshold T. Moreover, Sybils could try to overwhelm the honest nodes with traffic. But also here the honest nodes can easily counter that by preferentially filtering out packets from the youngest nodes in case of congestion, so that packets from nodes that joined the system before t_0 can still be served in a timely manner.

Putting it together. We can combine SHELL with consistent hashing in order to convert it into a DHT that is robust to the Sybil attacks described above. One can show the following result:

Theorem 31. *For the nodes injected before t_0, insert, delete and find operations have a runtime of $O(\log n)$, where n is the number of nodes currently in the system, and their congestion is bounded by $O(\log^2 n)$, irrespective of a Sybil attack taking place after t_0.*

3.2 h-SHELL

As a second example, we sketch how to use our heap structure to build a peer-to-peer overlay called h-SHELL. h-SHELL takes into account that nodes can have heterogeneous bandwidths. The system can be used, e.g., for streaming. In h-SHELL, $key(v)$ is defined as the inverse of the bandwidth of v, i.e., the higher its bandwidth, the lower its key and therefore the higher its place in h-SHELL. Nodes may propose a certain bandwidth, or (in order to avoid churn) its neighbors in h-SHELL are measuring its bandwidth over a certain time period and propose an average bandwidth value for a node v that may be used for its key. When the bandwidth at a node changes, a fast rekey operation will reestablish the heap condition.

From the description of the SHELL topology it follows that whenever two nodes u and v communicate with each other, only nodes w with a bandwidth that is at least the bandwidth of u or v are used for that. Thus, in the absence of other traffic, the rate at which u and v can exchange information is essentially limited by the one with the smaller bandwidth. But even for arbitrary traffic patterns h-SHELL has a good performance. Using Valiant's random intermediate destination trick, the following property can be shown using the analytical techniques in PAGODA [3].

Theorem 32. *For any communication pattern, the congestion in h-SHELL in order to serve it is at most by a factor of $O(\log^2 n)$ higher w.h.p. compared to a best possible network of bounded degree for that communication pattern.*

4 Conclusion

The runtime bounds obtained for SHELL are optimal in the sense that scalable distributed heaps cannot be maintained at asymptotic lower cost. In future research, it would be interesting to study the average case performance and robustness of SHELL "in the wild".

References

1. Bazzi, R., Choi, Y., Gouda, M.: Hop Chains: Secure Routing and the Establishment of Distinct Identities. In: Proc. 10th Intl. Conf. on Principles of Distributed Systems, pp. 365–379 (2006)
2. Bazzi, R., Konjevod, G.: On the Establishment of Distinct Identities in Overlay Networks. In: Proc. 24th Symp. on Principles of Distributed Computing (PODC), pp. 312–320 (2005)
3. Bhargava, A., Kothapalli, K., Riley, C., Scheideler, C., Thober, M.: Pagoda: A Dynamic Overlay Network for Routing, Data Management, and Multicasting. In: Proc. 16th Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA), pp. 170–179 (2004)
4. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to Algorithms, 2nd edn. MIT Press, Cambridge (2001)

5. Danezis, G., Lesniewski-Laas, C., Kaashoek, F., Anderson, R.: Sybil-resistant DHT Routing. In: Proc. 10th European Symp. on Research in Computer Security, pp. 305–318 (2005)
6. Douceur, J.R.: The Sybil Attack. In: Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS), pp. 251–260 (2002)
7. Fraigniaud, P., Gauron, P.: D2B: A de Bruijn Based Content-Addressable Network. Elsevier Theoretical Computer Science 355(1) (2006)
8. Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., Lewin, D.: Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In: Proc. 29th ACM Symposium on Theory of Computing (STOC), pp. 654–663 (1997)
9. Margolin, N., Levine, B.: Informant: Detecting Sybils Using Incentives. In: Proc. 11th Intl. Conf. on Financial Cryptography and Data Security, pp. 192–207 (2007)
10. Naor, M., Wieder, U.: Novel Architectures for P2P Applications: the Continuous-Discrete Approach. In: Proc. 15th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 50–59 (2003)
11. Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., Löser, A.: Super-Peer-Based Routing and Clustering Strategies for RDF-Based Peer-to-Peer Networks. In: Proc. 12th International Conference on World Wide Web (WWW), pp. 536–543 (2003)
12. Plaxton, C.G., Rajaraman, R., Richa, A.W.: Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In: Proc. 9th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA), pp. 311–320 (1997)
13. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A Scalable Content-Addressable Network. In: Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, pp. 161–172 (2001)
14. Rowstron, A., Druschel, P.: Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001)
15. Scheideler, C., Schmid, S.: A Distributed and Oblivious Heap. In: Technical University Munich, Tech Report TUM-I0906 (2009)
16. Shavit, N., Zemach, A.: Scalable Concurrent Priority Queue Algorithms. In: Proc. 18th Annual ACM Symposium on Principals of Distributed Computing (PODC), pp. 113–122 (1999)
17. Shavit, N., Zemach, A.: Combining Funnels: A Dynamic Approach to Software Combining. Journal of Parallel and Distributed Computing 60 (2000)
18. Srivatsa, M., Gedik, B., Liu, L.: Large Scaling Unstructured Peer-to-Peer Networks with Heterogeneity-Aware Topology and Routing. IEEE Trans. Parallel Distrib. Syst. 17(11), 1277–1293 (2006)
19. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In: Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (2001)
20. Yu, H., Kaminsky, M., Gibbons, P., Flaxman, A.: SybilGuard: Defending Against Sybil Attacks via Social Networks. In: Proc. ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (2006)
21. Zhao, B., Kubiatowicz, J.D., Joseph, A.: Tapestry: An Infrastructure for Fault-Tolerant Widearea Location and Routing. Technical report, UC Berkeley, Computer Science Division Tecnical Report UCB/CSD-01-1141 (2001)

