Anna Frid
Andrey Morozov
Andrey Rybalchenko
Klaus W. Wagner (Eds.)

LNCS 5675

Computer Science – Theory and Applications

4th International Computer Science Symposium in Russia, CSR 2009
Novosibirsk, Russia, August 2009, Proceedings

Springer

Anna Frid   Andrey Morozov
Andrey Rybalchenko   Klaus W. Wagner (Eds.)

Computer Science – Theory and Applications

4th International Computer Science Symposium
in Russia, CSR 2009
Novosibirsk, Russia, August 18-23, 2009
Proceedings

Springer

Volume Editors

Anna Frid
Andrey Morozov
Sobolev Institute of Mathematics SB RAS
4 Koptyug av, 630090 Novosibirsk, Russia
E-mail: {frid; morozov}@math.nsc.ru

Andrey Rybalchenko
Max Planck Institute for Software Systems
Campus E1 4, 66123 Saarbrücken, Germany
E-mail: rybal@mpi-sws.mpg.de

Klaus W. Wagner
Lehrstuhl für Theoretische Informatik
Institut für Informatik
Am Hubland, 97974 Würzburg, Germany
E-mail: wagner@informatik.uni-wuerzburg.de

Library of Congress Control Number: 2009931111

CR Subject Classification (1998): F.1.1, F.1.2, F.2, F.4, F.3, G.2

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

ISSN	0302-9743
ISBN-10	3-642-03350-4 Springer Berlin Heidelberg New York
ISBN-13	978-3-642-03350-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12724550 06/3180 5 4 3 2 1 0

Preface

The 4th International Computer Science Symposium in Russia (CSR 2009) was held August 18–23, 2009 in Novosibirsk, Russia, hosted by the Sobolev Institute of Mathematics and Novosibirsk State University. It was the fourth event in the series of regular international meetings, following CSR 2006 in St. Petersburg, CSR 2007 in Ekaterinburg, and CSR 2008 in Moscow.

The opening lecture was given by Andrei Voronkov, and four other invited plenary lectures were given by Sergei Odintsov, Wolfgang Thomas, Nikolai Vereshchagin, and Hongseok Yang.

This volume contains all the accepted papers and some of the abstracts of the invited speakers. The scope of the proposed topics for the symposium was quite broad and covered basically all areas of computer science. We received 66 papers in total, and the Program Committee selected 29.

Yandex provided the Best Student Paper Awards; the recepients of these awards were selected by the Program Committee:

- Dmitry Itsykson, “Structural complexity of AvgBPP”
- Yuri Pritykin and Julya Ulyashkina, “Aperiodicity measure for infinite sequences.”

The reviewing process was organized using the EasyChair conference system, created by Andrei Voronkov.

We are grateful to our sponsors:

- Russian Foundation for Basic Research
- Yandex (the largest Russian Internet portal providing key Web services).

We also thank the group of local organizers and in particular Pavel Salimov.

May 2009

Anna E. Frid
Andrey S. Morozov
Andrey Rybalchenko
Klaus Wagner

Organization

Program Committee

Farid Ablayev
Sergei Artemov
Thomas Ball
Lev Beklemishev
Josh Berdine
Véronique Bruyère
Cristian Calude
Bart Demoen
Christian Glaßer
Dmitry Grigoryev
Franjo Ivančić
Dietrich Kuske
Martin Leucker
Rupak Majumdar
Larisa Maksimova
Andrei Mantsivoda
Yuri Matiyasevich
Elvira Mayordomo
Andrey Morozov (Co-chair)
Pierre McKenzie
Greg Morrisett
Jean-Éric Pin
Arnd Poetzsch-Heffter
Andreas Rossberg
Andrey Rybalchenko (Co-chair)
Kai Salomaa
Victor Selivanov
Alexander Serebrenik
Natasha Sharygina
Henny Sipma
Ludwig Staiger
Helmut Veith
Andrei Voronkov
Klaus W. Wagner (Co-chair)
Eran Yahav
Andreas Zeller

Symposium Chair

Anna Frid

Steering Committee for CSR conferences

Volker Diekert
Anna Frid
Edward A. Hirsch
Juhani Karhumäki
Mikhail Volkov

External Reviewers

Pavel Alaev
Jean-Paul Allouche
Noga Alon
Andris Ambainis
Marcella Anselmo
Luís Antunes
Maxim Babenko
Dietmar Berwanger
Laurent Bienvenu
Albert Bifet

Henrik Björklund
Stephen Bloom
Manuel Bodirsky
Andrej Bogdanov
Bernd Borchert
Andreas Brandstädt
Roberto Bruttomesso
Peter Bürgisser
Andrei Bulatov
E. Rodney Canfield
Jean Cardinal
Olivier Carton
Baudouin Le Charlier
Arkadev Chattopadhyay
Alexander Chistov
Christian Choffrut
Evgeny Dantsin
Jürgen Dassow
Stéphane Demri
Hans van Ditmarsch
Michael Domaratzki
Roy Dyckhoff
Laurent Doyen
Guy Even
Uriel Feige
Bernard Fortz
Anna Frid
Aida Gainutdinova
Yuan Gao
Aristides Gionis
María Luisa González-Díaz
Geňa Hahn
Alfred Hairullin
Armin Hemmerling
Derek Holt
Markus Holzer
Oscar Ibarra
Gabriel Istrate
Kazuo Iwama
Gerold Jäger
Petr Jancar
Galina Jirásková
Valentine Kabanets
Marc Kaplan
Dmitry Karpov
Airat Khasianov
Daniel Kirsten
Johannes Köbler
Pascal Koiran
Fabrice Kordon
Margarita Korovina
Dieter Kratsch
Manfred Kufleitner
Sophie Laplante
Ranko Lazic
Ming Li
Yury Lifshits
Markus Lohrey
Irina Lomazova
María Lopez-Valdes
Antoni Lozano
Igor Makarov
Guillaume Malod
Radu Mardare
Nicole Megow
Daniel Meister
Dieter van Melkebeek
Leonid Melnikov
Wolfgang Merkle
Xavier Molinero
Rémi Morin
Philippe Moser
Matthias Müller-Hannemann
Anca Muscholl
Veta Murzina
Marius Nagy
Gethin Norman
Michael Nüsken
Alexander Okhotin
Elizaveta Okolnishnikova
Nicolas Ollinger
Vladimir Orevkov
Dmitry Pasechnik
Alexei Pastor
Xiaoxue Piao
Sylvain Perifel
Konstantin Pervyshev
Laure Petrucci
Giovanni Pighizzini
Ilia Ponomarenko

Denis Ponomaryov
Bert Randerath
Alexander Razborov
Christian Reitwießner
Boris Ryabko
Louis Salvail
Nicolae Santean
Patrice Séébold
Alexander Semenoff
Géraud Sénizergues
Alexander Shen
Nikolay Shilov
Vladimir Shpilrain
Howard Straubing
Alexei Stukachev
Alexei Talambutsa
Alain Tapp
Jacobo Torán
Sergey Vakulenko
Brigitte Vallée
Alexander Vasiliev
Nikolay Vereshchagin
Heribert Vollmer
Nikolai Vorobjov
Maximilian Witek
Ronald de Wolf
Sheng Yu
Liyu Zhang
Wieslaw Zielonka
Marius Zimand
Stefan van Zwam

Table of Contents

Well-Founded and Partial Stable Semantics Logical Aspects

Pedro Cabalar[1], Sergei Odintsov[2], and David Pearce[3,⋆]

[1] Corunna University, Corunna, Spain
cabalar@udc.es
[2] Sobolev Institute of Mathematics, Novosibirsk, Russia
odintsov@math.nsc.ru
[3] Universidad Rey Juan Carlos, Madrid, Spain
davidandrew.pearce@urjc.es

Abstract. This paper is devoted to logical aspects of two closely related semantics for logic programs: the partial stable model semantics of Przymusinski [20] and the well-founded semantics of Van Gelder, Ross and Schlipf [24]. For many years the following problem remained open: Which (non-modal) logic can be regarded as yielding an adequate foundation for these semantics in the sense that its minimal models (appropriately defined) coincide with the partial stable models of a logic program? Initial work on this problem was undertaken by Cabalar [5] who proposed a frame-based semantics for a suitable logic which he called HT^2. Preliminary axiomatics of HT^2 was presented in [6]. In this paper we analyse HT^2 frames and identify them as structures of a logic N^* having intuitionistic positive connectives and Routley negation and give a natural axiomatics for HT^2. We define a notion of minimal, total HT^2 model which we call *partial equilibrium model.* We show that for logic programs these models coincide with partial stable models, and we propose the resulting partial equilibrium logic as a logical foundation for partial stable and well-founded semantics. Finally, we discuss the strong equivalence for theories and programs in partial equilibrium logic.

Keywords: well-founded semantics, partial stable models, equilibrium logic, partial equilibrium logic.

1 Introduction

The *well-founded semantics* (WFS) of Van Gelder, Ross and Schlipf [24] has provided one of the most popular approaches to understanding and implementing default negation in logic programming. Its good computational properties

⋆ All authors gratefully acknowledge support from the CICyT project TIC-2003-9001-C02 and the MEC project TIN2006-15455-CO3. Additionally Pearce was partially supported by WASP (IST-2001-37004), the MICINN Consolider project Agreement Technologies and the project URJC-CM-2006-CET-0300. Odintsov was partially supported by Russian Foundation for Basic Research, project RFBR-08-07-00272-a.

have appealed to system developers and the well-known implementation XSB-Prolog[1] is now extensively used in AI problem solving. The partial stable model semantics suggested by Przymusinski [20] provides a natural bridge between the well-founded and the stable model semantics [15] that now forms the basis for the important, emerging paradigm of *answer set programming* (ASP).

The present paper describes the logical foundations of WFS and the partial stable (henceforth p-stable) model semantics. Our aim is to characterise a (non-modal) logic whose class of minimal models (in a suitable sense of minimal) coincides with the p-stable models and, on normal programs, contains the well-founded model as a special case.[2] We thereby obtain what we might term a *deductive base logic* for inference under p-stable and well-founded semantics.[3] We also obtain a means to extend p-stable semantics to arbitrary propositional theories.

Earlier this approach was successfully used to obtain a logical framework for the semantics of stable models and answer sets [17,18]. The logic of here-and-there, HT, (also known as Gödel's 3-valued logic) can be used to represent stable models as minimal models and can be shown to be a maximal logic with the property that equivalent theories have the same (stable model) semantics. It can also be shown [16] that HT characterises the important property of *strong equivalence* of programs (see below) under stable semantics. In the case of well-founded and p-stable semantics, foundational tools for studying program equivalences have been largely lacking.

P-stable models represent only one possible path to understanding and extending the well-founded semantics. For example the technique of [2,3,4] to capture WFS via a set of program transformations has been much discussed in the literature; and there have been various different attempts to extend WFS beyond the syntax of normal programs. One reason for proceeding via p-stable models is that, although they are defined with the help of program reducts, they are not too far removed conceptually from models in the usual sense of logical semantics. Thus we can hope to analyse them via standard logical and model-theoretic methods. Second, p-stable models are a generalization of ordinary (2-valued) stable models to a multi-valued setting; and, as noted, stable models do admit a natural logical foundation. Third, since the well-founded model of a normal logic program coincides with its unique least p-stable model, by capturing p-stable semantics for normal programs in terms of minimal models for some logic, a further minimization process will yield the well-founded model.

[1] See http://www.cs.sunysb.edu/ sbprolog/xsb-page.html

[2] We emphasise *non-modal* because there are well-known *embeddings* of say WFS into modal nonmonotonic logics, [22,1], but these modal logics have of course a richer logical syntax than that of the languages being embedded. Our method will be to work with ordinary logical connectives that correspond in a trivial way with the connectives used in logic programming.

[3] The concept of deductive base is defined and studied in [10,11]), see also the discussion in [18]. The main aspect to note here is that we obtain immediately the property that if programs, viewed as sets of formulas, are equivalent in this logic, then they are equivalent under these semantics.

Initial work on this problem was undertaken by Cabalar [5] who found a suitable concept of semantic frame that he called HT^2-frame, in view of the way it generalises the structure of ordinary HT-frames. A preliminary version of HT^2 axiomatics was presented in [6].

2 Partial Stable Models and Well-Founded Semantics

A central research topic in Logic Programming (LP) since its inception has been to provide a logical semantics for Prolog programs. In their seminal paper [14], van Endem and Kowalski showed that, if we set apart the control strategy of Prolog, the semantics of a (positive) logic program can be captured by the least Herbrand model of its set of rules, when these are interpreted as Horn clauses in classical logic. By a *positive* logic program we mean a set of rules of the form p :- $q_1, \ldots, q_n$ (being p called the *head* and $q_1, \ldots, q_n$ with $n \geq 0$ the *body*) and corresponding to the Horn clause $q_1 \wedge \cdots \wedge q_n \rightarrow p$. This foundational work was soon followed by different attempts to incorporate default negation in the rule bodies. Early efforts in this direction imposed syntactic restrictions on the set of rules (usually limiting cyclic dependences), and provided counterintuitive results when these restrictions were removed. The first satisfactory and widely accepted approach to dealing with default negation was the *stable model* semantics [15], leading to what is nowadays known as the *answer set programming* (ASP) paradigm.

Moving from the original Prolog semantics to ASP has, however, some important consequences: a program in ASP may have several models or even no model at all. Trying to get closer to the original "single model" orientation, van Gelder, Ross and Schlipf [24] introduced WFS, so that each program with negation had again a unique selected model (the *well-founded* model) but at the price of leaving some atoms undefined.

There exist several alternative descriptions of WFS, however, we will be particularly interested in Przymusinski's characteristation of WFS in terms of p-stable models [20,21], since they provide a less syntax-dependent semantic definition while they allow one to obtain the well-founded model by a simple minimisation criterion.

A *disjunctive logic program* is a set of rules like[4]:

$$p_1 \wedge \cdots \wedge p_n \wedge \neg q_1 \wedge \cdots \wedge \neg q_m \rightarrow r_1 \vee \cdots \vee r_h \tag{1}$$

with $n \geq 0$, $m \geq 0$ and $h > 0$ and all the q_i, r_j, p_k are atoms. The antecedent and consequent of (1) receive the names of rule *body* and *head*, respectively. When all rules satisfy $m = 0$, the program is said to be *positive disjunctive*, and just *positive* if all of them further satisfy $h = 1$. If the program has arbitrary m and n but $h = 1$ for all rules, then it is called a *normal* logic program.

[4] For simplicity sake, we use logical operators from the very beginning. Following the standard LP notation, rule (1) would be usually written as $r_1 \vee \cdots \vee r_k$:- $p_1, \ldots, p_n, \ldots,$ not $q_1, \ldots$ not q_m.

A *3-valued interpretation* $\mathbf{I}$ is a mapping from the propositional signature At to the set of truth values $\{0,1,2\}$ respectively standing for *false*, *undefined* and *true*. We can also represent the interpretation $\mathbf{I}$ as a pair of sets of atoms (I, I') satisfying $I \subseteq I'$ where $\mathbf{I}(p) = 0$ iff $p \not\in I'$, $\mathbf{I}(p) = 2$ iff $p \in I$ and $\mathbf{I}(p) = 1$ otherwise (ie, $p \in I' \setminus I$).

Two ordering relations among 3-valued interpretations are defined such that, if $\mathbf{I}_1 = (I_1, I_1')$ and $\mathbf{I}_2 = (I_2, I_2')$, then:

i) $\mathbf{I}_1 \leq \mathbf{I}_2$ iff $I_1 \subseteq I_2$ and $I_1' \subseteq I_2'$, ii) $\mathbf{I}_1 \preceq \mathbf{I}_2$ iff $I_1 \subseteq I_2$ and $I_2' \subseteq I_1'$.

The $\leq$ relation intuitively represents the situation that one interpretation contains "less truth" than the other. The other relation, $\preceq$, measures the degree of knowledge in terms of undefined atoms. Interpretations with shape (I, I) are called *complete* (they have no undefined atoms).

Given a 3-valued interpretation $\mathbf{I}$, Przymusinski's valuation of formulas is defined so that conjunction is the minimum, disjunction the maximum, negation is defined by the formula $\mathbf{I}(\neg\varphi) = 2 - \mathbf{I}(\varphi)$, and implication as:

$\mathbf{I}(\varphi \rightarrow \psi) = \begin{cases} 2 & \text{if } \mathbf{I}(\varphi) \leq \mathbf{I}(\psi) \\ 0 & \text{otherwise} \end{cases}$ We say that $\mathbf{I}$ is a *3-valued model* of a program Π, written $\mathbf{I} \models_3 \Pi$, when $\mathbf{I}(r) = 2$ for every rule $r \in \Pi$.

The definition of partial stable model relies on a generalisation of the program reduct [15] to the 3-valued case. Given a 3-valued interpretation $\mathbf{I}$, the *reduct* $\Pi^{\mathbf{I}}$ is formed by replacing each negated literal $\neg p$ in program Π by truth constants $\top, \mathbf{u}$ or $\bot$ depending on whether $\mathbf{I}(p)$ is $0, 1$ or 2 respectively. The valuation of new constants is fixed as $\mathbf{I}(\top) = 2$, $\mathbf{I}(\bot) = 0$ and $\mathbf{I}(\mathbf{u}) = 1$.

Definition 1 (Partial stable model). *A 3-valued interpretation* $\mathbf{I}$ *is a* partial stable model *of a disjunctive logic program* Π *if it is a* $\leq$*-minimal model of* $\Pi^{\mathbf{I}}$*.*

Well-founded models simply correspond to partial stable models with minimal information:

Definition 2 (Well-founded model). *A 3-valued interpretation* $\mathbf{I}$ *is a* well-founded model *of a disjunctive logic program* Π *if it is a* $\preceq$*-minimal partial stable model of* Π*.*

In [21] it is shown that, when Π is a normal logic program, the above definition of well-founded model corresponds to the original one in [24]. This means that any normal logic program has a $\preceq$-least p-stable model which corresponds to its well-founded model. Naturally, disjunctive programs may have several well-founded models (in fact, even no p-stable or well-founded model at all, as shown in [20]).

3 Combining Intuitionistic Connectives and Routley Negation

The logic that serves as a foundation for WFS and p-stable semantics is an extension of the logic N^*, which can be obtained via a direct combination of the standard semantics for intuitionistic positive connectives and the Routley semantics for negation operator [23]. Originally N^* was introduced in [6] as an extension of Došen's logic N [12], where the negation is treated as a kind of modal operator. We recall here the main definitions and facts regarding N^*. Formulas of N^* are built-up in the usual way using atoms from a given propositional signature At and the standard logical constants: $\wedge$, $\vee$, $\rightarrow$, $\neg$, respectively standing for conjunction, disjunction, implication and negation. We write For to stand for the set of all well-formed formulae of this language. The rules of inference for N are *modus ponens* and the anti-monotonicity rule for negation

$$\frac{\alpha \rightarrow \beta}{\neg\beta \rightarrow \neg\alpha} \tag{RC}$$

The set of axioms contains the axiom schemata of positive logic:

P1. $\alpha \rightarrow (\beta \rightarrow \alpha)$ **P2.** $(\alpha \wedge \beta) \rightarrow \alpha$
P3. $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$ **P4.** $(\alpha \wedge \beta) \rightarrow \beta$
P5. $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \gamma) \rightarrow (\alpha \rightarrow (\beta \wedge \gamma)))$ **P6.** $\alpha \rightarrow (\alpha \vee \beta)$
P7. $(\alpha \rightarrow \gamma) \rightarrow ((\beta \rightarrow \gamma) \rightarrow ((\alpha \vee \beta) \rightarrow \gamma))$ **P8.** $\beta \rightarrow (\alpha \vee \beta)$

together with the negation axioms:

N1. $\neg\alpha \wedge \neg\beta \rightarrow \neg(\alpha \vee \beta)$ **N2.** $\neg(\alpha \wedge \beta) \rightarrow \neg\alpha \vee \neg\beta$ **N3.** $\neg(\alpha \rightarrow \alpha) \rightarrow \beta$

The relation $\Gamma \vdash_{N^*} \varphi$, where Γ is a set of formulae and φ a formula, means that φ can be inferred from elements of Γ and theorems of N^* with the help of *modus ponens* only.

Definition 3 (N^* model). *A* frame *for N^* is a triple $\mathcal{W} = \langle W, \leq, * \rangle$ such that: (i) W is a non empty set (of worlds), (ii) $\leq$ is a partial ordering on W, (iii) $* : W \rightarrow W$ is a function such that $x \leq y$ iff $y^* \leq x^*$. An N^* model $\mathcal{M} = \langle W, \leq, *, V \rangle$ is an N^* frame $\mathcal{W} = \langle W, \leq, * \rangle$ augmented with a valuation function $V : At \times W \longrightarrow \{0, 1\}$ satisfying:*

$$V(p, u) = 1 \ \& \ u \leq w \ \Rightarrow \ V(p, w) = 1 \tag{2}$$

We say in this case that $\mathcal{M}$ is a model over *$\mathcal{W}$.*

V is extended to a valuation on all formulas via the following conditions.

- $V(\varphi \wedge \psi, w) = 1$ iff $V(\varphi, w) = V(\psi, w) = 1$
- $V(\varphi \vee \psi, w) = 1$ iff $V(\varphi, w) = 1$ or $V(\psi, w) = 1$
- $V(\varphi \rightarrow \psi, w) = 1$ iff for every w' $(w \leq w' \Rightarrow (V(\varphi, w') = 1 \Rightarrow V(\psi, w') = 1))$
- $V(\neg\varphi, w) = 1$ iff $V(\varphi, w^*) = 0$

It is easy to prove by induction that condition (2) above holds for any formula φ, ie

$$V(\varphi, u) = 1 \ \& \ u \leq w \Rightarrow V(\varphi, w) = 1. \tag{3}$$

Given a model $\mathcal{M} = \langle \mathcal{W}, V\rangle$ we write $\mathcal{M}, w \models \varphi$ or, simply $w \models \varphi$ instead of $V(\varphi, w) = 1$. A formula φ is *true* in an N^* model $\mathcal{M} = \langle W, \leq, *, V\rangle$, and we write $\mathcal{M} \models \varphi$, if $w \models \varphi$, for all $w \in W$. We say that φ is *true* in an N^* frame $\mathcal{W}$ if φ is true in every N^* model $\mathcal{M} = \langle \mathcal{W}, V\rangle$ over $\mathcal{W}$. We write in this case $\mathcal{W} \models \varphi$. And finally, a formula φ is *valid*, in symbols $\models_{N^*} \varphi$, if it is true in every N^* model (or, equivalently, in every N^* frame). The relation $\Gamma \models_{N^*} \varphi$, where Γ is a set of formulae, means that for every N^* model $\mathcal{M} = \langle W, \leq, *, V\rangle$ and $w \in W$ the condition $\forall \psi \in \Gamma (w \models \psi)$ implies $w \models \varphi$.

Theorem 1 (Strong comleteness for N^*). *For every set of formulae Γ and formula φ, the following equivalence holds:*

$$\Gamma \vdash_{N^*} \varphi \iff \Gamma \models_{N^*} \varphi.$$

Note that an intuitionistic negation can be defined in N^*. Fix some propositional variable p_0 and put

$$\bot := \neg(p_0 \rightarrow p_0) \text{ and } -\alpha := \alpha \rightarrow \bot.$$

In this way we obtain a definitional extension of N^*. Its $\langle \vee, \wedge, \rightarrow, -\rangle$-fragment coincides with intuitionistic logic.

4 HT^*-Models and HT^2-Models

First we consider an N^* extension determined by a two-element frame.

Definition 4 (HT^* model). *An HT^** model *is an N^* model $\mathcal{M} = \langle \mathcal{W}^{HT^2}, V\rangle$ over the frame $\mathcal{W}^{HT^2} = \langle \{h, t\}, \leq, *\rangle$ such that $h \leq t$, $h^* = t$, $t^* = h$.*

The relation $\models_{HT^*}$ is defined similar to $\models_{N^*}$, but on the class of HT^* models.

Every HT^* model $\mathcal{M} = \langle \mathcal{W}^{HT^*}, V\rangle$ can be identified with a 3-valued interpretation (H, T) consisting of sets of atoms respectively verified at worlds h and t, respectively. Indeed, by (2) we have $H \subseteq T$. The validity in HT^* models differs from the relation $\models_3$ (see Section 2), but for every disjunctive program Π, we have

$$(H, T) \models_3 \Pi \iff (H, T) \models \Pi,$$

which leads to the following conclusion.

Proposition 1. *A 3-valued interpretation (H, T) is a p-stable model of a disjunctive logic program Π iff it is a $\leq$-minimal HT^2 model of $\Pi^{(H,T)}$.*

To define p-stable models without using syntactic reducts we need more complicated N^* models.

Definition 5 (HT^2 model). *An HT^2* model *is an N^* model $\mathcal{M} = \langle \mathcal{W}^{HT^2}, V\rangle$ over the frame $\mathcal{W}^{HT^2}$ such that*

1. *W comprises 4 worlds denoted by h, h', t, t',*
2. *$\leq$ is a partial ordering on W satisfying $h \leq t$, $h \leq h'$, $h' \leq t'$ and $t \leq t'$,*
3. *$* : W \to W$ is given by $h^* = t^* = t'$, $(h')^* = (t')^* = t$,*
4. *V is an N-valuation.*

The ordering $\leq$ of the frame $\mathcal{W}^{HT^2}$ and the action of $*$ are represented in the following diagram.

$$\begin{tikzcd}
& t' \arrow[dl, leftrightarrow] & \\
t & & h' \arrow[ll] \\
& h \arrow[uu] &
\end{tikzcd}$$

Here $u < v$ iff v is strictly higher than u in the diagram.

Consider an HT^2 model $\mathcal{M} = \langle W, \leq, *, V\rangle$ denoting by H, H', T, T' the four sets of atoms respectively verified at each corresponding point or world h, h', t, t'. Since, by construction, $H \subseteq H'$ and $T \subseteq T'$, it is clear that we can represent $\mathcal{M}$ as a pair $\langle \mathbf{H}, \mathbf{T}\rangle$ of 3-valued interpretations $\mathbf{H} = (H, H')$ and $\mathbf{T} = (T, T')$.

An easy observation is that, by the semantics, if $(\mathbf{H}, \mathbf{T})$ is a model then necessarily $\mathbf{H} \leq \mathbf{T}$, since it is easy to check that this condition is equivalent to $H \subseteq T$ and $H' \subseteq T'$. Moreover, for any theory Π note that if $\langle \mathbf{H}, \mathbf{T}\rangle \models \Pi$ then also $\langle \mathbf{T}, \mathbf{T}\rangle \models \Pi$.

The ordering $\leq$ can be extended to a partial ordering $\trianglelefteq$ among models as follows. We set $\langle \mathbf{H}_1, \mathbf{T}_1\rangle \trianglelefteq \langle \mathbf{H}_2, \mathbf{T}_2\rangle$ if (i) $\mathbf{T}_1 = \mathbf{T}_2$; (ii) $\mathbf{H}_1 \leq \mathbf{H}_2$. A model $\langle \mathbf{H}, \mathbf{T}\rangle$ in which $\mathbf{H} = \mathbf{T}$ is said to be *total.* Note that the term *total* model does not refer to the absence of undefined atoms. To represent this, we further say that a total partial equilibrium model is *complete* if $\mathbf{T}$ has the form (T, T).

Definition 6 (Partial equilibrium model). *A model $\mathcal{M}$ of a theory Π is said to be a* partial equilibrium *model of Π if (i) $\mathcal{M}$ is total; (ii) $\mathcal{M}$ is minimal among models of Π under the ordering $\trianglelefteq$.*

Lemma 1. *For any disjunctive program Π and any HT^2 model $\mathcal{M} = \langle \mathbf{H}, \mathbf{T}\rangle$: $\mathcal{M} \models \Pi$ iff $\mathbf{H} \models \Pi^{\mathbf{T}}$ and $\mathbf{T} \models \Pi$ (here $\mathbf{H}$ and $\mathbf{T}$ are considered as HT^* models).*

The proof of this lemma easily follows from the definition of HT^2 model. Combining this statement and Proposition 1 we obain.

Theorem 2 (main correspondence result). *A total HT^2 model $\langle \mathbf{T}, \mathbf{T}\rangle$ is a partial equilibrium model of a disjunctive program Π iff the 3-valued interpretation $\mathbf{T}$ is a p-stable model of Π.*

Once partial stable models are captured, we can redefine well-founded models by just minimising the information of partial equilibrium models, with a simple rephrasing of Przymusinski's definition for total HT^2 interpretations.

Definition 7 (Well-founded, partial equilibrium model). *A partial equilibrium model $\langle \mathbf{T}, \mathbf{T}\rangle$ of a theory Π is further said to be* well-founded *if there is no partial equilibrium model $\langle \mathbf{T}_2, \mathbf{T}_2\rangle$ of Π such that $\mathbf{T}_2 \prec \mathbf{T}$.*

In this way, for disjunctive logic programs, well-founded partial equilibrium models trivially correspond to well-founded models in Przymusinski's sense and therefore, for the case of normal programs, to the standard definition of WFS.

5 Axiomatisation of HT^2

Up to now we have only defined the set of HT^2 tautoligies and the relation $\models_{HT^2}$. Now we define this logic syntactically. The axiomatic system given below differs in an essential way from that of [6] in that it does not include additional inference rules. We also obtain a new and more intuitive list of axioms, quite different from that of [6]. Finally, we prove here the strong completeness theorem. Let HT^2 be an N^* extension obtained by adding the following axioms:

A1. $\alpha \vee (\alpha \to (\beta \vee (\beta \to (\gamma \vee -\gamma))))$

A2. $\alpha \to \neg\neg\alpha$

A3. $\alpha \wedge \neg\alpha \to \neg\beta \vee \neg\neg\beta$

A4. $\alpha \wedge \neg\alpha \to \beta \vee (\beta \to \gamma) \vee -\gamma$

A5. $\neg\neg(\beta \vee (\beta \to \gamma) \vee -\gamma)$

A6. $\neg\neg\alpha \wedge \neg\neg\beta \to (\alpha \to \beta) \vee (\beta \to \alpha)$.

Theorem 3. *For every set of formulae Γ and formula φ, the following equivalence holds:*

$$\Gamma \vdash_{HT^2} \varphi \iff \Gamma \models_{HT^2} \varphi.$$

The axiomatics of HT^* can be obtained from that of N^* by adding the axioms:

$$\alpha \vee (\alpha \to \beta) \vee -\beta, \quad \alpha \leftrightarrow \neg\neg\alpha, \quad \alpha \wedge \neg\alpha \to \neg\beta \vee \neg\neg\beta.$$

6 Strong Equivalence wrt Partial Equilibrium Logic

The concept of strong equivalence [16] captures the idea that two theories or logic programs are equivalent in any context. This notion is important both conceptually and as a potential tool for simplifying nonmonotonic programs and theories and optimising their computation. For stable semantics, strong equivalence can be completely captured in the logic HT [16] and in ASP this fact has given rise to a programme of research into defining and computing different equivalence concepts, see eg [13,25]. In the case of WFS and p-stable semantics, it was proved that the strong equivalence of arbitrary theories wrt partial equilibrium logic can be captured in HT^2 [6].

Returning to arbitrary theories, formally in the present context:

Definition 8. *Two theories Γ_1 and Γ_2 are said to be* strongly equivalent *if for any theory Γ, theories $\Gamma_1 \cup \Gamma$ and $\Gamma_2 \cup \Gamma$ have the same partial equilibrium models.*

Theorem 4. [6] *Theories Γ_1 and Γ_2 are strongly equivalent iff Γ_1 and Γ_2 are equivalent in HT^2.*

The above result can be extended to show that HT^2 also captures strong equivalence wrt well-founded models (ie, $\preceq$-minimal partial equilibrium models). Two HT^2 theories Γ_1 and Γ_2 are called *WF equivalent* if for any HT^2 theory Γ, each well founded model of $\Gamma_1 \cup \Gamma$ is a well founded model of $\Gamma_2 \cup \Gamma$ and vice versa.

Theorem 5. *Theories Γ_1 and Γ_2 are WF equivalent iff Γ_1 and Γ_2 are equivalent as HT^2 theories.*

Note that unlike in the case of strong equivalence under stable model semantics, we cannot assume in the general case that the formulas in Γ have the syntax of logic program rules. So when Γ_1 and Γ_2 have the form of logic programs, it is clear that HT^2 equivalence is a sufficient condition for strong equivalence, but it is an open question whether Γ can be taken to be a logic program (of whatever kind) in the case of non-equivalence.

7 Conclusions

We have presented a logical foundation for the partial stable model and well-founded semantics of logic programs. Our approach has been to identify an underlying monotonic logical framework to be used as a basis. The natural choice is a logic in which partial stability can be expressed as a simple minimality condition with well-foundedness as a special case. The condition of equilibrium that captures stable models in the logic HT of here-and-there is generalised to a minimality condition that captures partial stability in a logic HT^2 which corresponds in a natural way to HT. In this paper we have shown how the resulting logic can be axiomatised as an extension of the logic N^* with Routley negation. We have also showed that HT^2 captures the strong equivalence of theories in partial equilibrium logic.

In this paper we have restricted attention to some of the more basic and challenging issues such as finding an axiomatic system for the logic HT^2 and establishing results on strong equivalence. Further work on partial equilibrium logic has been reported elsewhere. In particular, in [9] the following topics are covered:

– the complexity of reasoning with partial equilibrium logic;
– the behaviour of partial equilibrium logic on disjunctive and nested logic programs and its comparison with other semantics;
– further study of the relation of HT^2 to HT and of partial equilibrium logic to equilibrium logic;
– general properties of partial equilibrium entailment;
– other kinds of strong equivalence results for special classes of models;
– proof theory and implementation methods for partial equilibrium logic.

In addition, in [7] studies ways to add strong or explicit negation to partial equilibrium logic and compares this with the well-known system WFSX with explicit negation [19].

References

1. Bonatti, P.: Autoepistemic logics as a unifying framework for the semantic of logic programs. Journal of Logic Programming 22(2), 91–149 (1995)
2. Brass, S., Dix, J.: A disjunctive semantics based on unfolding and bottom-up evaluation. In: IFIP 1994 Congress, Workshop FG2: Disjunctive Logic Programming and Disjunctive Databases, pp. 83–91 (1994)
3. Brass, S., Dix, J.: Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. Journal of Logic Programming 30, 207–228 (1997)
4. Brass, S., Dix, J.: Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. Journal of Automated Reasoning 20, 143–165 (1998)
5. Cabalar, P.: Well-founded semantics as two-dimensional here-and-there. In: Proceedings of ASP 2001, 2001 AAAI Spring Symposium Series (2001)
6. Cabalar, P., Odintsov, S., Pearce, D.: Logical Foundations of Well-Founded Semantics. In: Proceedings KR 2006, pp. 25–35. AAAI Press, Menlo Park (2006)
7. Cabalar, P., Odintsov, S., Pearce, D.: Strong Negation in Well-Founded and Partial Stable Semantics for Logic Programs. In: Sichman, J.S., Coelho, H., Rezende, S.O. (eds.) IBERAMIA 2006 and SBIA 2006. LNCS, vol. 4140, pp. 592–601. Springer, Heidelberg (2006)
8. Cabalar, P., Odintsov, S., Pearce, D., Valverde, A.: Analysing and Extending Well-Founded and Partial Stable Semantics using Partial Equilibrium Logic. In: Etalle, S., Truszczyński, M. (eds.) ICLP 2006. LNCS, vol. 4079, pp. 346–360. Springer, Heidelberg (2006)
9. Cabalar, P., Odintsov, S., Pearce, D., Valverde, A.: Partial Equilibrium Logic. Annals of Mathematics and Artificial Intelligence 50, 305–331 (2007)
10. Dietrich, J.: Deductive bases of nonmonotonic inference operations. Technical report, NTZ Report, Universität Leipzig (1994)
11. Dietrich, J.: Inferenzframes. Doctoral Dissertaion, Universität Leipzig (1995)
12. Došen, K.: Negation as a modal operator. Rep. Math. Logic 20, 15–28 (1986)
13. Eiter, T., Fink, M., Woltran, S.: Semantical Characterizations and Complexity of Equivalences in Answer Set Programming. ACM Transactions on Computational Logic, 8 (2007)
14. van Endem, M., Kowalski, R.A.: The Semantics of Predicate Logic as a Programming Language. Journal of the ACM 23, 733–742 (1976)
15. Gelfond, M., Lifschitz, V.: The stable models semantics for logic programming. In: Proc. of the 5th Intl. Conf. on Logic Programming, pp. 1070–1080 (1988)
16. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic 2, 526–541 (2001)
17. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Dix, J., Przymusinski, T.C., Moniz Pereira, L. (eds.) NMELP 1996. LNCS, vol. 1216, pp. 57–70. Springer, Heidelberg (1997)
18. Pearce, D.: Equilibrium Logic. Annals of Mathematics and Artificial Intelligence 47, 3–41 (2006)
19. Pereira, L.M., Alferes, J.J.: Well founded semantics for logic programs with explicit negation. In: Proceedings of the European Conference on Artificial Intelligence (ECAI 1992), pp. 102–106. John Wiley & Sons, Montreal (1992)
20. Przymusinski, T.: Stable semantics for disjunctive programs. New Generation Computing 9, 401–424 (1991)

21. Przymusinski, T.: Well-founded and stationary models of logic programs. Annals of Mathematics and Artificial Intelligence 12, 141–187 (1994)
22. Przymusinski, T.: Static semantics for normal and disjunctive logic programs. Annals of Mathematics and Artificial Intelligence 14, 323–357 (1995)
23. Routley, R., Routley, V.: The semantics of first degree entailment. Noûs 6, 335–359 (1972)
24. van Gelder, A., Ross, K.A., Schlipf, J.S.: The well-founded semantics for general logic programs. Journal of the ACM 38(3), 620–650 (1991)
25. Woltran, S.: Characterizations for relativized notions of equivalence in answer set programming. In: Alferes, J.J., Leite, J. (eds.) JELIA 2004. LNCS, vol. 3229, pp. 161–173. Springer, Heidelberg (2004)

The Reachability Problem over Infinite Graphs

Wolfgang Thomas

RWTH Aachen University, Lehrstuhl Informatik 7, 52056 Aachen, Germany
thomas@informatik.rwth-aachen.de

Abstract. We survey classical and selected recent work on the reachability problem over finitely presented infinite graphs. The problem has a history of 100 years, and it is central for automatic verification of infinite-state systems. Our focus is on graphs that are presented in terms of word or tree rewriting systems.