Proportional Response Dynamics in the Fisher Market

Li Zhang

Microsoft Research
1065 La Avenida
Mountain View, CA 94043
lzha@microsoft.com

Abstract. In this paper, we show that the proportional response dynamics, a utility based distributed dynamics, converges to the market equilibrium in the Fisher market with constant elasticity of substitution (CES) utility functions. By the proportional response dynamics, each buyer allocates his budget proportional to the utility he receives from each good in the previous time period. Unlike the tâtonnement process and its variants, the proportional response dynamics is a large step discrete dynamics, and the buyers do not solve any optimization problem at each step. In addition, the goods are always cleared and assigned to the buyers proportional to their bids at each step. Despite its simplicity, the dynamics converges fast for strictly concave CES utility functions, matching the best upperbound of computing the market equilibrium via solving a global convex optimization problem.

1 Introduction

The market equilibrium characterizes the efficient outcome in a competitive market and is a central notion in Economics. While much recent studies have been devoted to computing the market equilibrium, it is desirable, from both economic and computational perspective, to know how such equilibrium emerges when the agents dynamically respond to the market condition in a distributed fashion. In this paper, we show that for certain widely studied markets, namely, the Fisher market with constant elasticity of substitution (CES) utility functions, there is a utility based proportional response dynamics that converges to the market equilibrium, and it may converges fast, matching the bound by solving a global convex program via the ellipsoid or interior point method.

We consider the Fisher market in which there are distinct sellers and buyers. Further, each seller has one unit of divisible good for sale (so we do not distinguish seller and good), and each buyer i has a budget b_i and a utility function with the form $u_i(x_1, \cdots, x_n) = \sum_j (w_{ij}x_j)^{\rho_i}$ for some $0 < \rho_i \leq 1$. Such utility functions are the standard constant elasticity of substitution (CES) utility functions.[1] It includes the well studied linear Fisher market. We consider the market

[1] The standard form is actually $u_i(x_{i1}, \cdots, x_{in}) = (\sum_j (w_{ij}x_{ij})^{\rho_i})^{1/\rho_i}$, to make it 1-homogeneous.

S. Albers et al. (Eds.): ICALP 2009, Part II, LNCS 5556, pp. 583–594, 2009.

rule that after the buyers place bids to the goods, each good is allocated to a buyer at a proportion of the buyer's bid to the total bids placed to that good. By the proportional response dynamics, the buyer submits bids in discrete time steps and adjusts his bids according to the *utility* he receives from each good in the previous time step. Formally, if we denote by $b_{ij}(t)$ the bid of buyer i to good j at time t, then $b_{ij}(t+1) = \frac{u_{ij}(t)}{u_i(t)} b_i$ where $u_{ij}(t) = (w_{ij} \frac{b_{ij}(t)}{\sum_i b_{ij}(t)})^{\rho_i}$ and $u_i(t) = \sum_j u_{ij}(t)$.

From the above description, we can see that the proportional response dynamics is characteristically different from the standard tâtonnement market dynamics and its variants. In the tâtonnement process, the price of each good is gradually adjusted according to the excess of demand in the previous time step. The proportional response dynamics does not explicitly involve a price mechanism as it is based on the user's utility. Consequently, it requires much less information and no need to solve an optimization problem at each step. It is naturally distributed and guarantees the market clearance at each step. In addition, it is a large step discrete dynamics in the sense that the buyer does not gradually change his bid, and therefore there is no need to choose a sufficiently small step size as typically done in tâtonnement process. Yet, for CES utility functions, we show that the proportional response dynamics converges to the market equilibrium, and in the case when each $\rho_i < 1$, the proportional response dynamics converges much faster than the discretized tâtonnement process.

In the tâtonnement process, at each time step, the buyer computes the optimum bundle. Such strategy bears similarity to the best response dynamics in multi-player games. In contrast, in the proportional response dynamics, each buyer adapts his bid according to the utility received in the previous time step. This is similar to the family of dynamics based on the payoff reinforcement learning. Examples include the replicator dynamics in evolutionary games [13] and the multiplicative update algorithm in zero-sum games [12]. In these dynamics, the probability of playing each strategy is adjusted by a multiplicative factor determined by the payoff corresponding to that strategy. As we shall see later in the convergence proof, the proportional response can be reformulated as a multiplicative process. One major contribution of our work is to show there exists utility based dynamics that converges to the market equilibrium.

The proportional response dynamics has been studied in [20] for a market that models the bandwidth allocation in the peer to peer file sharing system. One important property in that model is that each good, the upload bandwidth, brings the same utility to the interested users. This is no longer the case in a Fisher market. Consequently, we can no longer apply the techniques used in [20]. In particular, the proportional response is no longer equivalent to a matrix scaling process, an important tool used in that paper. Instead, in this paper, we show that the Kullback-Leibler divergence between the allocation defined by the dynamics and the limiting market equilibrium approaches 0. Our proof is facilitated by the equivalence between the Eisenberg-Gale program and the market equilibrium [11]. This also renders the proof simpler and the technique more general than that in [20].

Admittedly, compared to the tâtonnement process, the proportional response dynamics applies to more specific markets. It remains an interesting direction to discover similar dynamics that converge to the market equilibrium in more general economies.