1 Historical Introduction

In a rather unknown paper of 1910, titled *Die Lösung eines Spezialfalles eines allgemeinen logischen Problems* [20], Axel Thue introduced trees as representations of terms, tree rewriting rules, and the problem of deciding whether from a given term one can reach another given term by a finite number of rewrite steps. He treats a "special case" of this "general logical problem" since he is doubtful about solvability in general. His prophetical statement on the matter is: "Eine Lösung dieser Aufgabe im allgemeinsten Falle dürfte vielleicht mit unüberwindlichen Schwierigkeiten verbunden sein" ("A solution of this problem in the most general case may perhaps be connected with unsurmountable difficulties"). Thue's problem can be stated as the reachability problem over an infinite directed graph; this graph has terms as vertices, and the edge relation is defined by term rewriting rules.

Thue's comment marks the beginning of a long and as yet unfinished track of research – aiming at finding infinite graphs over which the reachability problem "Given vertices u, v, is there a path from u to v?" is decidable. A central motivation today is infinite-state system verification. Another motivation is more general, to develop an algorithmic theory of infinite models. In this context, the reachability problem is the most basic of a family of model-checking problems, also covering several logics in which reachability can be expressed.

The first undecidability results (in fact, showing also undecidability of Thue's problem) follow from the fundamental work of Turing and Post in the 1930's and 1940's. A Turing machine defines an infinite graph consisting of the words that represent Turing machine configurations and where an edge corresponds to a step from one configuration to the next. Assuming a unique acceptance configuration s, and considering a Turing machine that accepts a non-recursive (but recursively enumerable) language, the reachability problem "Is there a path from the initial configuration for the input word w to the configuration s?" is undecidable.

The configuration graph of a Turing machine can be presented with a regular set of words (configurations) as vertex set and an infix rewriting system defining the edge relation. As Post showed (see e.g. [16]), the reachability problem stays undecidable if instead one uses an edge relation defined by a coordinated prefix-suffix rewriting. Here (namely, in the normal form of Post's canonical systems) a rule consists of deleting a certain prefix of a word while adding a certain suffix.

There are several approaches for detecting classes of graphs where the reachability problem is decidable. An important branch of research has its focus on models that share certain monotonicity properties and are often called "well-structured" (see e.g. [10]). Petri nets and lossy channel systems are prominent cases of this type where reachability was shown decidable.

Here we pursue another approach, where different forms of word and tree rewriting (or corresponding automata theoretic concepts) are used for the presentation of graphs. The reachability problem serves here as a core problem in an algorithmic model theory (of graphs). As an example for the automata theoretic presentation of graphs we mention the *automatic graphs* (see [3]): Here the vertex set is given by a regular language (over a suitable alphabet), and the set of word pairs (u, v) that belong to the edge relation are defined by an automaton that scans u and v synchronously, letter by letter.

It is easy to see that both the infix rewriting and the coordinated prefix-suffix rewriting as mentioned above (for Turing machines and Post's canonical systems, respectively) lead to automatic graphs. The first-order theory of an automatic graph is decidable, but – as seen above – an extension of first-order logic by the reachability relation leads to an undecidable theory. So the leading question for the sequel has to focus on subclasses of the class of automatic graphs.

2 Variants of the Reachability Problem

We consider directed graphs in the format $G = (V, E)$ with vertex set V and edge relation E, and in the format $G = (V, (E_a)_{a \in \Sigma})$ with labelled edges (where E_a contains the edges labelled with a). Let us list a number of reachability problems over such graphs G.

1. *Plain reachability:* Given u and a (finite representation of a) set $T \subseteq V$ of "target vertices", is there a path from u to some vertex of T? (A special case deals with singleton sets $T = \{v\}$.)
2. *Termination:* Given u and a set $T \subseteq V$, does each path from u eventually meet T?
3. *Alternating reachability:* Here we consider a game where two players $1, 2$ choose edges in turn, thus building up a path: Given u and a set $T \subseteq V$, is there a strategy (say for Player 2) to build a path from u that reaches T?
4. *FO(E^*) model-checking:* Here we refer to the expansion of graphs to structures $G' = (V, E, E^*)$ and ask whether the first-order theory of G' is decidable.
5. *FO(Reg) model-checking:* This is defined like FO(E^*), but we refer to edge labeled graphs and the operators E_r with a regular expression r rather than

$E^* - E_r(x, y)$ meaning that there is a path from x to y labeled with a word that satisfies r.

6. *MSO model-checking:* Is the monadic second-order theory of G is decidable? (Note that MSO-logic allows to express, e.g., plain reachability of T from u by the formula saying that each set X containing u and closed under E intersects T.)

3 Prefix Rewriting

Consider a graph $G = (V, (E_a)_{a \in \Sigma})$ where V consists of words $qw \in Q \cdot \Gamma^*$ that represent configurations (control state, stack content) of a pushdown automaton, and where E_a consists of those configuration pairs that are induced by an a-transition of the pushdown automaton. Thus one calls G a "pushdown graph". It is known (e.g., from Büchi's analysis of "regular canonical systems" [4]) that the configurations reachable from an initial configuration $q_0\gamma_0$ of a pushdown automaton form a regular set whose representation (e.g., by a finite automaton) can be computed from the pushdown automaton. More precisely, and more abstractly: Given a prefix rewriting system S over the alphabet Γ, if $W \subseteq \Gamma^*$ is regular, then also the set $\text{post}^*(W)$ of words reachable from words in W using rules from S is again regular. The modern proof of this result is done by the "saturation method" ([2,9]), which basically extends a given finite automaton (recognizing W) by extra transitions to a new finite automaton that recognizes $\text{post}^*(W)$.

This basic result has been extended in several ways. In this section we discuss two extensions that adhere to the idea of prefix rewriting. In the next section other extensions are treated that involve more general mechanisms of rewriting.

First, one can generalize the prefix rewriting rules $u \to v$ (with words u, v) to rules $U \to V$ with regular sets U, V of words instead. An application as a prefix rewrite rule is a step from a word uw to a word vw where $u \in U, v \in V$. The graphs generated by these generalized prefix rewriting systems are called "prefix recognizable" ([5]); in contrast to pushdown graphs they may have vertices of infinite degree.

The second generalization rests on the idea of nested stacks, leading to the so-called higher-order pushdown automata and their associated configuration graphs. We do not give definitions here (see [6,7]) but just note that for each k one can introduce "level-k pushdown automata" that work over nested stacks of level k; here a standard stack is of level 1, a stack of stacks is of level 2, etc. It is known that a highly complex landscape of (level-k pushdown-) graphs is generated in this way. The two generalizations may also be merged by allowing "regular rules" in the definition of higher-order pushdown automata.

It is interesting to note that for the basic model of pushdown graphs, and for all generalizations of it mentioned above, a much stronger decidability result than that for reachability can be shown: In all these cases of graphs, the MSO-theory is decidable.

4 Ground Tree Rewriting, Mixed Prefix-Suffix Rewriting

The MSO-theory of a graph is of interest in connection to automata theory. Typical sentences expressible in MSO-logic formulate the existence of a "run" (or a "coloring") of some device over the considered structure. However, in verification other properties are more significant; they are related to existence of paths rather than the existence of colorings. It is therefore useful to consider structures that fall outside the general domain of prefix rewriting but nevertheless admit a decision procedure e.g. for plain reachability.

A first natural approach is the idea of ground tree rewriting. We are given a finite tree rewriting system, and the application of a rule $t \to t'$ to a tree s means to replace a subtree t of s by t'. In term rewriting, t, t' are called "ground terms". Considering ground terms as "prefixes" of terms, we can still speak of a kind of "prefix rewriting". We can use these conventions to define naturally the edge relation of a graph whose vertices are trees. In this way we build the so-called ground tree rewriting graphs. The concept of regular tree language can be invoked to single out vertex sets of graphs (as needed for the specification of instances of the reachability problem).

A simple example shows that these graphs go beyond the graphs obtained by prefix rewriting. Using the initial term (= tree) $f(c, d)$ and the two rules $c \to g(c)$ and $d \to g(d)$ we obtain a copy of the infinite grid, with vertices shortly denoted as $f(g^i(c), g^j(d))$. Since it is well-known that the MSO-theory of the infinite grid is undecidable, the MSO model-checking problem over ground tree rewriting graphs is in general undecidable. However, it turns out that the saturation method can be applied again for trees rather than words; so the plain reachability problem over ground tree rewriting graphs is decidable. Even the FO(E^*)-theory of a ground tree rewriting graph is decidable ([8]). An analysis by Löding [14] showed that slightly more ambitious reachability problems are undecidable, namely the termination problem, the alternating reachability problem, and the FO(Reg)-theory. For the extension of these results to graphs that arise from unranked trees (where there is no uniform bound on the branching index) see [15].

The second approach to be discussed here refers to words, but involves rewriting of either prefixes and suffixes. The difference to Post's canonical systems is the lack of coordination between applying prefix rewriting rules and suffix rewriting rules; the order in which they are applied is arbitrary. Not only is the reachability problem decidable over the corresponding graphs; the reachability relation between words turns out to be rational [11,12]. The same is true when the rules are regular (i.e., involving regular sets rather than single words), see [1]. Finally, all the undecidability results mentioned above for ground tree rewriting graphs also hold for graphs generated by mixed prefix-suffix rewriting.

Despite this correspondence of results for the two types of graphs, there are clearly intuitive differences between them. With the tree structure one can realize a dynamic sequence of stacks; see the left of Figure 1 below where two stacks are seen (the top positions being the leafs), and where an additional stack can be opened at the rightmost leaf. Only for two stacks this can directly be realized also over words, using mixed prefix-suffix rewriting; see the right of the figure where the symbol Z serves as a separator. On the other hand, mixed prefix-suffix rewriting allows to hand over information say from left to right (by adding symbols on the left and deleting symbols on the right), which corresponds to a migration of information from the bottom of one stack to the bottom of the other. This migration is not directly possible in the representation with ground tree rewriting rules.

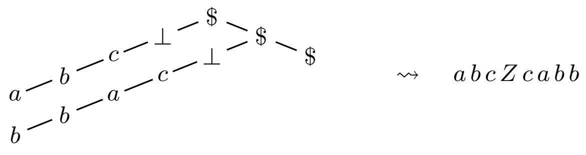


Fig. 1. Two realizations of a pair of stacks

These differences can be fixed also in a more precise sense, by showing that the two classes of ground tree rewriting graphs and mixed prefix-suffix rewriting graphs are indeed incompatible (w.r.t inclusion). First, ground tree rewriting can generate graphs with finite but unbounded degree (just consider the graph generated from $f(d, d)$ by the rule $d \to f(d, d)$). Secondly, we mentioned that the FO(E^*)-theory of a ground tree rewriting graph is decidable whereas it turns out to be undecidable for mixed prefix-suffix rewriting graphs. The latter claim is verified using a simple rewriting system with two rules, one that allows to delete a letter on the left, and one that allows to delete a letter on the right. The transitive closure of the rewrite relation E is $E^* = \{(u, v) \mid u \text{ has infix } v\}$. Now one can apply a result of Quine [18] that the first-order theory of the structure (Γ^*, is infix of) is undecidable.

5 Concluding Remarks

These concepts and examples (presented here with a bias to work done in the author's group) are a first step into a widely open field that amounts to new kind of model theory in which algorithmic results are central and where concepts of automata theory are useful.

In this overview we concentrated on the study of graphs and the reachability problem (in different variants). Let us end by mentioning two related aspects that suggest further tracks of investigation.

It would be nice to have structural characterizations of the graph classes discussed above. For a single class a very elegant characterization is known, namely for the pushdown graphs. A theorem of Muller and Schupp [17] characterizes these graphs as the directed graphs G with an origin v_0 and with bounded degree which have only "finitely many end types". An "end" of G is here a connected graph H that is obtained as follows: Delete, for some $n \geq 0$, all vertices of distance $\leq n$ from v_0 and let H be a connected component of the remaining graph. An end type is an isomorphism type of an end H. – For other types of graphs, in particular for the automatic graphs, such a description is lacking.

A more modest method to classify graphs is to compare their recognition power as acceptors of languages. In this case we deal with labelled graphs $G = (V, (E_a)_{a \in \Sigma})$ which are viewed as (possibly infinite) automata with labelled transitions. Assuming that the "state set" V is represented by a regular language (say of words), one introduces two further regular sets $I, F \subseteq V$ and declares a word w accepted if there is a w-labelled path from I to F (see [19]). Under these conventions, it is known that pushdown graphs with ε-transitions recognize precisely the context-free languages and automatic graphs precisely the context-sensitive languages. For the languages recognized by ground tree rewriting graphs and for mixed prefix-suffix rewriting graphs, only partial results are known (see [13]).

References

1. Altenbernd, J.: On bifix systems and generalizations. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 40–51. Springer, Heidelberg (2008)
2. Bouajjani, A., Esparza, J., Maler, O.: Reachability Analysis of Pushdown Automata: Application to Model-Checking. In: Mazurkiewicz, A., Winkowski, J. (eds.) CONCUR 1997. LNCS, vol. 1243, pp. 135–150. Springer, Heidelberg (1997)
3. Blumensath, A., Grädel, E.: Finite Presentations of Infinite Structures: Automata and Interpretations. Theory of Computing Systems 37, 641–674 (2004)
4. Büchi, J.R.: Finite Automata, Their Algebras and Grammars. In: Siefkes, D. (ed.). Springer, New York (1989)
5. Caucal, D.: On the Regular Structure of Prefix Rewriting. Theor. Comput. Sci. 106, 61–86 (1992)
6. Caucal, D.: On infinite graphs having a decidable monadic theory. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420, pp. 165–176. Springer, Heidelberg (2002)
7. Carayol, A., Wöhrle, S.: The Caucal hierarchy of infinite graphs in terms of logic and higher-order pushdown automata. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 112–123. Springer, Heidelberg (2003)
8. Dauchet, M., Tison, S., Heuillard, T., Lescanne, P.: Decidability of the Confluence of Ground Term Rewriting Systems. In: Proc. LICS 1987, pp. 353–359 (1987)
9. Esparza, J., Hansel, D., Rossmanith, P., Schwoon, S.: Efficient Algorithms for Model Checking Pushdown Systems. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 232–247. Springer, Heidelberg (2000)
10. Finkel, A., Schnoebelen, P.: Well-structured transition systems everywhere! Theor. Comput. Sci. 256, 63–92 (2001)
11. Karhumäki, J., Kunc, M., Okhotin, A.: Communication of Two Stacks and Rewriting. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006, Part II. LNCS, vol. 4052, pp. 468–479. Springer, Heidelberg (2006)
12. Karhumäki, J., Kunc, M., Okhotin, A.: Computing by commuting. Theor. Comput. Sci. 356, 200–211 (2006)
13. Löding, C.: Infinite graphs generated by tree rewriting. PhD thesis, RWTH Aachen, Germany (2003)

14. Löding, C.: Reachability problems on regular ground tree rewriting graphs. Theory of Computing Systems 39, 347–383 (2006)
15. Löding, C., Spelten, A.: Transition Graphs of Rewriting Systems over Unranked Trees. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 67–77. Springer, Heidelberg (2007)
16. Minsky, M.: Computation: Finite and Infinite Machines. Prentice-Hall, NJ (1967)
17. Muller, D.E., Schupp, P.E.: The theory of ends, pushdown automata, and second-order logic. Theor. Comput. Sci. 37, 51–75 (1985)
18. Quine, W.V.: Concatenation as a Basis for Arithmetic. J. Symb. Logic 11, 105–114 (1946)
19. Thomas, W.: A short introduction to infinite automata. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 130–144. Springer, Heidelberg (2002)
20. Thue, A.: Die Lösung eines Spezialfalles eines allgemeinen logischen Problems, Kra. Videnskabs-Selskabets Skrifter, I. Mat. Nat. Kl. 1910, No. 8, Kristiania (1910); reprinted in: Nagel, T., et. al (eds.) Selected Mathematical Papers of Axel Thue. Universitetsforlaget, Oslo (1977)

Kolmogorov Complexity and Model Selection

Nikolay Vereshchagin*

Moscow State University, Leninskie gory 1,
Moscow 119991, Russia
ver@mccme.ru
http://lpcs.math.msu.su/~ver

1 Stochastic Strings

The goal of statistics is to provide explanations (models) of observed data. We are given some data and have to infer a plausible probabilistic hypothesis explaining it. Consider, for example, the following scenario. We are given a "black box". We have turned the box on (only once) and it has produced a sequence x of million bits. Given x, we have to infer a hypothesis about the black box.

Classical mathematical statistics does not study this question. It considers only the case when we are given results of many independent tests of the box. However, in the real life, there are experiments that cannot be repeated. In some such cases the common sense does provide a reasonable explanation of x. Here are three examples: (1) The black box has printed million zeros. In this case we probably would say that the box is able to produce only zeros. (2) The box has produced a sequence without any regularities. In this case we would say that the box produces million independent random bits. (3) The first half of the sequence consists of zeros and the second half has no regularities. In this case we would say that the box produces 500000 zeros and then 500000 independent random bits.

Let us try to understand the mechanism of such common sense reasoning. First, we can observe that in each of the three cases we have inferred a finite set A including x. In the first case, A consists of x only. In the second case, A consists of all sequences of length million. In the third case, the set includes all sequences whose first half consists of only zeros. Second, in all the three cases the set A can be described in few number of bits. That is A has low Kolmogorov complexity.[1] Third, all regularities present in x are shared by all other elements of A. That is, x is a "typical element of A".

It seems that the common sense reasoning works as follows: given a string x of n bits we find a finite set A of strings of length n containing x such that

(1) A has low Kolmogorov complexity (we are interested in simple explanations) and

* The work was in part supported by a RFBR grant 09-01-00709.

[1] Roughly speaking, the Kolmogorov complexity of a string x is the length of a shortest program printing x. The Kolmogorov complexity of a finite set A is defined as the Kolmogorov complexity of the list of all elements of A in some fixed order, say, in the lexicographical one. For a rigorous definition we refer to [2,4].

(2) x is a typical member of A, that is x has no regularities which allow to single out x from A.

How to define rigorously what means that x is a typical element of A? To this end we use the notion of the randomness deficiency [5]:

$$d(x|A) = \log_2 |A| - C(x|A).$$

Here $C(x|A)$ stands for the conditional Kolmogorov complexity of x given the list of A[2]. The randomness deficiency has the following properties: $d(x|A)$ is non-negative for all $x \in A$ (up to a $O(\log n)$ error term) and for every finite A for almost all $x \in A$, $d(x|A)$ is negligible. Thus "random" elements of A have low randomness deficiency in A. We call x a "typical member of A" if $d(x|A)$ is small.

Strings that have explanations A with properties (1) and (2) are called *stochastic*. The first question, raised by Kolmogorov in 1983, was whether all strings are stochastic. Formally, a string x is called *α, β-stochastic* (where α, β are natural parameters) if there is a set $A \ni x$ of Kolmogorov complexity at most α such that $d(x|A) \leq \beta$. It turns out that some strings have no explanation: they are not α, β-stochastic for α and β proportional to n.

Theorem 1. *There is a constant c such that for all large enough n the following holds for all α, β. If $\alpha + \beta < n - c \log n$, then there is a string x of length n that is not α, β-stochastic. On the other hand, if $\alpha + \beta > n + c \log n$ then all strings of length n are α, β-stochastic (which is obvious).*

This theorem was first proved, in a weaker form, in [5] (with condition $2\alpha + \beta < n - c \log n$ in place of $\alpha + \beta < n - c \log n$). In the present form the theorem appeared in [7].

Note that we consider only uniform distributions on finite sets as possible probabilistic hypotheses. It is not hard to show that general distributions can be reduced to uniform ones [7].

2 Hypotheses Selection

The second question is the following: assume that x is α, β-stochastic for some small α, β. How do we find a set $A \ni x$ with small $C(A)$ and $d(x|A)$? Obviously we look for sets $A \ni x$ of low complexity. To see that a set A has low complexity we somehow find a short description of A. But how can we verify that $d(x|A)$ is small? We can only verify that $d(x|A)$ is large by describing x conditional to A in much fewer than $\log |A|$ bits. That is, we can refute (2) and not prove it.

It seems that instead of verifying that $d(x|A)$ is small we do what we are able: we try to refute that. If no such refutation is found for a long time, then it becomes plausible that $d(x|A)$ is indeed small. On the other hand, assume that we have found a "constructive refutation", that is, an easily described property

[2] Roughly speaking, the conditional Kolmogorov complexity of a string x given a string y is the length of a shortest program that prints x given y as an input.

P of elements of A such that x has the property P but the majority of elements of A do not. In this case we can switch to a new hypothesis $A' = \{x \in A \mid P(x)\}$. We then have $C(A') \approx C(A)$ (as P has a simple description) and $|A'| \ll |A|$ (as the majority of elements of A do not satisfy P). Therefore

$$d(x|A') = \log|A'| - C(x|A')$$

is much less than

$$\log|A| - C(x|A') \le \log|A| - C(x|A) + C(P) = d(x|A) + C(P) \approx d(x|A)$$

(the first inequality holds up to an $O(\log n)$ error term). Here $C(P)$ stands for the Kolmogorov complexity of P, which is assumed to be negligible. Thus A' is much better than A as an explanation of x.

Actually, if by any means, in our search for explanations of x, we have found a hypothesis $A' \ni x$ with lower (or equal) complexity than the current explanation A and such that $\log|A'|$ is significantly smaller than $\log|A|$, we usually switch to such A'. This strategy is essentially based on the Maximal Likelihood (ML) principle from classical statistics. Recall that ML estimator chooses a distribution μ that maximises $\mu(x)$ (among all contemplated probability distributions). In the case of uniform probability distributions, the probability that x is obtained by picking a random element of A is equal to $1/|A|$. Thus maximising $\mu(x)$ corresponds to minimising $|A|$.

So assume that we just look for an explanation that minimises $|A|$ among all simple explanations. Do we finally obtain a hypothesis with small randomness deficiency? More specifically, let $\mathrm{ML}_x(\alpha)$ stand for a set A that minimises $|A|$ among all $A \ni x$ of Kolmogorov complexity at most α. Is it true that

$$d(x|\mathrm{ML}_x(\alpha)) \le \beta + O(\log n)$$

for all α, β-stochastic x? Below we will show that this is indeed the case.

Let us see what explanations would we infer using the ML strategy in the examples (1)–(3) from the beginning of the paper. In the first example we would certainly choose the explanation $A = \{x\}$. In the second and third examples it depends on the complexity level α. If $\alpha = 100000$, say, then the ML strategy could choose the set A consisting of all sequences having the same prefix of length α as x has (in the second example) and the set A consisting of all sequences having the same prefix of length $\alpha + 500000$ as x has (in the third example). If α is very small, say $\alpha = 0$, then there will be no explanations at all of complexity at most α. For some small α the ML strategy might find the explanations obtained by common sense reasoning. However we do not know the right value α in advance.

We see that sometimes we prefer an explanation A' to an explanation A' even if $\log|A'| \gg \log|A|$ (the explanation A' is more general than A). This happens only when $C(A') \ll C(A)$. How do we compare hypotheses of different complexity? It seems that we use the Minimum Description Length principle (MDL). We prefer that hypothesis A for which $C(A) + \log|A|$ is smaller. And among two hypotheses with the same value of $C(A)+\log|A|$ we prefer the simpler one. The explanation of the term MDL is the following: the pair (the shortest description A^* of A, the index i of x in the list of all elements of A) is a two-part description of x. The total length of this description is $C(A)+\log|A|$. The minimal possible value for $C(A)+\log|A|$ is obviously $C(x)$ (the Kolmogorov complexity of x). Those A with $C(A)+\log|A| \approx C(x)$ are called *sufficient statistics of x*.

In the above examples (1)–(3), the common sense explanations are sufficient statistics of minimal complexity. Such sufficient statistics are called *minimal*.

If A is a sufficient statistics of x then $d(x|A)$ is negligible, as

$$d(x|A) = \log|A| - C(x|A) \leq \log|A| + C(A) - C(x) + O(\log n). \tag{1}$$

Note that sufficient statistics always exist, which is witnessed by $A = \{x\}$. Thus MDL based search always returns in the limit a hypothesis with negligible randomness deficiency.

However we are interested in *simple* explanations and not only in those having negligible randomness deficiency. If there is a simple sufficient statistic, then the MDL based search will find such statistic in the limit. But is there always such statistic provided that x is α,β-stochastic?

2.1 The Case of Small α

If α is small then, obviously, the question answers in positive. Indeed, let A be a set witnessing α,β-stochasticity of x? Then

$$\log|A| + C(A) \leq \log|A| + \alpha \leq C(x|A) + \beta + \alpha \leq C(x) + \beta + \alpha \tag{2}$$

(the last inequality holds up to an $O(1)$ error term). Thus A itself is a sufficient statistic (we assume that β is small, too). Besides, A witnesses that $\mathrm{ML}_x(\alpha) \leq \log|A|$, which together with (1) and (2) implies that

$$d(x|\mathrm{ML}_x(\alpha)) \leq \beta + \alpha \tag{3}$$

(with logarithmic precision). Thus $d(x|\mathrm{ML}_x(\alpha))$ is always small provided x is α,β-stochastic for small α,β.

2.2 The Case of Arbitrary α

Assume now that x was drawn at random from a set A that has large complexity. Say, x was obtained by adding noise to a clean musical record y. In other words, x was drawn at random from the set A consisting of all x' that can by obtained from y by adding noise of certain type. Then with high probability $d(x|A)$ is small. That is, we may assume that x is α,β-stochastic for small β and $\alpha = C(A) \approx C(y)$. Does MDL or ML based search work well for such x? The inequalities (2) and (3) do not guarantee that any more, if $C(y)$ is large. Nevertheless, the following theorem shows that both MDL search and ML search work well.

Theorem 2 **([7]).** *If x is α, β-stochastic and $\alpha \le C(x)$, then there is a set $A \ni x$ with $C(A) \le \alpha + O(\log n)$ and $\log |A| \le C(x) - \alpha + \beta$ and hence $C(A) + \log |A| \le C(x) + \beta + O(\log n)$ (the set A is a sufficient statistic).*

Note that the explanation A from the theorem witnesses

$$d(x|\mathrm{ML}_x(\alpha + O(\log n))) \le \beta + O(\log n). \tag{4}$$

3 Structure Sets of a String

The next question is whether the inequality (4) is indeed an improvement over the inequality (3). That is, are there α, β-stochastic strings (for small β and large α) that are not α', β'-stochastic for much smaller α' and, may be, slightly larger β'. More generally, what shape can have the " structure set"

$$S_x = \{\langle \alpha, \beta \rangle \mid x \text{ is } \alpha, \beta\text{-stochastic}\}?$$

The next theorem shows that S_x can have almost any shape. For instance, for all large enough n there is $n/2, O(\log n)$-stochastic string that is not $n/3, n/3$-stochastic.

Theorem 3 **([7]).** *For every string x of length n and Kolmogorov complexity k the set S_x is upward closed and contains some pairs that are $O(\log n)$-close*[3] *to the pairs $\langle k, 0 \rangle$ and $\langle 0, n-k \rangle$. On the other hand, for all n and $k \le n$, if an upward closed set $S \subset \mathbb{N} \times \mathbb{N}$ contains the pairs $\langle k, 0 \rangle$, $\langle 0, n-k \rangle$, then there is x of length n and complexity $k + O(\log n + C(\tilde{S}))$ such that S_x is $O(\log n + C(\tilde{S}))$-close to S. Here $\tilde{S}$ stands for the set of minimal points in S.*

By Theorem 2 the set S_x is $O(\log n)$-close to another structure set

$$L_x = \{\langle \alpha, \gamma \rangle \mid \text{there is } A \ni x \text{ with } C(A) \le \alpha,\ C(A) + \log |A| - C(x) \le \gamma\}.$$

Thus Theorem 3 describes also all possible shapes of the set L_x. Theorem 3 also provides a description of possible shapes of the following set:

$$P_x = \{\langle \alpha, \delta \rangle \mid \text{there is } A \ni x \text{ with } C(A) \le \alpha,\ \log |A| \le \delta\}.$$

This set is called the Kolmogorov's structure set, as it was defined by Kolmogorov in [3]. Indeed, by Theorem 2, the set P_x is $O(\log n)$-close to the set

$$\{\langle \alpha, C(x) - \alpha + \beta \rangle \mid \alpha \le C(x),\ \langle \alpha, \beta \rangle \in S_x\} \cup \{\langle \alpha, 0 \rangle \mid \alpha \ge C(x)\}.$$

[3] We say that u is ε-close to v if the Euclidean distance between u and v is at most ε. Sets U, V are ε-close if for every $u \in U$ there is $v \in V$ at the distance at most ε from u and vice versa.

References

1. Gács, P., Tromp, J., Vitányi, P.M.B.: Algorithmic statistics. IEEE Trans. Inform. Th. 47(6), 2443–2463 (2001)
2. Kolmogorov, A.N.: Three approaches to the quantitative definition of information. Problems Inform. Transmission 1(1), 1–7 (1965)
3. Kolmogorov, A.N.: Talk at the Information Theory Symposium in Tallinn, Estonia (1974)
4. Li, M., Vitányi, P.M.B.: An Introduction to Kolmogorov Complexity and its Applications , 2nd edn. Springer, New York (1997)
5. Shen, A.K.: The concept of (α, β)-stochasticity in the Kolmogorov sense, and its properties. Soviet Math. Dokl. 28(1), 295–299 (1983)
6. Shen, A.K.: Discussion on Kolmogorov complexity and statistical analysis. The Computer Journal 42(4), 340–342 (1999)
7. Vereshchagin, N.K., Vitányi, P.M.B.: Kolmogorov's structure functions and model selection. IEEE Trans. Information Theory 50(12), 3265–3290 (2004)

Automatic Verification of Heap-Manipulating Programs Using Separation Logic

Hongseok Yang

Queen Mary University of London, UK

Abstract. The incorrect use of pointers, such as null pointer dereference and memory leak, is one of the most common sources of program errors. In this talk, I will describe our techniques for automatically verifying the absence of such pointer errors, which we have been developing for the past three years, based on a new program logic called separation logic.

This talk has two goals. The first is to show, by demo, the current status of techniques for automatically verifying pointer safety. The second is to present interesting instances of the interplay between automatic verification and program logic. In order to reduce the complexity of formal (manual) verification of programs, separation logic has unusual proof rules that exploit programming disciplines used by skilled software developers. I will explain how such rules have been used to improve the performance of our automatic verification techniques. Regarding the influence of automatic verification on program logic, I will describe new types of theorem proving questions on separation logic that were motivated by automatic verification.

Canonical Calculi: Invertibility, Axiom Expansion and (Non)-determinism

Arnon Avron[1], Agata Ciabattoni[2], and Anna Zamansky[1]

[1] Tel-Aviv University
[2] Vienna University of Technology

Abstract. We apply the semantic tool of non-deterministic matrices to characterize two important properties of canonical Gentzen-type calculi: invertibility of rules and axiom expansion. We show that in every canonical calculus G satisfying a natural condition, the following are equivalent: (i) the connectives of G admit axiom expansion, (ii) the rules of G are invertible, and (iii) G has a characteristic finite deterministic matrix.

1 Introduction

Canonical systems are sequent calculi which in addition to the standard axioms and structural rules have only logical rules in which exactly one occurrence of a connective is introduced and no other connective is mentioned. Intuitively, the term "canonical systems" refers to systems in which the introduction rules of a logical connective determine the semantic meaning of that connective[1]. It was shown in [1,2] that such systems are semantically characterized by two-valued *non-deterministic matrices (2Nmatrices)*. These structures form a natural generalization of the standard multi-valued matrices, in which the truth-value assigned to a complex formula is chosen *non-deterministically* out of a given set of options. Moreover, there is a remarkable triple correspondence between the existence of a characteristic 2Nmatrix for a canonical system, the ability to eliminate cuts in it and a constructive syntactic criterion called coherence. Here we show that in the context of canonical systems, 2Nmatrices play a prominent role not only in the phenomena of cut-elimination, but also in two other important properties of sequent calculi: invertibility of logical rules and completeness of atomic axioms (axiom expansion). The former is a key property in many deduction formalisms, such as Rasiowa-Sikorski (R-S) systems [12,10] (also known as dual tableaux), where it induces an algorithm for finding a proof of a complex formula, if such a proof exists. The latter is also often considered crucial when designing "well-behaved" systems (see e.g. [9]). There are a number of works providing syntactic and semantic criteria for these properties in various calculi. Syntactic sufficient conditions for invertibility and axiom expansion in sequent calculi possibly without structural rules and with quantifier rules were

[1] This is according to a long tradition in the philosophy of logic, established by Gentzen in his classical paper *"Investigations Into Logical Deduction"* ([8]).

introduced in [6] and [11]. A semantic characterization of axiom expansion in single-conclusioned sequent calculi with arbitrary structural rules was provided in [7] in the framework of phase spaces. In the context of labeled sequent calculi (of which canonical calculi are a particular instance), [5] shows that the existence of a finite deterministic matrix is a necessary condition for axiom expansion. In this paper we extend these results by showing that the existence of a finite deterministic matrix for a coherent canonical calculus is also a *sufficient* condition for axiom expansion. Furthermore, we prove that it is also a necessary condition for invertibility. For coherent canonical calculi G in *normal form* (to which every canonical calculus can be transformed), an even stronger correspondence is established: (i) the connectives of G admit axiom expansion, iff (ii) the rules of G are invertible, iff (iii) G has a two-valued deterministic characteristic matrix.

2 Preliminaries

Henceforth $\mathcal{L}$ is a propositional language and $Frm_{\mathcal{L}}$ the set of its wffs. We use the metavariables $\Gamma, \Delta, \Sigma, \Pi$ for sets of $\mathcal{L}$-formulas. By a *sequent* we shall mean an expression of the form $\Gamma \Rightarrow \Delta$, where Γ and Δ are *finite* sets of $\mathcal{L}$-formulas. A *clause* is a sequent consisting of atomic formulas. We use the metavariable Θ for sets of sequents, and the metavariable Ω for sequents.

Non-deterministic Matrices and Canonical Calculi

Below we shortly reproduce the basic definitions of the framework of Nmatrices and of canonical Gentzen-type systems from [1,2,4].

Definition 1. *A* non-deterministic matrix (Nmatrix) *for $\mathcal{L}$ is a tuple $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$, where (i) $\mathcal{V}$ is a non-empty set of truth values, (ii) $\mathcal{D}$ (designated truth values) is a non-empty proper subset of $\mathcal{V}$, and (iii) for every n-ary connective $\diamond$ of $\mathcal{L}$, $\mathcal{O}$ includes a corresponding function $\tilde{\diamond}_{\mathcal{M}} : \mathcal{V}^n \to 2^{\mathcal{V}} \setminus \{\emptyset\}$. A valuation $v : Frm_{\mathcal{L}} \to \mathcal{V}$ is* legal in an Nmatrix $\mathcal{M}$ *if for every n-ary connective $\diamond$ of $\mathcal{L}$: $v(\diamond(\psi_1, ..., \psi_n)) \in \tilde{\diamond}(v(\psi_1), ..., v(\psi_n))$.*

Ordinary (deterministic) matrices correspond to the case when each $\tilde{\diamond}$ is a function taking singleton values only. Thus in such matrices the truth-value assigned to $\diamond(\psi_1, ..., \psi_n)$ is uniquely determined by the truth-values of its subformulas: $v(\psi_1), ..., v(\psi_n)$. This, however, is not the case in Nmatrices, as v makes a non-deterministic choice out of the set of options $\tilde{\diamond}(v(\psi_1), ..., v(\psi_n))$.

Definition 2. *Let $\mathcal{M} = \langle \mathcal{V}, \mathcal{D}, \mathcal{O} \rangle$ be some Nmatrix for $\mathcal{L}$.*

1. *A valuation v satisfies a formula ψ (a set of formulas Γ) in $\mathcal{M}$, denoted by $v \models_{\mathcal{M}} \psi$ ($v \models_{\mathcal{M}} \Gamma$), if $v(\psi) \in \mathcal{D}$ ($v(\psi) \in \mathcal{D}$ for every $\psi \in \Gamma$).*
2. *A valuation v satisfies a sequent $\Omega = \Gamma \Rightarrow \Delta$ in $\mathcal{M}$ if whenever $v \models_{\mathcal{M}} \Gamma$, there is some $\psi \in \Delta$, such that $v \models_{\mathcal{M}} \psi$. v satisfies a set of sequents if it satisfies every sequent in this set.*
3. *For two sets of formulas Γ, Δ, we write $\Gamma \vdash_{\mathcal{M}} \Delta$ if for every $\mathcal{M}$-legal valuation v, $v \models_{\mathcal{M}} \Gamma$ implies that $v \models_{\mathcal{M}} \psi$ for some $\psi \in \Delta$.*

Notation 1. *Let G be any Gentzen-type calculus. We denote $\Gamma \vdash_G \Delta$ when a sequent $\Gamma_0 \Rightarrow \Delta_0$ is provable in G for some $\Gamma_0 \subseteq \Gamma$ and $\Delta_0 \subseteq \Delta$. For a set of sequents Θ and a sequent Ω, we denote $\Theta \vdash_G \Omega$ if Ω has a proof in G from Θ.*

Definition 3. *An Nmatrix $\mathcal{M}$ is* characteristic *for a calculus G if for every two sets of formulas Γ, Δ: $\Gamma \vdash_G \Delta$ iff $\Gamma \vdash_{\mathcal{M}} \Delta$. An Nmatrix $\mathcal{M}$ is* strongly characteristic *for G if for every set of sequents Θ and every sequent Ω: $\Theta \vdash_G \Omega$ iff $\Theta \vdash_{\mathcal{M}} \Omega$.*

As shown by the next theorem, Nmatrices can be used for characterizing logics that cannot be characterized by finite ordinary matrices.

Theorem 1. *([2]) Let $\mathcal{M}$ be a two-valued Nmatrix which has at least one proper non-deterministic operation. Then there is no finite deterministic matrix P, such that for every two sets of formulas Γ, Δ: $\Gamma \vdash_{\mathcal{M}} \Delta$ iff $\Gamma \vdash_P \Delta$.*

Definition 4. *A canonical rule of arity n is an expression $\{\Pi_i \Rightarrow \Sigma_i\}_{1 \leq i \leq m}/C$, where $m \geq 0$, C is either $\diamond(p_1, ..., p_n) \Rightarrow$ or $\Rightarrow \diamond(p_1, ..., p_n)$ for some n-ary connective $\diamond$, and for all $1 \leq i \leq m$, $\Pi_i \Rightarrow \Sigma_i$ is a clause such that $\Pi_i, \Sigma_i \subseteq \{p_1, ..., p_n\}$. An application of a canonical left introduction rule of the form $\{\Pi_i \Rightarrow \Sigma_i\}_{1 \leq i \leq m}/ \diamond (p_1, ..., p_n) \Rightarrow$ is any inference step of the form:*

$$\frac{\{\Gamma, \Pi_i^* \Rightarrow \Delta, \Sigma_i^*\}_{1 \leq i \leq m}}{\Gamma, \diamond(\psi_1, ..., \psi_n) \Rightarrow \Delta}$$

where Π_i^* and Σ_i^* are obtained from Π_i and Σ_i respectively by substituting ψ_j for p_j for all $1 \leq j \leq n$ and Γ, Δ are arbitrary sets of formulas. An application of a right introduction rule is defined similarly.