Related work. The Fisher market is a special case of the general exchange market. According to [5], it was first defined by Irving Fisher in 1891. The linear Fisher market is equivalent to the pari-mutuel method studied in [11]. In [11], Eisenberg and Gale established that the market equilibrium, which they called equilibrium probabilities, is the solution to a convex program, now commonly referred to as the Eisenberg-Gale program, and laid the foundation for many subsequent works. In Computer Science community, [9] first presented a polynomial time algorithm to approximate the market equilibrium in the linear market with bounded number of goods. [10] proposed a polynomial time combinatorial algorithm for computing the market equilibrium for the linear Fisher market. In [14,21], polynomial time algorithms are presented for computing the market equilibrium by solving the Eisenberg-Gale program.

There has also been a long history in studying the dynamics for converging to the market equilibrium. One particularly well studied dynamics is the tâtonnement process in which the price changes gradually according to the excess of demand. It was first considered by Walras in 1874. In a series of papers [2,1,3,17,19], several economists formulated the process and studied the convergence of the continuous tâtonnement dynamics, and it was shown that tâtonnement converges locally for economies satisfying weak gross substitutability (WGS). In [18], Norvig showed that a greedy bidding strategy, which can be regarded as a variant of tâtonnement process, converges to the market equilibrium in the set up considered by Eisenberg and Gale in [11]. More recently, in [7,6], it is shown that the discretized tâtonnement process converges for WGS utility functions, and [8] showed that an asynchronous variant also converges. In [4], a dynamics is presented for a perturbed keyword auction mechanism and shown to converge to the market equilibrium. That dynamics can also be regarded as a variant of tâtonnement process.

2 Preliminaries

Fisher market. A Fisher market is a bipartite market which distinguishes the role of buyer and seller. Each buyer i has a budget b_i, and each seller has a unit of divisible good for sale (and therefore we do not distinguish the sellers and the goods). Suppose there are m buyers and n goods. Each buyer's utility is defined as a function of the amount of each good he receives. In this paper, we consider the family of markets where a buyer's utility function has the form:

$$u_i(x_{i1}, \cdots, x_{in}) = \sum_{j=1}^{n} (w_{ij} x_{ij})^{\rho_i} ,$$

where $0 < \rho_i \le 1$, $w_{ij} \ge 0$, and x_{ij} represents the amount of good j allocated to the user i. Such utility functions are standard in Economics and satisfy constant elasticity of substitution (CES) property. One special case is the linear Fisher market where $\rho_i = 1$ for all $1 \leq i \leq m$. Without loss of generality, we assume that for each i, there exists j such that $w_{ij} > 0$, and for each j, there exists i, such that $w_{ij} > 0$. We denote $\rho_M = \max_i \rho_i \leq 1$.

Market equilibrium. In a Fisher market, given a price vector $\mathbf{p} = (p_1, \cdots, p_n)$ where p_j is the price of the good j, each buyer i can maximize his utility within his budget constraint. The optimum bidding is the solution to the following optimization problem.

$$\begin{aligned} &\max u_i(x_{i1}, \cdots, x_{in}) \,, \text{s.t.} \\ &\forall i, j \quad x_{ij} = b_{ij}/p_j \,, \\ &\forall i \quad \sum_j b_{ij} \leq b_i \,, \\ &\forall i, j \quad b_{ij} \geq 0 \,. \end{aligned} \tag{1}$$

If it happens that there exists a solution $\mathbf{b} = \{b_{ij}\}$ to the optimization problem for each buyer i and such that $\forall j \quad \sum_i b_{ij} = p_j$, we call the price vector together with the corresponding bidding and allocation $\mathbf{x} = \{x_{ij} = b_{ij}/p_j\}$ a *market equilibrium*.

It is known that

Lemma 1. *The Fisher market with CES utility functions always has an equilibrium. At the equilibrium, each good's price and each buyer's utility is unique.*

Approximate market equilibrium. The notion of approximate market equilibria measures the closeness of an allocation to an equilibrium. Suppose that p^* is the market equilibrium price. Following [9,15], the bidding vector $\mathbf{b} = \{b_{ij}\}$, with price vector $\mathbf{p} = \{p_j = \sum_i b_{ij}\}_j$ is an *ε-approximate market equilibrium* if

1. $(1-\varepsilon)p_j^* \leq p_j \leq (1+\varepsilon)p_j^*$.
2. For each i, $u_i \geq (1-\varepsilon)\tilde{u}_i$ where $\tilde{u}_i$ is the maximum utility of buyer i given the price vector $\mathbf{p}$.

We also define a stronger notion of the approximate market equilibrium. A bidding vector $\mathbf{b} = \{b_{ij}\}$ is called a *strong ε-approximate market equilibrium* if there exists a market equilibrium $\mathbf{b}^*$ such that for all i, j, $(1-\varepsilon)b_{ij}^* \leq b_{ij} \leq (1+\varepsilon)b_{ij}^*$. It is easily seen that in the Fisher market with concave utility functions, a strong ε-approximate market equilibrium is an $O(\varepsilon)$-approximate market equilibrium. The reverse might not be true.

Proportional response dynamics. As standard in the study of market dynamics, we consider the setup where the buyers face the same market, i.e. the same set of goods, budget constraint, and utility function, at each time step while a buyer decides his bids according to the outcome in the previous time steps.

Denote by $b_{ij}(t)$ the bid of buyer i on the good j at time t. The proportional response dynamics considered in this paper is defined as $b_{ij}(t+1) = b_i \frac{u_{ij}(t)}{u_i(t)}$, where $p_j(t) = \sum_i b_{ij}(t)$, $u_{ij}(t) = (w_{ij}b_{ij}(t)/p_j(t))^{\rho_i}$, and $u_i(t) = \sum_j u_{ij}(t)$. Intuitively, at each step, each buyer allocates the bid proportional to the utility he receives from each good in the previous time period. In addition, we require that $b_{ij}(0) > 0$ whenever $w_{ij} > 0$. We have that,

Lemma 2. *A market equilibrium is a fixed point of the proportional response dynamics.*

Proof. Consider a market equilibrium with the price vector $\mathbf{p}$ and the bidding vector $\mathbf{b}$. By the definition, the market equilibrium is the solution of the optimization problem (1).

Using Lagrangian multiplier, we have that for each i, there exists λ_i such that if $b_{ij} > 0$, then $\rho_i \left(\frac{w_{ij}}{p_j}\right)^{\rho_i} b_{ij}^{\rho_i - 1} = \lambda_i$. Thus, $u_{ij} = \left(\frac{w_{ij}b_{ij}}{p_j}\right)^{\rho_i} = b_{ij}\lambda_i/\rho_i$. That is, for any j, k with $b_{ij}, b_{ik} > 0$, $u_{ij}/u_{ik} = b_{ij}/b_{ik}$. Hence, $\mathbf{b}$ is a fixed point of the proportional response dynamics. □

Main results. The main result of the paper is

Theorem 1. *The proportional response dynamics converges to a market equilibrium in the Fisher market with CES utility functions.*

We establish the convergence rate of the dynamics in two cases, when $\rho_M < 1$ and when $\rho_M = 1$. Without loss of generality, let $\sum_i b_i = 1$ and $\sum_j w_{ij} = 1$ for every i. Let $W_1 = \frac{1}{\min_{w_{ij}>0} w_{ij}}$, $W_2 = \frac{1}{\min_i b_i}$, $W = nW_1W_2$, and $L = \log W$. Throughout this paper, log is assumed to be the natural logarithm. We assume that initially $b_{ij}(0) = \Omega\left(\frac{b_i}{n^{O(1)}}\right)$. This includes the case where each buyer splits his bid evenly among all the goods. About the convergence rate, we have that

Theorem 2. *When $\rho_M < 1$, it takes $O\left(\frac{L+\log(1/\varepsilon)}{(1-\rho_M)^2}\right)$ steps to reach a strong ε-approximate market equilibrium. When $\rho_M = 1$, it takes $O(W^3/\varepsilon^2)$ steps to reach an ε-approximate market equilibrium.*

We note that in the above bounds, the number of buyers m appears implicitly as $W_2 \geq m$. In the proportional response dynamics, each step takes $O(mn)$ arithmetic computation. When $\rho_M < 1$, the overall running time is bounded by $O(mn(L + \log(1/\varepsilon))/(1 - \rho_M)^2)$. This bound is comparable to $O((mnL)^{O(1)}$ $\log(1/\varepsilon))$ computation time obtained by solving a convex program via the ellipsoid or interior point methods, and it is much faster than the discretized tâtonnement process which typically takes either $O((mnL/\varepsilon)^{O(1)})$ [6] or $O(W^{O(1)}$ $\log(1/\varepsilon))$ [8] steps.

3 The Convergence Proof

In [11], Eisenberg and Gale characterize the market equilibrium in the linear Fisher market as the solution of a convex optimization problem. Their result easily extends to CES utility functions. Consider the Eisenberg-Gale program defined as

$$\begin{aligned} &\max \sum_i \frac{b_i}{\rho_i} \log u_i\,, \quad \text{s.t.} \\ &\forall i \quad u_i = \sum_j (w_{ij} x_{ij})^{\rho_i}\,, \\ &\forall j \quad \sum_i x_{ij} = 1\,, \\ &\forall i,j \quad x_{ij} \geq 0\,. \end{aligned} \tag{2}$$

The following statement is a straight forward extension of [11].

Lemma 3. *For the Fisher market with CES utility functions, an allocation $\boldsymbol{x} = \{x_{ij}\}$ is an equilibrium if and only if it is a solution to (2). Further, the value of each u_i is unique at a solution of (2).*

Now we proceed to prove the convergence of the proportional response dynamics.

Proof (Theorem 1).

The convergence proof consists of two steps. Consider the sequence of bids $\mathbf{b}(t) = \{b_{ij}(t)\}$ for $t = 0, 1, \cdots$. We first show that any limiting point of this sequence is a market equilibrium. This is done by showing that each user's utility and each good's price all converge to that at the market equilibrium. This is sufficient to guarantee the convergence when there is a unique market equilibrium, such as in the case where $\rho_M < 1$. When $\rho_M = 1$, an additional argument is needed to show that there can be at most one limiting point starting from any given initial condition.

1. Any limiting point of the dynamics is a market equilibrium

Take any market equilibrium $\mathbf{b}^* = \{b^*_{ij}\}$. Let u^*_i represent the utility of user i and p^*_j the price of good j at the equilibrium. Let $z_{ij}(t) = \left(\frac{b^*_{ij}}{b_{ij}(t)}\right)^{b^*_{ij}}$. When $b^*_{ij} = 0$, we define $z_{ij}(t) = 1$. For $b^*_{ij} > 0$, we have that

$$\begin{aligned} z_{ij}(t+1) &= \left(\frac{b^*_{ij}}{b_{ij}(t+1)}\right)^{b^*_{ij}} \\ &= \left(\frac{b^*_{ij} u_i(t)}{u_{ij}(t) b_i}\right)^{b^*_{ij}} \qquad \text{by that } b_{ij}(t+1) = \tfrac{u_{ij}(t)}{u_i(t)} b_i \\ &= \left(\frac{u^*_{ij}}{u^*_i} \cdot \frac{u_i(t)}{u_{ij}(t)}\right)^{b^*_{ij}} \qquad \text{by Lemma 2 } b^*_{ij}/b_i = u^*_{ij}/u^*_i. \end{aligned}$$