We call an application an identity application *when $\Sigma_i^* = \Sigma_i$ and $\Pi_i^* = \Pi_i$ for all $1 \leq i \leq n$.*

Definition 5. *A Gentzen-type calculus G is canonical if in addition to the standard axioms: $\psi \Rightarrow \psi$ (for any formula ψ), the cut rule*

$$\frac{\Gamma, A \Rightarrow \Delta \quad \Gamma \Rightarrow \Delta, A}{\Gamma \Rightarrow \Delta} \ (cut)$$

and the structural rule of weakening, G has only canonical logical rules.

Definition 6. *An* extended axiom *is any sequent of the form $\Gamma \Rightarrow \Delta$, where $\Gamma \cap \Delta \neq \emptyset$. An extended axiom is atomic if $\Gamma \cap \Delta$ contains an atomic formula.*

Definition 7. *A canonical calculus G is* coherent *if for every two rules of the forms $\Theta_1/ \Rightarrow \diamond(p_1, ..., p_n)$ and $\Theta_2/ \diamond (p_1, ..., p_n) \Rightarrow$, the set of clauses $\Theta_1 \cup \Theta_2$ is classically inconsistent (i.e., the empty set can be derived from it using cuts).*

The following well-known fact follows from the completeness of propositional resolution:

Proposition 1. *A set of clauses is satisfiable iff it is consistent.*

Notation 2. *Denote the clause $\Rightarrow p_i$ by S_i^t and the clause $p_i \Rightarrow$ by S_i^f. Let $\overline{a} = \langle a_1, ..., a_n \rangle \in \{t, f\}^n$. We denote $\mathsf{C}_{\overline{a}} = \{S_i^{a_i}\}_{1 \leq i \leq n}$.*

Lemma 1. *Let Θ be a set of clauses over $\{p_1, ..., p_n\}$. Let $\overline{a} = \langle a_1, ..., a_n \rangle \in \{t, f\}^n$ and let v be any valuation, such that $v(p_i) = a_i$ for all $1 \leq i \leq n$. Then $\Theta \cup \mathsf{C}_{\overline{a}}$ is consistent iff v is a (classical) model of Θ.*

Definition 8. *([4]) Let G be a coherent canonical calculus. The Nmatrix $\mathcal{M}_G$ is defined as follows for every n-ary connective $\diamond$ and $\overline{a} = \langle a_1, ..., a_n \rangle \in \{t, f\}^n$:*

$$\tilde{\diamond}_{\mathcal{M}_G}(a_1, ..., a_n) = \begin{cases} \{t\} & \text{if } \Theta / \Rightarrow \diamond(p_1, ..., p_n) \in G \text{ and } \Theta \cup \mathsf{C}_{\overline{a}} \text{ is consistent.} \\ \{f\} & \text{if } \Theta / \diamond(p_1, ..., p_n) \Rightarrow \in G \text{ and } \Theta \cup \mathsf{C}_{\overline{a}} \text{ is consistent.} \\ \{t, f\} & \text{otherwise} \end{cases}$$

Theorem 2. *([4]) $\mathcal{M}_G$ is a strongly characteristic Nmatrix for G.*

The following theorem from [2,4] establishes an exact correspondence between cut-elimination, two-valued Nmatrices, coherence of canonical calculi and their non-triviality, where a consequence relation $\vdash_G$ between sets of formulas is said to be *trivial* if for every two non-empty Γ, Δ: $\Gamma \vdash_G \Delta$.

Theorem 3. *([2,4]) Let G be a canonical calculus. Then the following statements are equivalent: (1) G is coherent, (2) $\vdash_G$ is non-trivial, (3) G has a strongly characteristic two-valued Nmatrix, (4) G has a characteristic two-valued Nmatrix, (5) G admits cut-elimination.*

Proposition 2. *Let G be a coherent canonical calculus. Then the following is equivalent: (i) $\mathcal{M}_G$ is deterministic, (ii) G has a finite characteristic two-valued deterministic matrix, (iii) G has a finite characteristic deterministic matrix.*

Proof. $((i) \Rightarrow (ii))$ and $((ii) \Rightarrow (iii))$ are trivial. For $((iii) \Rightarrow (i))$, assume that $\mathcal{M}_G$ has at least one non-deterministic operation. Then by Theorem 1, there is no finite ordinary matrix P, such that $\vdash_P = \vdash_{\mathcal{M}}$. Hence, there is no characteristic finite deterministic matrix for G.

Equivalence of Calculi

Definition 9. *Two sets of canonical rules S_1 and S_2 are* equivalent *if for every application of $R \in S_1$, its conclusion is derivable from its premises using rules from S_2 together with structural rules, and vice versa. Two canonical calculi G_1 and G_2 are* cut-free equivalent *if their rules are equivalent.*

Proposition 3. *For every two coherent canonical calculi G_1 and G_2 which are cut-free equivalent, $\mathcal{M}_{G_1} = \mathcal{M}_{G_2}$.*

Proof. First we shall need the following technical propositions and notations:

Notation 3. *For a set of formulas Γ, denote by $\mathsf{At}(\Gamma)$ the set of atomic formulas occurring in Γ. For a sequent $\Omega = \Gamma \Rightarrow \Delta$, denote by $\mathsf{At}(\Omega)$ the sequent*

$\mathsf{At}(\Gamma) \Rightarrow \mathsf{At}(\Delta)$. *For a clause Ω (a set of clauses Θ), denote by $mod(\Omega)$ the set of all the atomic valuations*[2] *which satisfy Ω (Θ).*

Lemma 2. *Let $R = \Theta/C$ be a canonical rule, where $\Theta = \{\Sigma_i \Rightarrow \Pi_i\}_{1\leq i\leq m}$. Consider an identity application (Defn. 4) of R with premises $\Omega_1, \ldots, \Omega_m$ and conclusion Ω. Then $(\bigcap_{1\leq i\leq m} mod(\mathsf{At}(\Omega_i))) \setminus mod(\mathsf{At}(\Omega)) \subseteq mod(\Theta)$.*

Proof. Let Ω be either $\Gamma \Rightarrow \Delta, \diamond(p_1, \ldots, p_n)$ or $\diamond(p_1, \ldots, p_n), \Gamma \Rightarrow \Delta$. Let $\Omega_i = \Gamma, \Sigma_i \Rightarrow \Pi_i, \Delta$. Let $v \in (\bigcap_{1\leq i\leq m} mod(\mathsf{At}(\Omega_i))) \setminus mod(\mathsf{At}(\Omega))$. $v \notin mod(\mathsf{At}(\Gamma \Rightarrow \Delta))$ (otherwise it would be the case that $v \in mod(\mathsf{At}(\Omega))$). Thus v satisfies $\mathsf{At}(\Gamma)$ but does not satisfy any of the formulas in $\mathsf{At}(\Delta)$. Let $1 \leq i \leq m$. If v satisfies Σ_i, then since v satisfies $\mathsf{At}(\Omega_i) = \mathsf{At}(\Gamma), \Sigma_i \Rightarrow \mathsf{At}(\Delta), \Pi_i$, there is some $\psi \in \Pi_i$, of which v is a model. Thus v satisfies $\Sigma_i \Rightarrow \Pi_i$ for all $1 \leq i \leq m$ and so $v \in mod(\Theta)$.

Corollary 1. *Let G be a canonical calculus. Suppose that Ω has a derivation in G from extended atomic axioms, which consists only of identity applications of canonical rules. If an atomic valuation v does not satisfy $\mathsf{At}(\Omega)$, then there is some canonical rule Θ/C applied in this derivation, such that $v \in mod(\Theta)$.*

Proof. By induction on the length l of the derivation of Ω. For $l = 1$ the claim trivially holds (v satisfies $\mathsf{At}(\Omega)$). Otherwise, consider the last rule applied in the derivation, which must be an identity application of some canonical rule Θ/C, where $\Theta = \{\Sigma_i \Rightarrow \Pi_i\}_{1\leq i\leq m}$. Denote its premises by $\Omega_1, \ldots, \Omega_m$ and its conclusion by Ω. Let $v \notin mod(\mathsf{At}(\Omega))$. If v satisfies $\mathsf{At}(\Omega_i)$ for all $1 \leq i \leq m$, then by Lemma 2, $v \in mod(\Theta)$. Otherwise there is some $1 \leq i \leq m$, such that v does not satisfy $\mathsf{At}(\Omega_i)$. By the induction hypothesis, v satisfies Θ' for some canonical rule Θ'/C' applied in the derivation of Ω_i.

Back to the proof of Proposition 3, let G_1 and G_2 be two coherent canonical calculi that are cut-free equivalent. Let $\diamond$ be some n-ary connective and $\overline{a} = \langle a_1, \ldots, a_n\rangle \in \{t, f\}^n$. Suppose that $\tilde{\diamond}_{\mathcal{M}_{G_1}}(\overline{a}) = \{t\}$. Then there is a rule in G_1 of the form $R = \Theta/ \Rightarrow \diamond(p_1, \ldots, p_n)$, such that $\Theta \cup \mathsf{C}_{\overline{a}}$ is consistent. Consider the application of R with premises Θ and conclusion $\Rightarrow \diamond(p_1, \ldots, p_n)$. Let v be any atomic valuation, such that $v(p_i) = a_i$ for all $1 \leq i \leq n$. Since $\Theta \cup \mathsf{C}_{\overline{a}}$ is consistent, by Lemma 1, $v \in mod(\Theta)$. Now since G_1 and G_2 are cut-free equivalent, there is a derivation D of $\Rightarrow \diamond(p_1, \ldots, p_n)$ from Θ using the rules of G_2 and weakening. Since $\mathsf{At}(\Rightarrow \diamond(p_1, \ldots, p_n)) = \emptyset$, $v \notin \mathsf{At}(\Rightarrow \diamond(p_1, \ldots, p_n))$, and by Corollary 1, there is some rule Θ'/S of G_2 applied in D, such that $v \in mod(\Theta')$. Since the derivation of $\Rightarrow \diamond(p_1, \ldots, p_n)$ from Θ is cut-free, it must be the case that this application is an identity application and S is the sequent $\Rightarrow (p_1, \ldots, p_n)$. By Lemma 1, $\Theta' \cup \mathsf{C}_{\overline{a}}$ is consistent. Hence, $\tilde{\diamond}_{\mathcal{M}_{G_2}}(\overline{a}) = \{t\}$. The case when $\tilde{\diamond}_{\mathcal{M}_{G_1}}(\overline{a}) = \{f\}$ is handled similarly. If $\tilde{\diamond}_{\mathcal{M}_{G_2}}(\overline{a}) = \{t\}$ (or $\tilde{\diamond}_{\mathcal{M}_{G_2}}(\overline{a}) = \{f\}$), the proof that $\tilde{\diamond}_{\mathcal{M}_{G_1}}(\overline{a}) = \{t\}$ (or $\tilde{\diamond}_{\mathcal{M}_{G_1}}(\overline{a}) = \{f\}$) is symmetric to the previous case.

[2] By an atomic valuation we mean any mapping from the atomic formulas of $\mathcal{L}$ to $\{t, f\}$.

We leave the following easy proposition to the reader:

Proposition 4. *If a canonical calculus G is coherent, so is any canonical calculus G' which is cut-free equivalent to G.*

Canonical Calculi in Normal Form

A canonical calculus may have a number of right (and left) introduction rules for the same connective. However, below we show (an adaptation of proofs from [3] and [5]) that any canonical calculus can be transformed (normalized) into a cut-free equivalent calculus with at most one right and one left introduction rule for each connective.

Definition 10. *We say that sequent $\Gamma \Rightarrow \Delta$ is* subsumed *by a sequent $\Gamma' \Rightarrow \Delta'$ if $\Gamma' \subseteq \Gamma$ and $\Delta' \subseteq \Delta$. A canonical calculus G is in* normal form *if (i) G has at most one left and at most one right introduction rule for each connective, (ii) its introduction rules have no extended axioms as their premises, and (iii) its introduction rules have no clauses in their premises which are subsumed by some other clause in their premises.*

Lemma 3. *Let R be a canonical rule having an extended axiom as one of its premises. The rule obtained by discarding this premise is equivalent to R.*

Proposition 5. *Every canonical calculus G has a cut-free equivalent calculus G^n in normal form.*

Proof. Let us describe the transformation of G into a calculus G^n in normal form. Take a pair of rules in G of the forms $R_1 = \{\Sigma_i^1 \Rightarrow \Pi_i^1\}_{1\leq i\leq m}/ \Rightarrow \diamond(p_1 \ldots p_n)$ and $R_2 = \{\Sigma_j^2 \Rightarrow \Pi_j^2\}_{1\leq j\leq l}/ \Rightarrow \diamond(p_1 \ldots p_n)$. Replace R_1 and R_2 in G by $R = \{\Sigma_i^1, \Sigma_j^2 \Rightarrow \Pi_i^1, \Pi_j^2\}_{1\leq i\leq m, 1\leq j\leq l}/ \Rightarrow \diamond(p_1 \ldots p_n)$. Clearly, any application of R can be simulated by applying R_1 and R_2. Moreover, any application of R_1 and of R_2 can be simulated by weakening and R. Hence, $\{R\}$ and $\{R_1, R_2\}$ are cut-free equivalent. By repeatedly applying this step, we get at most one left and one right introduction rule for each connective. Next, in the obtained rules discard the premises which are extended axioms. By Lemma 3, G and the resulting calculus G' are cut-free equivalent. Finally, in each rule of G' discard any premise $\Gamma \Rightarrow \Delta$ subsumed by any other premise $\Gamma' \Rightarrow \Delta'$. The resulting calculus is cut-free equivalent to G' as $\Gamma \Rightarrow \Delta$ can be derived from $\Gamma' \Rightarrow \Delta'$ using weakening.

Example 1. Consider the canonical calculus G_X with four introduction rules for the binary connective X (representing XOR):

$$\{\Rightarrow p_1 \ ; \ p_2 \Rightarrow\}/ \Rightarrow p_1 X p_2 \quad \{\Rightarrow p_2 \ ; \ p_1 \Rightarrow\}/ \Rightarrow p_1 X p_2$$

$$\{\Rightarrow p_1 \ ; \ \Rightarrow p_2\}/p_1 X p_2 \Rightarrow \quad \{p_1 \Rightarrow \ ; \ p_2 \Rightarrow\}/p_1 X p_2 \Rightarrow$$

This calculus can be transformed into a cut-free equivalent calculus G_X^n in normal form as follows. We start by replacing the first two rules by the following rule:

$$\{\Rightarrow p_1, p_2 \ ; \ p_1, p_2 \Rightarrow; \ p_1 \Rightarrow p_1 \ ; \ p_2 \Rightarrow p_2\}/ \Rightarrow p_1 X p_2$$

The second pair of rules can be replaced by:

$$\{p_1 \Rightarrow p_2 \ ; \ p_2 \Rightarrow p_1; \ p_1 \Rightarrow p_1 \ ; \ p_2 \Rightarrow p_2\}/p_1 X p_2 \Rightarrow$$

Finally, by Lemma 3, the axioms in the premises can be discarded and we get the following cut-free equivalent calculus G^n_X in normal form:

$$\{\Rightarrow p_1, p_2 \ ; \ p_1, p_2 \Rightarrow\}/ \Rightarrow p_1 X p_2 \quad \{p_1 \Rightarrow p_2 \ ; \ p_2 \Rightarrow p_1\}/p_1 X p_2 \Rightarrow$$

3 Investigating Invertibility

In this section we investigate the connection between invertibility and determinism in coherent canonical calculi. We show that the latter is a necessary condition for invertibility, which turns out to be also sufficient for calculi in normal form. The usual definition of invertibility of rules is the following:

Definition 11. *A rule R is* invertible in a calculus G *if for every application of R it holds that whenever its conclusion is provable in G, also each of its premises is provable in G.*

Notation 4. *Henceforth we use the metavariable R to refer to a canonical rule of the form $\{\Sigma_i \Rightarrow \Pi_i\}_{1\leq i\leq m}/ \Rightarrow \diamond(p_1, ..., p_n)$.*

We now introduce a useful notion which is equivalent to invertibility in the context of canonical calculi.

Definition 12. *Let G be a canonical calculus. A rule R is* canonically invertible *in G if for every $1 \leq i \leq m$: $\Sigma_i \Rightarrow \Pi_i$ has a proof in G from $\Rightarrow \diamond(p_1, ..., p_n)$. Canonical invertibility for left introduction rules is defined similarly.*

Remark 1. It is important to note that unlike standard invertibility, canonical invertibility is defined for *rules*, and not their instances.

Proposition 6. *A canonical rule is invertible in a canonical calculus G iff it is canonically invertible in G.*

Proof. ($\Leftarrow$) Assume w.l.o.g. that a rule R is canonically invertible in G. Consider an application of R with premises $\Gamma, \Sigma^*_1 \Rightarrow \Delta, \Pi^*_1; \ \ldots \ ; \ \Gamma, \Sigma^*_m \Rightarrow \Delta, \Pi^*_m$ and conclusion $\Gamma \Rightarrow \Delta, \diamond(\psi_1, ..., \psi_n)$ where for all $1 \leq j \leq m$, Σ^*_j, Π^*_j are obtained from Σ_j, Π_j by replacing each p_k by ψ_k for all $1 \leq k \leq n$. Suppose that $\vdash_G \Gamma \Rightarrow \Delta, \diamond(\psi_1, ..., \psi_n)$. We need to show that $\vdash_G \Gamma, \Sigma^*_j \Rightarrow \Delta, \Pi^*_j$ for all $1 \leq j \leq m$. Being R canonically invertible, there is a proof of $\Sigma_j \Rightarrow \Pi_j$ from $\Rightarrow \diamond(p_1, ..., p_n)$. By replacing in this proof each p_k by ψ_k and adding the contexts Γ and Δ everywhere, we obtain a proof of $\Gamma, \Sigma^*_j \Rightarrow \Delta, \Pi^*_j$ from $\Gamma \Rightarrow \Delta, \diamond(\psi_1, ..., \psi_n)$. Thus if $\Gamma \Rightarrow \Delta, \diamond(\psi_1, ..., \psi_n)$ is provable, so is $\Gamma, \Sigma^*_j \Rightarrow \Delta, \Pi^*_j$. Hence R is invertible. ($\Rightarrow$) Assume that R is invertible in G. Consider the application of R with conclusion $\diamond(p_1, ..., p_n) \Rightarrow \diamond(p_1, ..., p_n)$. Being G canonical, $\diamond(p_1, ..., p_n) \Rightarrow \diamond(p_1, ..., p_n)$ is provable in G. Since R is invertible, its premises $\Sigma_i, \diamond(p_1, ..., p_n) \Rightarrow \Pi_i$ are provable as well. By applying cut and weakening, we obtain a proof of $\Sigma_i \Rightarrow \Pi_i$ from $\Rightarrow \diamond(p_1, ..., p_n)$ for every $1 \leq i \leq m$ and the claim follows.