Since $u_{ij}(t) = (w_{ij} b_{ij}(t)/p_j(t))^{\rho_i}$ and $u^*_{ij} = (w_{ij} b^*_{ij}/p^*_j)^{\rho_i}$, we further have that

$$\begin{aligned}
z_{ij}(t+1) &= \left(\frac{b^*_{ij}/p^*_j}{b_{ij}(t)/p_j(t)}\right)^{\rho_i b^*_{ij}} \left(\frac{u_i(t)}{u^*_i}\right)^{b^*_{ij}} \\
&= \left(\frac{b^*_{ij}}{b_{ij}(t)}\right)^{\rho_i b^*_{ij}} \left(\frac{p_j(t)}{p^*_j}\right)^{\rho_i b^*_{ij}} \left(\frac{u_i(t)}{u^*_i}\right)^{b^*_{ij}} \\
&= z_{ij}(t)^{\rho_i} \left(\frac{p_j(t)}{p^*_j}\right)^{\rho_i b^*_{ij}} \left(\frac{u_i(t)}{u^*_i}\right)^{b^*_{ij}} . \qquad (3)
\end{aligned}$$

By the above equation, we effectively reformulate the proportional response dynamics to a multiplicative process which, as we shall prove soon, allows us define a potential function and prove its convergence. Let $\phi_i(t) = \prod_j z_{ij}(t)$, and $\phi(t) = \prod_i \phi_i(t)^{1/\rho_i}$, we have that

$$\begin{aligned}
\phi(t+1) &= \prod_i \prod_j z_{ij}(t+1)^{1/\rho_i} = \prod_i \prod_{j:b^*_{ij}>0} z_{ij}(t+1)^{1/\rho_i} \\
&= \prod_i \prod_{j:b^*_{ij}>0} z_{ij}(t) \left(\frac{p_j(t)}{p^*_j}\right)^{b^*_{ij}} \left(\frac{u_i(t)}{u^*_i}\right)^{b^*_{ij}/\rho_i} \qquad \text{by (3)} \\
&= \prod_i \phi_i(t) \prod_j \left(\frac{p_j(t)}{p^*_j}\right)^{\sum_{i:b^*_{ij}>0} b^*_{ij}} \prod_i \left(\frac{u_i(t)}{u^*_i}\right)^{\sum_{j:b^*_{ij}>0} b^*_{ij}/\rho_i} \\
&= \prod_i \phi_i(t) \prod_j \left(\frac{p_j(t)}{p^*_j}\right)^{p^*_j} \prod_i \left(\frac{u_i(t)}{u^*_i}\right)^{b_i/\rho_i} .
\end{aligned}$$

Write $\psi(t) = \prod_j \left(\frac{p_j(t)}{p^*_j}\right)^{p^*_j} \prod_i \left(\frac{u_i(t)}{u^*_i}\right)^{b_i/\rho_i}$, then $\phi(t+1) = \prod_i \phi_i(t)\psi(t)$. We have that

Lemma 4. *1. $\phi_i(t) \geq 1$, and $\phi_i(t) \leq \phi(t)$.*

2. $\psi(t) \leq 1$, and $\psi(t) = 1$ if and only if $u_i(t) = u^*_i$ for any i and $p_j(t) = p^*_j$ for any j.

3. $\phi(t+1) \leq \phi(t)^{\rho_M}\psi(t) \leq \phi(t)\psi(t)$.

Proof. 1. We observe that

$$\log \phi_i(t) = \sum_j b^*_{ij} \log \frac{b^*_{ij}}{b_{ij}(t)}$$

is the Kullback-Leibler(KL) divergence between $\mathbf{b}^*_i = \{b^*_{ij}\}$ and $\mathbf{b}_i = \{b_{ij}(t)\}$. Since $\sum_j b^*_{ij} = \sum_j b_{ij}(t) = b_i$, $\log \phi_i(t) \geq 0$ and hence $\phi_i(t) \geq 1$. By that $\rho_i \leq 1$ and $\phi_i(t) \geq 1$, we immediately have that $\phi_i(t) \leq \phi(t)$.

2. By that $\sum_j p^*_j = \sum_j p_j(t) = \sum_i b_i$, we have that $\sum_j p^*_j \log(p^*_j/p_j(t)) \geq 0$, and hence $\prod_j \left(\frac{p_j(t)}{p^*_j}\right)^{p^*_j} \leq 1$. By Lemma 3, $\sum_i \frac{b_i}{\rho_i} \log u_i(t) \leq \sum_i \frac{b_i}{\rho_i} \log u^*_i$. Hence

$\prod_i \left(\frac{u_i(t)}{u_i^*}\right)^{b_i/\rho_i} \leq 1$. Therefore $\psi(t) \leq 1$. The second half of the statement follows from the equality condition in the above two inequalities and Lemma 3.

3. By that $\phi_i(t) \geq 1$ and $\rho_M \geq \rho_i$, we have that

$$\phi(t+1) = \prod_i \phi_i(t)\psi(t) \leq \prod_i \phi_i(t)^{\rho_M/\rho_i}\psi(t) = \phi(t)^{\rho_M}\psi(t)\,.$$

Since $\rho_M \leq 1$ and $\phi(t) \geq 1$, $\phi(t+1) \leq \phi(t)\psi(t)$. □

By the above lemma, we now show that $\psi(t) \to 1$ when $t \to \infty$. By repeatedly applying Lemma 4.3, we have that $\phi(t+1) \leq \phi(0)\prod_{\tau=0}^{t}\psi(\tau)$. That is

$$\prod_{\tau=0}^{t}\psi(\tau) \geq \phi(t+1)/\phi(0) \geq 1/\phi(0)\,. \tag{4}$$

Since $b_{ij}(0) > 0$ if $w_{ij} > 0$, $\phi(0)$ is upper bounded. Together with the fact that $\psi(t) \leq 1$, (4) implies that $\psi(t) \to 1$ when $t \to \infty$. By Lemma 4.2, this in turn implies that $u_i(t) \to u_i^*$ for any i and $p_j(t) \to p_j^*$ for any j. By Lemma 3, any limiting point of the dynamics is a market equilibrium.

2. The dynamics always converges to a single market equilibrium

When $\rho_M < 1$, the market equilibrium is unique. The proportional dynamics converges to that unique market equilibrium from any initial condition. However, when some $\rho_i = 1$, there may exist multiple market equilibria. We shall show that it is impossible that the sequence $b_{ij}(t)$ has two distinctive limiting points. Suppose that $\mathbf{b}' = \{b'_{ij}\}$ is a limiting point of the sequence $b(t_0), b(t_1), \cdots$. By 1, we know that $\mathbf{b}'$ is a market equilibrium. Since we can choose any market equilibrium in the definition of z_{ij}, we now choose $\mathbf{b}^* = \mathbf{b}'$. Since $\mathbf{b}(t_k) \to \mathbf{b}'$, we have that $z_{ij}(t_k) \to 1$ and therefore $\phi(t_k) \to 1$ when $k \to \infty$. By that $\phi(t)$ is monotonically decreasing, and that $\phi(t) \geq 1$, we have for any infinite sequence $s_0, s_1, \cdots$, $\phi(s_k) \to 1$ when $k \to \infty$. Therefore, $\mathbf{b}(s_k) \to \mathbf{b}'$. That is, the dynamics always converges to a single market equilibrium. □

4 The Rate of Convergence

In this section, we present the convergence rate of the proportional response dynamics. We consider two cases, when $\rho_M < 1$ and when $\rho_M = 1$. In the former case, we are able to show a fast convergence of the dynamics; and in the latter case, we show that the dynamics converges in pseudo-polynomial time.

We make the assumption that $\sum_i b_i = 1$ and for each i, $\sum_j w_{ij} = 1$. Recall that $W_1 = \frac{1}{\min_{w_{ij}>0} w_{ij}}$ and $W_2 = \frac{1}{\min_i b_i}$, $W = nW_1W_2$, and $L = \log W$. We have that

Lemma 5. *At the equilibrium, $p_j^* \geq \frac{1}{W}$ for any j. When $\rho_M < 1$, $b_{ij}^* = \Omega\left(\left(\frac{1}{W^2}\right)^{1/(1-\rho_M)}\right)$ whenever $w_{ij} > 0$.*

Proof. For any j, consider a buyer i with $w_{ij} > 0$. If for every other k, $w_{ik} = 0$, then $p_j^* \geq b_i \geq 1/W_2$. Otherwise, suppose that $w_{ij}, w_{ik} > 0$. As in the proof of Lemma 2,

$$\left(\frac{w_{ij}}{p_j^*}\right)^{\rho_i} {b_{ij}^*}^{\rho_i - 1} = \left(\frac{w_{ik}}{p_k^*}\right)^{\rho_i} {b_{ik}^*}^{\rho_i - 1} . \tag{5}$$

By that $b_{ij}^* \leq p_j^*$, $b_{ik}^* \leq p_k^*$, and $\rho_i \leq 1$, we have

$$\left(\frac{w_{ij}}{p_j^*}\right)^{\rho_i} {p_j^*}^{\rho_i - 1} \leq \left(\frac{w_{ik}}{b_{ik}^*}\right)^{\rho_i} {b_{ik}^*}^{\rho_i - 1} .$$

Rearranging the terms, we have that $p_j^* \geq \left(\frac{w_{ij}}{w_{ik}}\right)^{\rho_i} b_{ik}^* \geq \frac{1}{W_1} b_{ik}^*$. Since there exists k such that $b_{ik}^* \geq b_i/n$, $p_j^* \geq \frac{b_i}{nW_1} \geq \frac{1}{W}$.

When $\rho_M < 1$, by applying (5) again, we have that

$$\begin{aligned} b_{ij}^* = b_{ik}^* \left(\frac{w_{ij} p_k^*}{w_{ik} p_j^*}\right)^{\rho_i/(1-\rho_i)} &\geq b_{ik}^* \left(\frac{1}{W_1} \cdot \frac{1}{nW_1 W_2}\right)^{\rho_i/(1-\rho_i)} \\ \geq b_{ik}^* \left(\frac{1}{nW_1^2 W_2}\right)^{\rho_M/(1-\rho_M)} &= \Omega\left(\left(\frac{1}{W^2}\right)^{1/(1-\rho_M)}\right) . \end{aligned}$$

□

We will need the following technical lemma that bounds the difference between two vectors from their KL divergence.

Lemma 6. *For two positive sequences x_j and y_j for $j = 1, \cdots, n$ that satisfy $\sum_j x_j = \sum_j y_j$, let $\eta = \max_j \frac{|x_j - y_j|}{x_j}$. Then*

$$\sum_j x_j \log(x_j/y_j) \geq \frac{1}{16} \min(1, \eta) \eta \min_j x_j .$$

Proof. We use the well known inequality of $\sum_j x_j \log(x_j/y_j) \geq \frac{1}{2} \sum_j (\sqrt{x_j} - \sqrt{y_j})^2$. Suppose that $k = \arg\max_j \frac{|x_j - y_j|}{x_j}$. Then

$$\begin{aligned} \sum_j x_j \log(x_j/y_j) \geq \frac{1}{2}(\sqrt{x_k} - \sqrt{y_k})^2 &= \frac{1}{2}\left(\frac{x_k - y_k}{\sqrt{x_k} + \sqrt{y_k}}\right)^2 \\ &= \frac{1}{2}\eta^2 \frac{x_k^2}{(\sqrt{x_k} + \sqrt{y_k})^2} . \end{aligned}$$

When $y_k \leq 2x_k$, $x_k^2/(\sqrt{x_k} + \sqrt{y_k})^2 \geq x_k^2/(\sqrt{x_k} + \sqrt{2x_k})^2 \geq \frac{1}{8} x_k$.