Next we introduce the notion of *expandability* of rules, and show that it is equivalent to invertibility in coherent canonical calculi.

Definition 13. *A canonical right introduction rule R is* expandable *in a canonical calculus G if for every $1 \leq i \leq m$: $\diamond(p_1, ..., p_n), \Sigma_i \Rightarrow \Pi_i$ has a* cut-free *proof in G. The notion of expandability in G for a left introduction rule is defined symmetrically.*

Proposition 7. *For any canonical calculus G, every expandable rule is invertible. If G is coherent, then every invertible rule is expandable.*

Proof. Let G be any canonical calculus. Assume w.l.o.g. that the rule R is expandable in G. Hence $\Sigma_i, \diamond(p_1, ..., p_n) \Rightarrow \Pi_i$ is provable for each $1 \leq i \leq m$. By cut, $\Sigma_i \Rightarrow \Pi_i$ is provable from $\Rightarrow \diamond(p_1, ..., p_n)$. Thus R is canonically invertible, and hence invertible by Proposition 6. Now assume that G is coherent and R is invertible in G. By Proposition 6, R is canonically invertible, and so for all $1 \leq i \leq m$: $\Sigma_i \Rightarrow \Pi_i$ is derivable from $\Rightarrow \diamond(p_1, \ldots, p_n)$. By adding $\diamond(p_1, \ldots, p_n)$ on the left side of all the sequents in the derivation, we obtain a derivation of $\diamond(p_1, \ldots, p_n), \Sigma_i \Rightarrow \Pi_i$ in G. Since G is coherent, by Theorem 3 it admits cut-elimination, thus we have a cut-free derivation of $\diamond(p_1, \ldots, p_n), \Sigma_i \Rightarrow \Pi_i$ in G, and hence R is expandable.

Although expandability and invertibility are equivalent for coherent canonical calculi, checking the former is an easier task, as it amounts to checking whether a sequent is cut-free provable.

Not surprisingly, in canonical calculi which are not coherent (and hence do not admit cut-elimination by Theorem 3), expandability is strictly stronger than invertibility. This is demonstrated by the following example.

Example 2. Consider the following non-coherent calculus G_B:

$$R_1 = \{p_1 \Rightarrow p_2\} / \Rightarrow p_1 \star p_2 \qquad R_2 = \{p_1 \Rightarrow p_2\} / p_1 \star p_2 \Rightarrow$$

Neither $p_1 \star p_2, p_1 \Rightarrow p_2$ nor $p_1 \Rightarrow p_2, p_1 \star p_2$ have a cut-free derivation in G_B. Indeed, while trying to find a proof bottom-up, the only rules which could be applied are either introduction rules for $\star$ or structural rules but these do not lead to (extended) axioms. Thus the above rules are not expandable. However, $p_1 \Rightarrow p_2$ has a derivation[3] (using cuts) in G_B:

$$\dfrac{\dfrac{\dfrac{p_1 \Rightarrow p_1}{p_1, p_2 \Rightarrow p_1}\,(w,l)}{p_1 \Rightarrow p_2 \star p_1}\,(R_1) \qquad \dfrac{\dfrac{p_2 \Rightarrow p_2}{p_2 \Rightarrow p_1, p_2}\,(w,r)}{p_2 \star p_1 \Rightarrow p_2}\,(R_2)}{p_1 \Rightarrow p_2}\,(cut)$$

Thus R_1 and R_2 are invertible, although not expandable.

Proposition 8. *Let G be a coherent canonical calculus. If G has an invertible rule for $\diamond$, then $\tilde{\diamond}_{\mathcal{M}_G}$ is deterministic.*

[3] Note that by Theorem 3, G_B is trivial as it is not coherent. Hence, for any two atoms p, q: $\vdash_{G_B} p \Rightarrow q$.

Proof. Assume w.l.o.g. that R is invertible in G. Suppose by contradiction that $\tilde{\diamond}_{\mathcal{M}_G}$ is not deterministic. Then there is some $\overline{a} = \langle a_1, ..., a_n\rangle \in \{t, f\}^n$, such that $\tilde{\diamond}_{\mathcal{M}_G}(\overline{a}) = \{t, f\}$. Let v be any $\mathcal{M}_G$-legal valuation, such that $v(p_i) = a_i$ and $v(\diamond(p_1, ..., p_n)) = t$ (such v exists since $\tilde{\diamond}_{\mathcal{M}_G}(\overline{a}) = \{t, f\}$). $\Theta \cup \mathsf{C}_{\overline{a}}$ is inconsistent (since otherwise by the definition of $\mathcal{M}_G$, it would be the case that $\tilde{\diamond}_{\mathcal{M}_G}(\overline{a}) = \{t\}$ due to the rule R). Thus $(*)$ there is some $1 \leq j_v \leq m$, for which v does not satisfy the sequent $\Sigma_{j_v} \Rightarrow \Pi_{j_v}$ (otherwise, since v also satisfies $\mathsf{C}_{\langle a_1,...,a_n\rangle}$ the set of clauses $\Theta \cup \mathsf{C}_{\overline{a}}$ would be consistent). Since R is invertible, by Proposition 6 it is also canonically invertible. Then for every $1 \leq i \leq m$, $\Sigma_i \Rightarrow \Pi_i$ is provable in G from $\Rightarrow \diamond(p_1, ..., p_n)$. Since $\mathcal{M}_G$ is strongly characteristic for G, $\Rightarrow \diamond(p_1, ..., p_n) \vdash_{\mathcal{M}_G} \Sigma_i \Rightarrow \Pi_i$ for every $1 \leq i \leq m$. Since v satisfies $\Rightarrow \diamond(p_1, ..., p_n)$, it should also satisfy $\Sigma_{j_v} \Rightarrow \Pi_{j_v}$, which contradicts $(*)$.

The following theorem establishes the correspondence between determinism, invertibility and expandability:

Theorem 4. *Let $\mathcal{L}$ be a propositional language and G a coherent canonical calculus in normal form. The following statements are equivalent:*

1. *G has an invertible rule for $\diamond$.*
2. *G has an expandable rule for $\diamond$.*
3. *$\tilde{\diamond}_{\mathcal{M}_G}$ is deterministic.*
4. *G has a rule for $\diamond$ and all its rules are invertible.*
5. *G has a rule for $\diamond$ and all its rules are expandable.*

Proof. $1 \Rightarrow 3$ follows by Proposition 8. $1 \Leftrightarrow 2$ and $4 \Leftrightarrow 5$ follow by Proposition 7. $4 \Rightarrow 1$ follows trivially. It remains to show that $3 \Rightarrow 5$. Suppose that $\mathcal{M}_G$ is deterministic. By the definition of $\mathcal{M}_G$, there must be at least one rule for $\diamond$, as otherwise $\tilde{\diamond}_{\mathcal{M}_G}(\overline{a}) = \{t, f\}$ for every $\overline{a} \in \{t, f\}^n$. Let R be any such rule w.l.o.g. Suppose by contradiction that R is not expandable in G. Then there is some $1 \leq i \leq m$, such that $\diamond(p_1, ..., p_n), \Sigma_i {\Rightarrow} \Pi_i$ has no cut-free proof in G. Since G is coherent, by Theorem 3 it admits cut-elimination, and so $\diamond(p_1, ..., p_n), \Sigma_i {\Rightarrow} \Pi_i$ is not provable in G. Since $\mathcal{M}_G$ is a characteristic Nmatrix for G, $\Sigma_i, \diamond(p_1, ..., p_n) \not\vdash_{\mathcal{M}_G} \Pi_i$. Then there is an $\mathcal{M}_G$-legal valuation v, such that $v \models_{\mathcal{M}_G} \{\diamond(p_1, ..., p_n)\} \cup \Sigma_i$ and for every $\psi \in \Pi_i$: $v \not\models_{\mathcal{M}_G} \psi$. Let $\overline{a} = \langle v(p_1), ..., v(p_n)\rangle$. By Lemma 1, $(*)$ $\{\Sigma_i \Rightarrow \Pi_i\}_{1 \leq i \leq m} \cup \mathsf{C}_{\overline{a}}$ is inconsistent. Since $\mathcal{M}_G$ is deterministic, either $\tilde{\diamond}_{\mathcal{M}_G}(\overline{a}) = \{t\}$ or $\tilde{\diamond}_{\mathcal{M}_G}(\overline{a}) = \{f\}$. But the first case is impossible by definition of $\mathcal{M}_G$ and the fact that R is the only right introduction rule for $\diamond$. Thus $\tilde{\diamond}_{\mathcal{M}_G}(\overline{v}) = \{f\}$, in contradiction to our assumption that $v \models_{\mathcal{M}_G} \diamond(p_1, ..., p_n)$. Therefore R is expandable in G.

The next example demonstrates that Theorem 4 does not hold for calculi which are not in normal form.

Example 3. Consider the calculus G_X in Example 1 and its associated (deterministic) Nmatrix $\mathcal{M}_{G_X}$:

X	t	f
t	$\{f\}$	$\{t\}$
f	$\{t\}$	$\{f\}$

It is easy to see that $\Rightarrow p_1 X p_2 \not\vdash_{\mathcal{M}_{G_X}} \Rightarrow p_1$. Hence $\Rightarrow p_1$ is not derivable in G_X from $\Rightarrow p_1 X p_2$ and so the first rule is not canonically invertible. By Proposition 6 it is not invertible, and by Proposition 7, it is also not expandable.

Corollary 2. *If a coherent canonical calculus G in normal form has a right (left) invertible rule for $\diamond$ with a non-empty set of premises, then it also has a left (right) invertible rule for $\diamond$.*

Proof. Let G be a canonical coherent calculus in normal form with an invertible rule $[\Theta / \Rightarrow \diamond(p_1, \dots, p_n)]$. By Theorem 4, $\tilde{\diamond}_{\mathcal{M}_G}$ is deterministic. Since Θ is non-empty and it cannot be a set of extended axioms (recall that G is in normal form), there is some $v \notin mod(\Theta)$ (cf. Notation 3). But since $\tilde{\diamond}_{\mathcal{M}_G}(v(p_1), \dots, v(p_n))$ is deterministic, there must be a rule $[\Theta'/C']$, such that $\Theta' \cup \mathsf{C}_{\langle v(p_1), \dots, v(p_n)\rangle}$ is consistent. Since G is in normal form and $\Theta' \neq \Theta$, this cannot be a right introduction rule for $\diamond$, hence C' is $\diamond(p_1, \dots, p_n) \Rightarrow$. By Theorem 4, this rule is invertible.

4 Investigating Axiom Expansion

Axiom expansion is an important property of deduction systems, which allows for the reduction of logical axioms to the atomic case. We show that for coherent canonical calculi this property fully characterizes the existence for a calculus of a two-valued deterministic characteristic matrix. Furthermore we show that in coherent canonical calculi axiom expansion is a necessary condition for invertibility, which turns out to be also sufficient for calculi in normal form.

Definition 14 ([7]). *An n-ary connective $\diamond$ admits axiom expansion in a calculus G if whenever $\diamond(p_1, ..., p_n) \Rightarrow \diamond(p_1, ..., p_n)$ is provable in G, it has a cut-free derivation in G from atomic axioms of the form $\{p_i \Rightarrow p_i\}_{1 \le i \le n}$.*

Proposition 9. *Let G be a canonical calculus. If G has an expandable rule for $\diamond$, then $\diamond$ admits axiom expansion in G.*

Proof. Suppose without loss of generality that G has a right introduction rule $R = \{\Sigma_i \Rightarrow \Pi_i\}_{1 \le i \le m} / \Rightarrow \diamond(p_1, ..., p_n)$, which is expandable in G. Then $(*)$ $\Sigma_i, \diamond(p_1, ..., p_n) \Rightarrow \Pi_i$ has a cut-free derivation in G for every $1 \le i \le m$. Note that $\Sigma_i, \Pi_i \subseteq \{p_1, ..., p_n\}$ and hence the sequents denoted by $(*)$ are derivable from atomic axioms $\{p_i \Rightarrow p_i\}_{1 \le i \le n}$. By applying R with premises $\{\Sigma_i, \diamond(p_1, ..., p_n) \Rightarrow \Pi_i\}_{1 \le i \le m}$, we obtain the required cut-free derivation of $\diamond(p_1, ..., p_n) \Rightarrow \diamond(p_1, ..., p_n)$ in G from atomic axioms. Thus $\diamond$ admits axiom expansion in G.

Lemma 4. *Let G be a canonical calculus. If a sequent Ω has a cut-free proof in G from atomic axioms, then Ω also has a cut-free proof in G from atomic (extended) axioms with no application of weakening.*

Theorem 5. *Let G be a coherent canonical calculus. $\diamond$ admits axiom expansion in G iff $\tilde{\diamond}_{\mathcal{M}_G}$ is deterministic.*

Proof. ($\Rightarrow$) If $\diamond$ admits axiom expansion in G then $\diamond(p_1, \dots, p_n) \Rightarrow \diamond(p_1, \dots, p_n)$ is cut-free derivable from atomic axioms. By Lemma 4, we can assume that the derivation contains only extended atomic axioms and applications of canonical rules. Since there are no cuts, it is easy to see that the applications of canonical rules in this derivation must be identity applications of introduction rules for $\diamond$. Now since $\mathsf{At}(\diamond(p_1, \dots, p_n) \Rightarrow \diamond(p_1, \dots, p_n))$ is the empty sequent, Corollary 1 ensures that for every valuation v there is some logical rule Θ/C (where C is either $\Rightarrow \diamond(p_1, \dots, p_n)$ or $\diamond(p_1, \dots, p_n) \Rightarrow$) used in this derivation, such that $v \in mod(\Theta)$. By Lemma 1, for every $\overline{a} = \langle a_1, \dots, a_n \rangle \in \{t, f\}^n$ there is some canonical rule Θ/C for $\diamond$, such that $\Theta \cup \mathsf{C}_{\overline{a}}$ is consistent. Thus $\tilde{\diamond}_{\mathcal{M}_G}(a_1, \dots, a_n)$ is a singleton, and so $\tilde{\diamond}_{\mathcal{M}_G}$ is deterministic.

($\Leftarrow$) First transform G into a cut-free equivalent calculus G^n in normal form (cf. Proposition 5). By Proposition 4, G^n is coherent, and by Proposition 3, $\mathcal{M}_{G^n}$ is deterministic. By Theorem 4 and Proposition 9, $\diamond$ admits axiom expansion in G^n and therefore also in G, since G is cut-free equivalent to G^n.

Remark 2. An alternative proof of ($\Rightarrow$) is contained in [5] for a generalization of canonical calculi.

Corollary 3. *For a coherent canonical calculus G, every connective admits axiom expansion in G iff G has a two-valued characteristic deterministic matrix.*

Proof. Follows from the theorem above and Proposition 2.

Corollary 4. *If a coherent canonical calculus G has an invertible rule for $\diamond$, then $\diamond$ admits axiom expansion in G.*

Proof. If G has an invertible rule for $\diamond$, then by Proposition 7 it is also expandable. By Proposition 9, $\diamond$ admits axiom expansion in G.

We finish the paper by summarizing the correspondence between determinism, invertibility and axiom expansion:

Corollary 5. *Let $\mathcal{L}$ be a propositional language and G a coherent canonical calculus in normal form with introduction rules for each connective in $\mathcal{L}$. The following are equivalent: (i) The rules of G are invertible, (ii) G has a characteristic two-valued deterministic matrix, and (iii) Every connective of $\mathcal{L}$ admits axiom expansion in G.*

Proof. By Proposition 2, the existence of a two-valued characteristic deterministic matrix for G is equivalent to $\mathcal{M}_G$ being deterministic. The rest follows by Theorem 4, Corollary 2 and Theorem 5.

As shown by the following example the above correspondence does not hold for calculi which are not in normal form.

Example 4. Consider the calculus G_X of Example 1. Although the rules for the connective X are not invertible, X admits axiom expansion:

$$\dfrac{\dfrac{\dfrac{p_1 \Rightarrow p_1}{p_1 \Rightarrow p_1, p_1 X p_2} \quad \dfrac{\dfrac{p_2 \Rightarrow p_2}{p_2 \Rightarrow p_2, p_1} \quad \dfrac{p_1 \Rightarrow p_1}{p_2, p_1 \Rightarrow p_1}}{p_2 \Rightarrow p_1, p_1 X p_2}}{p_1 X p_2 \Rightarrow p_1, p_1 X p_2} \quad \dfrac{\dfrac{\dfrac{p_2 \Rightarrow p_2}{p_1 X p_2, p_2 \Rightarrow p_2} \quad \dfrac{\dfrac{p_1 \Rightarrow p_1}{p_2, p_1 \Rightarrow p_1} \quad \dfrac{p_2 \Rightarrow p_2}{p_2, p_1 \Rightarrow p_2}}{p_1 X p_2, p_2, p_1 \Rightarrow}}{p_1 X p_2, p_2 \Rightarrow p_1 X p_2}}{p_1 X p_2 \Rightarrow p_1 X p_2}$$

Acknowledgements

The first and the third authors are supported by the ISRAEL SCIENCE FOUNDATION, grant No 809-06. The second author is supported by the Austrian Science Fund (FWF), project No P18731.

References

1. Avron, A., Lev, I.: Canonical Propositional Gentzen-type Systems. In: Goré, R.P., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS (LNAI), vol. 2083, pp. 529–544. Springer, Heidelberg (2001)
2. Avron, A., Lev, I.: Non-deterministic Multi-valued Structures. Journal of Logic and Computation 15, 241–261 (2005)
3. Avron, A., Konikowska, B.: Proof Systems for Logics Based on Non-deterministic Multiple-valued Structures. Logic Journal of the IGPL 13, 365–387 (2005)
4. Avron, A., Zamansky, A.: Canonical Signed Calculi, Non-deterministic Matrices and Cut-elimination. In: Artemov, S., Nerode, A. (eds.) LFCS 2009. LNCS, vol. 5407, pp. 31–45. Springer, Heidelberg (2009)
5. Baaz, M., Fermüller, C.G., Salzer, G., Zach, R.: Labeled Calculi and Finite-valued Logics. Studia Logica 61, 7–33 (1998)
6. Chapman, P.: Syntactic Conditions for Invertibility in Sequent Calculi. Forthcoming in the Proceedings of CATS 2009. ACM Digital Library (2009)
7. Ciabattoni, A., Terui, K.: Towards a semantic characterization of cut-elimination. Studia Logica 82(1), 95–119 (2006)
8. Gentzen, G.: Investigations into Logical Deduction. In: Szabo, M.E. (ed.) The collected works of Gerhard Gentzen, pp. 68–131. North Holland, Amsterdam (1969)
9. Girard, J.-Y.: On the meaning of logical rules I: syntax vs. semantics. In: Berger, U., Schwichtenberg, H. (eds.) Computational Logic, pp. 215–272. Springer, Heidelberg (1999)
10. Konikowska, B.: Rasiowa-Sikorski Deduction Systems in Computer Science Applications. Theoretical Computer Science 286, 323–366 (2002)
11. Miller, D., Pimentel, E.: On the specification of sequent systems. In: Sutcliffe, G., Voronkov, A. (eds.) LPAR 2005. LNCS, vol. 3835, pp. 352–366. Springer, Heidelberg (2005)
12. Rasiowa, H., Sikorski, R.: The Mathematics of Metamathematics. PWN (Polish Scientific Publishers), Warsaw (1963)

Integrality Property in Preemptive Parallel Machine Scheduling[⋆]

Philippe Baptiste[1], Jacques Carlier[2], Alexander Kononov[3,4], Maurice Queyranne[5], Sergey Sevastyanov[3,4], and Maxim Sviridenko[6]

[1] CNRS, École Polytechnique, Paris, France
[2] CNRS, Heudiasyc, Univ. de Tech. de Compiègne, France
[3] Sobolev Institute of Mathematics, Novosibirsk, Russia
[4] Novosibirsk State University, Novosibirsk, Russia
[5] Faculty of Commerce and Business Administration, University of British Columbia, Vancouver, B.C., Canada
[6] IBM T.J. Watson Research Center, Yorktown Heights, USA

Abstract. We consider parallel machine scheduling problems with identical machines and preemption allowed. It is shown that every such problem with chain precedence constraints and release dates and an integer-concave objective function satisfies the following *integrality property*: for any problem instance with integral data there exists an optimal schedule where all interruptions occur at integral dates. As a straightforward consequence of this result, for a wide class of scheduling problems with unit processing times a so-called *preemption redundancy* property is valid. This means that every such preemptive scheduling problem is equivalent to its non-preemptive counterpart from the viewpoint of both its optimum value and the problem complexity. The equivalence provides new and simpler proofs for some known complexity results and closes a few open questions.