When $y_k > 2x_k$, i.e. when $y = (\eta + 1)x_k$ and $\eta > 1$,

$$\frac{x_k^2}{(\sqrt{x_k} + \sqrt{y_k})^2} = \frac{x_k^2}{(\sqrt{x_k} + \sqrt{(1+\eta)x_k})^2} = \frac{1}{(1 + \sqrt{1+\eta})^2} x_k \geq \frac{1}{8\eta} x_k .$$

The last inequality follows from that $\eta > 1$. Hence, we have that

$$\sum_j x_j \log(x_j/y_j) \geq \frac{1}{16}\min(1,\eta)\eta x_k \geq \frac{1}{16}\min(1,\eta)\eta \min_j x_j\,.$$

□

According to Lemma 6, in order to show $\max_j \frac{|x_j - y_j|}{x_j} \leq \varepsilon < 1$, it suffices to show that $\sum_j x_j \log(x_j/y_j) = O(\varepsilon^2)\min_j x_j$. Now, we are ready to prove Theorem 2.

Proof (Theorem 2). (1) When $\rho_M < 1$, we show that the dynamics reaches a strong ε-approximate market equilibrium. By Lemma 4.2 and 3, $\phi(t+1) \leq \phi(t)^{\rho_M}\psi(t) \leq \phi(t)^{\rho_M}$. Applying the inequality iteratively, we have that $\phi(t) \leq \phi(0)^{\rho_M^t}$ and hence, $\phi_i(t) \leq \phi(t) \leq \phi(0)^{\rho_M^t}$, or $\log\phi_i(t) \leq \rho_M^t \log\phi(0)$.

As observed earlier, $\log\phi_i(t) = \sum_j b^*_{ij}\log\frac{b^*_{ij}}{b_{ij}(t)}$ is the KL divergence between b^*_{ij} and $b_{ij}(t)$. By Lemma 6, $b_{ij}(t)$ is a strong ε-approximate equilibrium as long as $\log\phi_i(t) \leq \frac{1}{16}\varepsilon^2\min_{j:b^*_{ij}>0} b^*_{ij}$ for $\varepsilon < 1$. Of course, how fast the process converges also depends on the initial choice of $b_{ij}(0)$. By the assumption that $b_{ij}(0) = \frac{b_i}{n^{O(1)}}$, $\phi_i(0) = \prod_j (b^*_{ij}/b_{ij}(0))^{b^*_{ij}} = O(\prod_j(\frac{b_i}{b_i/n^{O(1)}})^{b^*_{ij}}) = O(n^{O(b_i)})$. Hence $\phi(0) = O(n^{O(1)})$ by that $\sum_i b_i = 1$.

By choosing $t = c\frac{L+\log(1/\varepsilon)}{(1-\rho_M)^2}$ for sufficiently large c, we have that $\log\phi(t) \leq \rho_M^t\log\phi(0) = O((\frac{1}{W^2})^{1/(1-\rho_M)}\varepsilon^2)$. By Lemma 5 and 6, we have that $\frac{|b^*_{ij} - b_{ij}(t)|}{b^*_{ij}} \leq \varepsilon$ for any i, j, that is $\mathbf{b}(t) = \{b_{ij}(t)\}$ is a strong ε-approximation to $\mathbf{b}^*$.

(2) When $\rho_M = 1$, the convergence could be slower. Indeed, it may never converge to a strong ε-approximate equilibrium as at the market equilibrium, some b^*_{ij} may be 0 even when $w_{ij} > 0$. We will show that it converges to the standard notion of ε-approximate equilibrium in $O(W^3/\varepsilon^2)$ steps. We will establish the bound in the worst case of $\rho_i = 1$ for all i.

By Lemma 4.3, we have that $\phi(t+1) \leq \psi(t)\phi(t)$. We claim that if $\psi(t) \geq 1-\delta$ for $\delta = \frac{1}{256WW_2^2}\varepsilon^2$, then $\mathbf{b}(t)$ is an ε-approximate equilibrium. Recall that $\psi(t) = \prod_j\left(\frac{p_j(t)}{p^*_j}\right)^{p^*_j}\prod_i\left(\frac{u_i(t)}{u^*_i}\right)^{b_i}$. In what follows, we omit (t) for the simplicity of the notations. Since $\prod_j\left(\frac{p_j}{p^*_j}\right)^{p^*_j} \leq 1$ and $\prod_i\left(\frac{u_i}{u^*_i}\right)^{b_i} \leq 1$. By that $\psi \geq 1-\delta$, we have that

$$\prod_j\left(\frac{p_j}{p^*_j}\right)^{p^*_j} \geq 1-\delta\,, \tag{6}$$

$$\prod_i\left(\frac{u_i}{u^*_i}\right)^{b_i} \geq 1-\delta\,. \tag{7}$$

By (6), Lemma 5 and 6, we have that for every j, $(1-\varepsilon')p^*_j \leq p_j \leq (1+\varepsilon')p^*_j$ where $\varepsilon' = \varepsilon/(4W_2)$. Let $\tilde{u}_i$ be the maximum utility of the buyer i under the price vector $\mathbf{p}$. It remains to show that for each i, $u_i \geq (1-\varepsilon)\tilde{u}_i$. Since $\tilde{u}_i = b_i \max_j w_{ij}/p_j$ and $u_i^* = b_i \max_j w_{ij}/p_j^*$, we have that $u_i^* \geq (1-\varepsilon')\tilde{u}_i \geq (1-\varepsilon')u_i$. Then for any i,

$$\prod_j \left(\frac{u_j}{u_j^*}\right)^{b_j} \leq \left(\frac{u_i}{u_i^*}\right)^{b_i} \prod_{j\neq i} \left(\frac{1}{1-\varepsilon'}\right)^{b_j} \leq \frac{1}{1-\varepsilon'} \left(\frac{u_i}{u_i^*}\right)^{b_i} .$$

By (7), we have that $\frac{u_i}{u_i^*} \geq ((1-\delta)(1-\varepsilon'))^{1/b_i} \geq 1 - W_2(\delta+\varepsilon')$. Hence $u_i \geq (1-W_2(\delta+\varepsilon'))u_i^* \geq (1-W_2(\delta+\varepsilon'))(1-\varepsilon')\tilde{u}_i$. By the choice of δ, we have that $\delta, \varepsilon' \leq \varepsilon/(4W_2)$ and consequently $u_i \geq (1-\varepsilon)\tilde{u}_i$. Hence, $\mathbf{b}$ is an ε-approximate market equilibrium.

By (4), if for any $\tau \in [0,t]$, $\psi(\tau) < 1-\delta$, then $(1-\delta)^t \geq 1/\phi(0) \geq 1/n^{O(1)}$. Hence if $t \geq c \log n/\delta$ for some constant $c > 0$, there exists $\tau \leq t$ such that $\psi(\tau) \geq 1-\delta$. We thus have that the dynamics converges to an ε-approximate market equilibrium in $O(WW_2^2 \log n/\varepsilon^2) = O(W^3/\varepsilon^2)$ steps. □

5 Future Directions

One crucial property used in the convergence proof is the equivalence between the market equilibrium and the solution to the Eisenberg-Gale program. It would be interesting to know if the technique can extend to other Eisenberg-Gale markets as defined in [16]. We note that the proportional response dynamics most naturally applies to separable utility functions, i.e. the utility of each buyer is the sum of the utility obtained from different goods. It would be interesting to know if similar dynamics can be defined for more general families of such utility functions. It is also interesting to derive tight convergence bound when $\rho_M = 1$ and to know if the dynamics converges under certain asynchronous model.

References

1. Arrow, K., Block, H.D., Hurwicz, L.: On the stability of the competitive equilibrium, II. Econometrica 27(4), 82–109 (1959)
2. Arrow, K., Hurwicz, L.: On the stability of the competitive equilibrium, I. Econometrica 26(4), 522–552 (1958)
3. Arrow, K.J., Hurwicz, L.: Competitive stability and weak gross substitutability: the Euclidean distance approach. International Economic Review 1(1), 38–49 (1960)
4. Borgs, C., Chayes, J.T., Immorlica, N., Jain, K., Etesami, O., Mahdian, M.: Dynamics of bid optimization in online advertisement auctions. In: WWW, pp. 531–540 (2007)
5. Brainard, W., Scarf, H.: How to compute equilibrium prices in 1891. Cowles Foundation Discussion Paper, 1270 (2000)
6. Codenotti, B., McCune, B., Varadarajan, K.R.: Market equilibrium via the excess demand function. In: STOC, pp. 74–83 (2005)
7. Codenotti, B., Pemmaraju, S.V., Varadarajan, K.R.: On the polynomial time computation of equilibria for certain exchange economies. In: SODA, pp. 72–81 (2005)

8. Cole, R., Fleischer, L.: Fast-converging tatonnement algorithms for one-time and ongoing market problems. In: STOC, pp. 315–324 (2008)
9. Deng, X., Papadimitriou, C., Safra, S.: On the complexity of equilibria. In: STOC, pp. 67–71 (2002)
10. Devanur, N., Papadimitriou, C., Saberi, A., Vazirani, V.: Market equilibrium via a primal-dual-type algorithm. In: FOCS, pp. 389–395 (2002)
11. Eisenberg, E., Gale, D.: Consensus of subjective probabilities: the Pari-Mutuel method. Annals of Mathematical Statistics 30, 165–168 (1959)
12. Freund, Y., Schapire, R.E.: Adaptive game playing using multplicative weights. Games and Economic Behavior 29, 79–103 (1999)
13. Fudenberg, D., Levine, D.K.: The Theory of Learning in Games, ch. 3. MIT Press, Cambridge (1998)
14. Jain, K.: A polynomial time algorithm for computing the Arrow-Debreu market equilibrium for linear utilities. In: FOCS, pp. 286–294 (2004)
15. Jain, K., Mahdian, M., Saberi, A.: Approximating market equilibria. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. LNCS, vol. 2764, pp. 98–108. Springer, Heidelberg (2003)
16. Jain, K., Vazirani, V.V.: Eisenberg-gale markets: algorithms and structural properties. In: STOC, pp. 364–373 (2007)
17. Nikaido, H., Uzawa, H.: Stability and non-negativity in a Walrasian Tatonnement process. International Economic Review 1(1), 50–59 (1960)
18. Norvig, T.: Consensus of subjective probabilities: A convergence theorem. Annals of Mathematical Statistics 38(1), 221–225 (1967)
19. Uzawa, H.: Walras' tatonnement in the theory of exchange. Review of Economic Studies 27(3), 182–194 (1960)
20. Wu, F., Zhang, L.: Proportional response dynamics leads to market equilibrium. In: STOC, pp. 354–363 (2007)
21. Ye, Y.: A path to the Arrow-Debreu competitive market equilibrium. Mathematical Programming 111(1-2), 315–348 (2008)

Author Index