In the current paper we present some new structural results for preemptive scheduling problems. This work proceeds our previous research of structural properties of optimal solutions for preemptive scheduling problems initiated in [2], where some general results on the existence of optimal schedules and the existence of optimal schedules with a finite number of interruptions were established for a wide range of scheduling problems. Furthermore, two Rational Structure Theorems were proved in [2] for wide classes of scheduling problems, according to which for any problem instance heaving a nonempty set of feasible solutions there exists an optimal schedule with the following properties:

(1) the total number of interruptions grows polynomially with the number of operations and with the number of fixed dates specified in that instance;
(2) all operation start times and completion times and all interruptions occur at integer multiples of a rational number $\delta > 0$ with size polynomially bounded in the input size;

[⋆] Research of the Russian authors is supported by RFBR grant no. 08-01-00370 and Russian-Taiwan grant no. 08-06-92000. Research of the third author is partially supported by ADTP grant 2.1.1/3235.

(3) the optimal value of the objective function is an integer multiple of δ, where the size of the integer multiplier is also polynomially bounded in the input size.

These results were established for a wide class of preemptive scheduling models including both classical and non-traditional machine scheduling and project scheduling models with constrained resources and a large variety of objective functions including all classical ones. An important consequence of these Rational Structure Theorems is the fact that the decision versions of preemptive scheduling problems under consideration belong to class $\mathcal{NP}$.

A significantly stronger structural property (compared to that formulated in the above mentioned Rational Structure Theorems) is assumed, when we speak about the *integrality property*. The latter means that for any problem instance there exists an optimal preemptive schedule where all interruptions occur at *integral* dates. This property is investigated in the current paper for parallel machine problems with identical machines.

New Results. In the current paper we establish the integrality property for the preemptive version of the parallel machine scheduling problem with chain precedence constraints, release dates and an arbitrary regular integer-concave objective function. As a straightforward consequence of this result, a so-called *preemption redundancy* property holds for a wide class of scheduling problems with unit processing times. In particular, this closes two open questions on the preemption redundancy property of problems $P|p_j = 1, r_j, pmtn|\sum T_j$ and $P|p_j = 1, pmtn|\sum w_j T_j$ (see Brucker [4]). This property also implies that every such preemptive scheduling problem is equivalent to its non-preemptive counterparts from the viewpoint of both its optimum value and the problem complexity. The equivalence provides new and sometimes simpler proofs for some known complexity results. Specifically, our Theorem 3.2 implies the NP-hardness of $Pm|p_j = 1, chains, pmtn|\sum w_j C_j$ and $Pm|p_j = 1, chains, pmtn|\sum U_j$ (previously proved in [1, 6, 19]), and the polynomial time solvability of problem $P|p_j = 1, r_j, pmtn|\sum w_j T_j$ whose complexity status remained open. Furthermore, due to the strong NP-hardness of problem $1|p_j = 1, chains|\sum T_j$ established by Leung and Young [13], our result implies the strong NP-hardness of the $1|p_j = 1, chains, pmtn|\sum T_j$ problem. To our knowledge (see also the Brucker's home page [21]), the complexity status of this problem was open before.

Related Results. There are few systematic studies of such structural questions in the literature on preemptive scheduling, and most known results follow from either (i) the fact that there is no advantage to preemption [3], [4], or (ii) the existence or properties of polynomial time algorithms. Results following from (i) are clearly the strongest type of structural results one could hope for for scheduling problems. We refer to the standard scheduling literature (e.g., [12]) for many such classical results; another extensive reference is the book by Tanaev, Gordon and Shafransky [18].

Structural results following from (ii) have been obtained mostly for parallel machine and open shop problems. We use the standard three-field notation [12] to describe such scheduling problems. McNaughton [14] constructs an optimal schedule with at most $m-1$ interruptions for problem $P|pmtn|C_{\max}$ on m identical parallel machines and makespan objective. Sauer and Stone [16] (see also [15]) prove that for the parallel machine scheduling problem with n jobs, precedence constraints, unit processing times and the minimum makespan objective there is an optimal preemptive schedule with at most $n-1$ preemption dates. Gonzalez and Sahni [8] construct an optimum schedule with at most $2(m-1)$ interruptions for the uniform parallel machine version $Q|pmtn|C_{\max}$ of this problem. The bounds of McNaughton and of Gonzalez and Sahni on the number of interruptions (and preemption dates) are tight. Labetoulle et al. [10] prove that the natural greedy algorithm for the problem $Q|r_j, pmtn|\sum C_j$ with m machines and n jobs finds an optimal solution with at most $2n-m$ interruptions. For the unrelated parallel machine problem $R|pmtn|C_{\max}$, Lawler et al. [12] state that a procedure of Lawler and Labetoulle [11] can be modified to yield an optimal schedule with no more than $O(m^2)$ interruptions. Turning now to open shop problems, Gonzalez and Sahni [7] construct an optimal schedule for the problem $O|pmtn|C_{max}$ with m machines, n jobs and ξ operations, which has at most $\xi+n+m$ preemption dates. Du and Leung [5] proved the corresponding result for $O2|pmtn|\sum_j C_j$. Little attention seems to have been given in the literature to investigating of structural properties of optimal solutions for other preemptive scheduling problems.

Paper Outline. In the next section we give definitions of basic notions. In Section 2 we prove the integrality property for a class of parallel machine scheduling problems. Next we give a short conclusion in the last section.

1 Definitions

For two vectors $x' = (x'_1, \ldots, x'_n)$ and $x'' = (x''_1, \ldots, x''_n)$ we write $x' \le x''$, if the inequality $x'_i \le x''_i$ holds for each component i.

Definition 1. We say that a function $F(x)$ $(x \in \mathbb{R}^n)$ defined on a domain $D \subseteq \mathbb{R}^n$ is *nondecreasing* if $F(x') \le F(x'')$ holds for any pair of vectors $x', x'' \in D$ such that $x' \le x''$.

Definition 2. We say that a function $F(x)$ $(x \in \mathbb{R}^n)$ defined on a domain $D \subseteq \mathbb{R}^n$ is *continuous from the left*, if for any point $x \in D$ and any $\varepsilon > 0$ there exists a number $\delta > 0$ such that the inequality $|F(x) - F(x')| < \varepsilon$ holds for every $x' \in D$ such that $x' \le x$ and $x_i - x'_i \le \delta$, $\forall\ i = 1, \ldots, n$.

Definition 3. A real valued function $F(x)$ $(x \in \mathbb{R}^n)$ is called *regular*, if it is nondecreasing and continuous from the left.

Definition 4. We say that a function $F(x)$ $(x \in \mathbb{R}^n)$ is *integer-concave*, if the inequality

$$F(\lambda x' + (1-\lambda)x'') \ge \lambda F(x') + (1-\lambda)F(x'')$$

holds for any $\lambda \in [0,1]$ and any $x' = (x'_1, \ldots, x'_n),\ x'' = (x''_1, \ldots, x''_n)$ such that $x'_i, x''_i \in [t_i, t_i + 1]$ for some integer t_i, $i = 1, \ldots, n$.

For instance, *tardiness* ($T_j(C_j)$) is a classical example of a penalty function (of either an operation or a job completion time) which is integer-concave for any integral due date, but not concave. Another example of such function is the *being late* function ($U_j(C_j)$) taking 0-1 values. Consequently, the total weighted tardiness and the weighted number of late jobs ($\sum w_j T_j$ and $\sum w_j U_j$) are integer-concave functions. Of course, any linear function (like $\sum w_j C_j$), or just concave function (like $\sum \sqrt{C_j}$ or $\sum \log C_j$) is integer-concave, as well.

On the other hand, such a classical objective function as the makespan is not integer-concave. Indeed, let us consider the example with two jobs and $F(C_1, C_2) = \max\{C_1, C_2\} \doteq C_{\max}$, where C_i denotes the completion time of job i. Let $x' = (C'_1, C'_2) = (1, 0)$; $x'' = (C''_1, C''_2) = (0, 1)$. Then for the point $x = \frac{1}{2}x' + \frac{1}{2}x''$ we have $F(x) = \max\{0.5, 0.5\} = 0.5$, whereas $\frac{1}{2}F(x') + \frac{1}{2}F(x'') = 1$, and the required inequality is failed.

2 Integer Interruptions in Preemptive Parallel Machine Scheduling with Integer-Concave Objective Functions

Consider the situation when n jobs $J_1, \ldots, J_n$ with integer processing times $p_1, \ldots, p_n$ and release dates $r_1, \ldots, r_n$ have to be scheduled on m parallel identical machines preemptively. Following the notation of Graham *et al.* [9], this scheduling problem is denoted as $P|r_j, pmtn|F$.

The following theorem includes some of the results from [3] and [4] as special cases. In particular, our theorem implies the preemption redundancy for $P|r_j, p_j = 1, pmtn|\sum w_j U_j$ and $P|r_j, p_j = 1, pmtn|\sum w_j T_j$ thereby resolving an open question of [4].

Theorem 1. *For any instance of $P|r_j, pmtn|F(C_1, \ldots, C_n)$ with integer processing times and release dates and with the objective to minimize a regular integer-concave function $F(C_1, \ldots, C_n)$, there exists an optimal schedule where all interruptions occur at integral dates.*

Proof. Clearly, for any instance of the above problem the set of its feasible schedules is nonempty, and once the objective function is regular, there always exists an optimal schedule (as follows from Theorem 3.3 of [2]). Assume that we are given an optimal schedule S for a given instance of the above problem. Let $C_i(S)$ denote the completion time of job J_i ($i = 1, \ldots, n$) in that schedule, and $C_{\max}(S) = \max\limits_{i=1,\ldots,n} C_i(S)$. Let $\mathcal{T} = \{t \in \mathbb{Z} \mid \min_i r_i \le t < C_{\max}(S)\}$ be the set of time units occupied by the schedule S. We define a bipartite graph $G = (\{J_1, \ldots, J_n\}, \mathcal{T}; E)$ such that for any job J_i and a time point $t \in \mathcal{T}$, (J_i, t) belongs to E if and only if $r_i \le t < C_i(S)$. Every edge $e \in E$ has a unit *capacity.* We say that a flow $x = \{x_{it} \mid (J_i, t) \in E\}$ in network G is *feasible* if

$$\sum_t x_{it} = p_i, \ \forall\ i = 1, \ldots, n; \tag{1}$$

$$\sum_i x_{it} \le m, \ \forall\ t \in \mathcal{T}; \tag{2}$$

$$0 \leq x_{it} \leq 1, \ \forall\ i, t. \tag{3}$$

The set of feasible flows is denoted by X.

Let $x_{it}(S)$ be the volume of job J_i scheduled in S within the integral time interval $[t, t+1]$. It is clear that the set of values $x(S) = \{x_{it}(S) \,|\, (J_i, t) \in E\}$ meets requirements (1)–(3), and therefore, the flow $x(S)$ determined by schedule S belongs to X.

Next, for any $i = 1, \ldots, n$ we define an integral function t_i of $x \in X$:

$$t_i(x) \doteq \max\{t \,|\, x_{it} > 0\}.$$

(The function is well-defined, since we may assume, w.l.o.g., all p_i being positive.) Let

$$\bar{C}_i(x) \doteq t_i(x) + x_{it_i(x)}, \ \bar{C}(x) = (\bar{C}_1(x), \ldots, \bar{C}_n(x)).$$

Evidently, we have

$$\bar{C}_i(x(S)) \leq C_i(S). \tag{4}$$

Now we define a cost function $\tilde{F}$ on the set X of feasible flows:

$$\tilde{F}(x) \doteq F(\bar{C}(x)).$$

It follows from (4) and the fact that F is nondecreasing

$$\tilde{F}(x(S)) \leq F(C(S)). \tag{5}$$

Let us now prove that function $\tilde{F}(x)$ is concave on the set X of feasible flows. Consider $x', x'' \in X, \ \lambda \in (0,1), \ x = \lambda x' + (1-\lambda)x''$. First let us show that

$$t_i(x) = \max\{t_i(x'), t_i(x'')\}. \tag{6}$$

Indeed, since both coefficients λ and $(1-\lambda)$ are strictly positive, the value of $x_{it} = \lambda x'_{it} + (1-\lambda)x''_{it}$ is positive, while at least one of two components x'_{it} and x''_{it} is positive. On the other hand, if both components x'_{it} and x''_{it} are zero, then $x_{it} = 0$ as well, which implies (6).

Let $\tilde{C}_i(x') \doteq \max\{\bar{C}_i(x'), t_i(x'')\}$ and $\tilde{C}_i(x'') \doteq \max\{t_i(x'), \bar{C}_i(x'')\}$. Using (6), we now prove that

$$\tilde{C}_i(x'), \tilde{C}_i(x'') \in [t_i(x), t_i(x)+1]. \tag{7}$$

and

$$\bar{C}_i(x) = \lambda\tilde{C}_i(x') + (1-\lambda)\tilde{C}_i(x''). \tag{8}$$

Indeed, in the case $t_i(x') > t_i(x'')$ we have $\bar{C}_i(x'') \leq t_i(x'') + 1 \leq t_i(x') = t_i(x)$, hence, $\tilde{C}_i(x'') = t_i(x')$, whereas $\tilde{C}_i(x') = \bar{C}_i(x')$ (which provides (7)). Note also that $x''_{it_i(x)} = 0$. Therefore,

$$\begin{aligned}
\lambda\tilde{C}_i(x') + (1-\lambda)\tilde{C}_i(x'') &= \lambda\bar{C}_i(x') + (1-\lambda)t_i(x') = t_i(x') + \lambda x'_{it_i(x')} \\
&= t_i(x) + \lambda x'_{it_i(x)} + (1-\lambda)x''_{it_i(x)} \\
&= t_i(x) + x_{it_i(x)} = \bar{C}_i(x).
\end{aligned}$$

In the case $t_i(x') < t_i(x'')$ the proof of (7),(8) is similar. Finally, if $t_i(x') = t_i(x'') = t_i(x)$, then we have

$$\tilde{C}_i(x') = \bar{C}_i(x') \in [t_i(x), t_i(x)+1] \quad \text{and} \quad \tilde{C}_i(x'') = \bar{C}_i(x'') \in [t_i(x), t_i(x)+1].$$

So, we have (7). Using these relations, we obtain (8):

$$\begin{aligned} \lambda\tilde{C}_i(x') + (1-\lambda)\tilde{C}_i(x'') &= \lambda\bar{C}_i(x') + (1-\lambda)\bar{C}_i(x'') \\ &= t_i(x) + \lambda x'_{it_i(x)} + (1-\lambda)x''_{it_i(x)} \\ &= t_i(x) + x_{it_i(x)} = \bar{C}_i(x). \end{aligned}$$

Now, we can derive

$$\begin{aligned} \tilde{F}(x) \quad &= \quad F(\bar{C}(x)) \overset{(8)}{=} F(\lambda\tilde{C}(x') + (1-\lambda)\tilde{C}(x'')) \\ &\overset{(7)\text{ and i.c.}}{\geq} \quad \lambda F(\tilde{C}(x')) + (1-\lambda)F(\tilde{C}(x'')) \\ &\overset{\text{n.d.}}{\geq} \quad \lambda F(\bar{C}(x')) + (1-\lambda)F(\bar{C}(x'')) = \lambda\tilde{F}(x') + (1-\lambda)\tilde{F}(x''), \end{aligned}$$

which implies that function $\tilde{F}(x)$ is concave in X. (Here "i.c." means *integer-concaveness* and "n.d." means *nondecreasing*.)

Let $\tilde{x}$ be the flow minimizing $\tilde{F}(x)$ over all $x \in X$. Due to (5), we have

$$\tilde{F}(\tilde{x}) \leq \tilde{F}(x(S)) \leq F(C(S)). \tag{9}$$

Since the minimum of every concave function over a polytope is achieved on a vertex of the polytope, and since the transportation polytope (1)–(3) is integral, $\tilde{x}$ is an integer flow. Therefore, we can map the flow to a new schedule $\tilde{S}$, in which all interruptions occur at integral time points. The integrality of flow $\tilde{x}$ implies $\bar{C}_i(\tilde{x}) = C_i(\tilde{S})$ for every i, and therefore,

$$\tilde{F}(\tilde{x}) = F(\bar{C}_1(\tilde{x}), \ldots, \bar{C}_n(\tilde{x})) = F(C_1(\tilde{S}), \ldots, C_n(\tilde{S})) = F(C(\tilde{S})).$$

From (9) we have $F(C(\tilde{S})) \leq F(C(S))$, and therefore, schedule $\tilde{S}$ is also optimal.

A natural attempt to generalize the above theorem is to add precedence constraints. Unfortunately, Baptiste and Timkovsky [3] gave examples showing that preemptions are not redundant for $P2|p_j = 1, intree, pmtn|\sum C_j$ and $P2|p_j = 1, outtree, pmtn|\sum w_jC_j$. This means that the most we could hope for is to add chain precedence constraints, i.e., the directed graph corresponding to these precedence relations must be a collection of chains.

Theorem 2. *For any instance of $P|r_j, chains, pmtn|F(C_1, \ldots, C_n)$ with integer processing times and release dates and with the objective to minimize a regular integer-concave function $F(C_1, \ldots, C_n)$, there exists an optimal schedule where all interruptions occur at integral dates.*

Proof. The general idea for the proof is very similar to the one of Theorem 1. The only difference is in item (3) below.

1. Given an optimal preemptive schedule S, we construct a network G and a flow in this network. Show that the flow is feasible for this network.
2. On the set of feasible flows of maximum capacity in network G we find the optimal network minimizing the objective function. Since the polyhedron corresponding to the set of maximum flows in network G is integral and the optimal solution for a regular integer-concave objective function is attained at a node of the polytope, this implies the existence of an integral optimal solution (a flow in network G). This flow corresponds to an integral schedule S^* in which all interruptions occur at integral dates and which minimizes our objective function.
3. We finally prove that schedule S^* meets precedence constraints, and so, it is feasible and optimal.

Instead of presenting the complete proof, we just mention the points in which the proof differs from that of Theorem 1. First of all, network $G = (V, E)$ is slightly different from the one defined in the proof of Theorem 1. We define here the vertex set as $V = \{s\} \cup \{\tau\} \cup \{J_1, \dots, J_n\} \cup \varLambda \cup \mathcal{T}$, where the set $\varLambda$ contains a vertex v_{ij} for every pair of jobs J_i and J_j such that J_i precedes J_j in the precedence relations and both jobs share the same unit time interval $[t, t+1]$ for some $t \in \mathcal{T}$. The edge set E is defined as follows.

1. There are n edges of capacity p_j $(j = 1, \dots, n)$ connecting the source s with jobs $\{J_1, \dots, J_n\}$.
2. There are $|\mathcal{T}|$ edges of capacity m connecting vertices from the set $\mathcal{T}$ with the sink τ.
3. If J_i precedes J_j and for some $t \in \mathcal{T}$ we have $t < C_i(S) \leq \varGamma_j(S) < t+1$ (where $\varGamma_j(S)$ is the starting time of job J_j in schedule S), then we define three edges (J_i, v_{ij}), (J_j, v_{ij}) and (v_{ij}, t) of unit capacity. In this case we say that J_i and J_j compete for the same time interval $[t, t+1]$. Every job J_j has at most two time intervals where it competes with other jobs.
4. Let $I_j \subseteq \mathcal{T}$ be the set of "noncompetitive" time units for job J_j, i.e., $\lfloor \varGamma_j(S) \rfloor \leq t < C_j(S)$ and job J_j does not compete with any other job in the time interval $[t, t+1]$ for $t \in I_j$. For any time unit $t \in I_j$ we define an edge (J_j, t) of unit capacity.

The intuition behind the above definition is that we define nearly the same graph, with the only difference: if two jobs related by a precedence constraint are using the same time interval $[t, t+1]$ in a preemptive schedule, then we would like to create some capacity constraints preventing the concurrent unit assignment of such jobs to the same time unit t.

Given an optimal preemptive schedule S, we define the flow variables $\{x_{ij}\}$ on edges of graph G as follows. We send p_i units of flow from the source s into job vertex J_i for $i = 1, \dots, n$. After that we split the flow according to schedule S. If $x_{it}(S) > 0$, i.e., there is a nonzero amount of job J_i processed in the time interval $[t, t+1]$, we send $x_{it}(S)$ units of flow from vertex J_i to vertex $t \in \mathcal{T}$ if there is a direct edge (J_i, t). Otherwise, there exists either a successor or predecessor J_j of J_i processed in time unit t in schedule S. In this case we send $x_{it}(S)$ units of flow to the intermediate vertex $v_{ij} \in \Lambda$. All other flow variables are defined by the flow conservation constraint. This s-to-t flow is feasible since it satisfies the flow conservation and capacity constraints.

The proof that the flow defined above is feasible for network G is straightforward, and the arguments to prove item (2) are the same as those used for Theorem 1. So, it only remains to prove that any integral flow in network G corresponds to a feasible schedule which meets precedence constraints.

Indeed, if J_i precedes J_j, then all the time units $t \in \mathcal{T}$ to which there is a positive flow from vertex J_i are located before the time units to which there may exist a positive flow from vertex J_j (due to the properly defined set of edges of graph G). The only intersection of these two sets of "time" vertices is the "boundary" time unit t^* to which we have an edge from vertex v_{ij}. But due to the integrality of the flow only one of two jobs $\{J_i, J_j\}$ may produce a positive (in fact, unit) flow from vertex J_ν ($\nu \in \{i, j\}$) to vertex t^*. Thus, all time units to which job J_i has a nonnegative flow in network G are located strictly before the time units to which job J_j has a nonnegative flows, which means that precedence constraints are satisfied.

The remainder of the proof is similar to the one of Theorem 1.

3 Concluding Remarks

In this paper we establish the integrality property for the preemptive version of the parallel machine scheduling problem with chain precedence constraints and release dates and an arbitrary regular integer-concave objective function.

The integrality property implies the preemption redundancy for a wide class of scheduling problems with unit processing times. It follows that all such preemptive scheduling problems are equivalent to their non-preemptive counterparts from the complexity viewpoint. The equivalence provides a unified proof for many previously known complexity results. Another consequence is the establishing of strong NP-hardness of the $1|p_j = 1, chains, pmtn|\sum T_j$ problem and polynomial time solvability of the $P|p_j = 1, r_j, pmtn|\sum w_j T_j$ problem; the complexity status of both problems was open before.

References

[1] Baptiste, P., Brucker, P., Knust, S., Timkovsky, V.: Ten notes on equal-processing-time scheduling: at the frontiers of solvability in polynomial time. 4OR 2, 111–127 (2004)

[2] Baptiste, P., Carlier, J., Kononov, A., Queyranne, M., Sevastyanov, S., Sviridenko, M.: Structural Properties of Optimal Preemptive Schedules. To appear in Discrete Analysis and Operations Research (in Russian)

[3] Baptiste, P., Timkovsky, V.: On Preemption Redundancy in Scheduling Unit Processing Time Jobs on Two Parallel Machines. Operations Research Letters 28, 205–212 (2001)

[4] Brucker, P., Heitmann, S., Hurink, J.: How Useful are Preemptive Schedules? Operations Research Letters 31(2), 129–136 (2003)

[5] Du, J., Leung, J.Y.T.: Minimizing mean flow time in two-machine open shops and flow shops. J. Algorithms 14, 24–44 (1993)

[6] Du, J., Leung, Y., Young, G.: Scheduling chain-structured tasks to minimize makespan and mean flow time. Inform. and Comput. 92, 219–236 (1991)

[7] Gonzalez, T., Sahni, S.: Open Shop Scheduling to Minimize Finish Time. Journal of the ACM 23, 665–679 (1976)

[8] Gonzalez, T., Sahni, S.: Preemptive Scheduling of Uniform Processor Systems. Journal of the ACM 25, 92–101 (1978)

[9] Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. Annals of Discrete Mathematics 5, 287–326 (1979)

[10] Labetoulle, J., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Preemptive scheduling of uniform machines subject to release dates. In: Pulleyblank, W.R. (ed.) Progress in Combinatorial Optimization, pp. 245–261. Academic Press, London (1984)

[11] Lawler, E.L., Labetoulle, J.: On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming. Journal of the ACM 25, 612–619 (1978)

[12] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and Scheduling: Algorithms and Complexity. In: Graves, S.C., Rinnooy Kan, A.H.G., Zipkin, P.H. (eds.) Logistics of Production and Inventory. Handbooks in Operations Research and Management Science, vol. 4, pp. 445–522. North-Holland, Amsterdam (1993)

[13] Leung, J.Y.-T., Young, G.H.: Minimizing total tardiness on a single machine with precedence constraints. ORSA J. Comput. 2(4), 346–352 (1990)

[14] McNaughton, R.: Scheduling with deadlines and loss functions. Management Science 6, 1–12 (1959)

[15] Sauer, N., Stone, M.: Preemptive scheduling. In: Algorithms and order, Ottawa, ON, 1987, pp. 307–323. Kluwer Acad. Publ., Dordrecht (1989)

[16] Sauer, N., Stone, M.: Rational preemptive scheduling. Order 4(2), 195–206 (1987)

[17] Schrijver, A.: Theory of Linear and Integer Programming. Wiley, Chichester (1986)

[18] Tanaev, V.S., Gordon, V.S., Shafransky, Y.M.: Scheduling theory. Single-stage systems. Kluwer, Dordrecht (1994)

[19] Timkovsky, V.: Identical parallel machines vs. unit-time shops and preemptions vs. chains in scheduling complexity. European J. Oper. Res. 149, 355–376 (2003)

[20] Woeginger, G.: On the approximability of average completion time scheduling under precedence constraints. Discrete Appl. Math. 131(1), 237–252 (2003)

[21] http://www.mathematik.uni-osnabrueck.de/research/OR/class/

Characterizing the Existence of Optimal Proof Systems and Complete Sets for Promise Classes

Olaf Beyersdorff[1] and Zenon Sadowski[2]

[1] Institute of Theoretical Computer Science, Leibniz University Hannover, Germany
beyersdorff@thi.uni-hannover.de
[2] Institute of Mathematics, University of Białystok, Poland
sadowski@math.uwb.edu.pl

Abstract. In this paper we investigate the following two questions:

Q1: Do there exist optimal proof systems for a given language L?
Q2: Do there exist complete problems for a given promise class C?

For concrete languages L (such as TAUT or SAT) and concrete promise classes C (such as $\mathsf{NP}\cap\mathsf{coNP}$, UP, BPP, disjoint NP-pairs etc.), these questions have been intensively studied during the last years, and a number of characterizations have been obtained. Here we provide new characterizations for Q1 and Q2 that apply to almost all promise classes C and languages L, thus creating a unifying framework for the study of these practically relevant questions.

While questions Q1 and Q2 are left open by our results, we show that they receive affirmative answers when a small amount on advice is available in the underlying machine model. This continues a recent line of research on proof systems with advice started by Cook and Krajíček [6].

1 Introduction

A general proof system in the sense of Cook and Reckhow [7] can be understood as a nondeterministic guess-and-verify algorithm. The question whether there exist optimal or p-optimal proof systems essentially asks whether there exists the best such verification procedure. For practical purposes, such an optimal proof system would be extremely useful, as both the search for good verification algorithms as well as the quest for lower bounds to the proof size could concentrate on the optimal system. Thus the following question is of great significance:

Q1: Do there exist (p-)optimal proof systems for a given language L?

Posed by Krajíček and Pudlák [15], this question has remained unresolved for almost twenty years. Sufficient conditions were established by Krajíček and Pudlák [15] by $\mathsf{NE} = \mathsf{coNE}$ for the existence of optimal and $\mathsf{E} = \mathsf{NE}$ for p-optimal propositional proof systems, and these conditions were subsequently weakened by Köbler, Messner, and Torán [13]. Necessary conditions for a positive answer to Q1 are tightly linked to the following analogue of Q1 for promise complexity classes lacking an easy syntactic machine model:

Q2: Do there exist complete problems for a given promise class C?

Like the first question also Q2 has a long research record, dating back to the 80's when Kowalczyk [14] and Hartmanis and Hemachandra [12] considered this question for $\mathsf{NP} \cap \mathsf{coNP}$ and UP. This research agenda continues to recent days where, due to cryptographic and proof-theoretic applications, disjoint NP-pairs have been intensively studied (cf. [8,9,11,2] and [10] for a survey).

As many computational tasks are formulated as function problems [20], it is also interesting to extend Q2 to function classes. In this formulation Q1 becomes a special case of Q2 because all proof systems for a given language can be understood as a promise function class in which complete functions correspond to p-optimal proof systems. In fact, Köbler, Messner, and Torán [13] have shown that, with respect to Q2, proof systems provide the most difficult instances among all promise classes, i.e., a positive answer to Q1 implies a positive answer for Q2 for many choices of L and C.

In the present paper we continue this line of research. While Köbler, Messner, and Torán [13] focused on the implication Q1 $\Rightarrow$ Q2, we provide new characterizations for both Q1 and Q2. In fact, from these characterizations we can also easily read off the implication Q1 $\Rightarrow$ Q2 (under suitable assumptions), thus in addition, we provide alternative proofs for some results of [13]. Köbler, Messner, and Torán used the notion of a test set to measure the complexity of the promise. Here we pursue a different but related approach by representing the promise in a language L and then using a proof system for L to verify the promise. On the propositional level, such representations have been successfully used to express the consistency of propositional proof systems (known as the reflection principle, cf. [5,15]) or the disjointness of NP-pairs [16,2]. We create a unifying framework which generalizes these methods to arbitrary languages.

We will now describe in more detail our results and the organization of the paper. After developing the notion of representations in Sects. 2 and 3 we examine Q1 in Sect. 4 where we prove that a language L has a p-optimal proof system if and only if all polynomial-time computable subsets of L are recursively enumerable. A similar characterization also holds for the existence of optimal proof systems. This widely generalizes previous results from [18] for propositional proof systems and provides interesting characterizations for a number of applications like the graph isomorphism and automorphism problems.

In Sect. 5 we proceed with question Q2 where we discuss a characterization of Q2 in terms of uniform enumerations of promise obeying machines. Section 6 then contains our results on the connections between Q1 and Q2. We show that, under suitable assumptions, a promise class C has complete problems if and only if there exists a proof system for some language L in which C is representable. This also yields a general method to show the equivalence of reductions of varying strength with respect to Q2. In addition, we obtain that L has a p-optimal proof system if and only if every promise class expressible in L has a complete set or function. Different versions of these results hold for both optimality and p-optimality. We also apply these general theorems to concrete promise classes like UP, $\mathsf{NP} \cap \mathsf{coNP}$, and disjoint NP-pairs.

Finally, in Sect. 7 we show that the relation between proof systems and promise classes also holds in the presence of advice. Employing recent advances of Cook and Krajíček [6] who show that optimal propositional proof systems exist which use only one bit of advice, we obtain complete sets for a large number of promise classes when advice is available.

Due to space restrictions we sketch or omit proofs in this extended abstract.

2 Preliminaries

We assume basic familiarity with complexity classes (cf. [1]). Our basic model of computation are polynomial-time Turing machines and transducers. Tacitly we assume these machines to be suitably encoded by strings. We also assume that they always have a polynomial-time clock attached bounding their running time such that this running time is easy to detect from the code of the machine.

For a language L and a complexity class C, the set of all C-*easy subsets* of L consists of all sets $A \subseteq L$ with $A \in \mathsf{C}$. A class C of languages has a *recursive* P-*presentation* (resp. NP-presentation) if there exists a recursively enumerable list $N_1, N_2, \dots$ of (non-)deterministic polynomial-time clocked Turing machines such that $L(N_i) \in \mathsf{C}$ for $i \in \mathbb{N}$, and, conversely, for each $A \in \mathsf{C}$ there exists an index i with $A \subseteq L(N_i)$. In this definition, it would also be natural to replace $A \subseteq L(N_i)$ by the stronger requirement $A = L(N_i)$, but the weaker concept suffices for our purpose.

Proof Systems. Cook and Reckhow [7] defined the notion of a *proof system* for a language L quite generally as a polynomial-time computable function f with range L. A string w with $f(w) = x$ is called an f-proof for $x \in L$. By $f \vdash_{\leq m} x$ we indicate that x has an f-proof of size $\leq m$. For a subset $A \subseteq L$ we write $f \vdash_* A$ if there is a polynomial p such that $f \vdash_{\leq p(|x|)} x$ for all $x \in A$.

Proof systems are compared by simulations [7,15]. If f and g are proof systems for L, we say that g *simulates* f (denoted $f \leq g$), if there exists a polynomial p such that for all $x \in L$ and f-proofs w of x there is a g-proof w' of x with $|w'| \leq p(|w|)$. If such a proof w' can even be computed from w in polynomial time, we say that g *p-simulates* f (denoted $f \leq_p g$). A proof system for L is called *(p-)optimal* if it (p-)simulates all proof systems for L.

Promise Classes. Following the approach of Köbler, Messner, and Torán [13], we define promise classes in a very general way. A promise R is described as a binary predicate between nondeterministic polynomial-time Turing machines N and strings x, i.e., $R(N, x)$ means that N obeys promise R on input x. A machine N is called an *R-machine* if N obeys R on any input $x \in \Sigma^*$. Given a promise predicate R, we define the language class $\mathsf{C}_R = \{L(N) \mid N \text{ is an R-machine}\}$ and call it the promise class generated by R. Instead of R-machines we will also speak of C_R-machines. Similarly, we define function promise classes by replacing $L(N)$ by the function computed by N (cf. [13]). For functions we use the following variant of many-one reductions (cf. [13]): $f \leq g$ if there exists a polynomial-time computable function t such that $f(x) = g(t(x))$ for all x in the domain of f.

In this general framework it is natural to impose further restrictions on promise classes. One assumption which we will make throughout the paper is the presence of *universal machines*, i.e., we only consider promise conditions R such that there exists a universal machine U_R which, given an R-machine N, input x, and time bound 0^m, efficiently simulates $N(x)$ for m steps such that U_R obeys promise R on $\langle N, x, 0^m \rangle$.

Occasionally, we will need that C-machines can perform nondeterministic polynomial-time computations without violating the promise. We make this precise via the following notion from [13]: for a complexity class A and a promise class C defined via promise R, we say that A-*assertions are useful for* C if for any language $A \in \mathsf{A}$ and any nondeterministic polynomial-time Turing machine N the following holds: if N obeys promise R on any $x \in A$, then there exists a language $C \in \mathsf{C}$ such that $C \cap A = L(N) \cap A$. A similar definition also applies for function classes. Throughout this paper we will only consider promise classes C for which P-assertions are useful. If also NP-assertions are useful for C, then we say that C *can use nondeterminism.*

The set of all proof systems for a language L is an example for a promise function class, where the promise for a given function f is $rng(f) = L$. We define a larger class $PS(L)$ where we only concentrate on correctness but not on completeness of proof systems. This is made precise in the following definition.

Definition 1. *For a language L, the promise function class $PS(L)$ contains all polynomial-time computable functions f with $rng(f) \subseteq L$.*

3 Representations

In order to verify a promise, we need appropriate encodings of promise conditions. In the next definition we explain how a promise condition for a machine can be expressed in an arbitrary language.

Definition 2. *A promise R is* expressible *in a language L if there exists a polynomial-time computable function $corr : \Sigma^* \times \Sigma^* \times 0^* \to \Sigma^*$ such that the following conditions hold:*

1. Correctness: *For every Turing machine N, for every $x \in \Sigma^*$ and $m \in \mathbb{N}$, if $corr(x, N, 0^m) \in L$, then N obeys promise R on input x.*
2. Completeness: *For every R-machine N with polynomial time bound p, the set $Correct(N) = \{corr(x, N, 0^{p(|x|)}) \mid x \in \Sigma^* \}$ is a subset of L.*
3. Local recognizability: *For every Turing machine N, the set $Correct(N)$ is polynomial-time decidable.*

We say that the promise class C generated by R is expressible in L if R is expressible in L. If the elements $corr(x, N, 0^m)$ only depend on $|x|$, N, and m, but not on x, we say that C is expressible in L by a length-depending *promise.*

This definition applies to both language and function promise classes. One of the most important applications for the above concept of expressibility is to use

$L = \mathrm{TAUT}$. Expressing promise conditions by propositional tautologies is a well known approach with a long history. For propositional proof systems, leading to the promise function class $PS(\mathrm{TAUT})$, propositional expressions are constructed via the reflection principle of the proof system (cf. [5,15]). Propositional expressions have also been used for other promise classes like disjoint NP-pairs and its generalizations [2,3]. Typically, these expressions are even length depending. We remark that Köbler, Messner, and Torán [13] have used a related approach, namely the notion of a test set, to measure the complexity of promise conditions.

As a first example, consider the set of all P-easy subsets of a language L. The next lemma shows that this promise class is always expressible in L.

Lemma 1. *For every language L, the P-easy subsets of L are expressible in L.*

Using expressibility of a promise class in a language L, we can verify the promise for a given machine with the help of short proofs in some proof system for L. This leads to the following concept:

Definition 3. *Let C be a promise class which is expressible in a language L. Let further A be a language from C and P be a proof system for L. We say that A is* representable in P *if there exists a C-machine N for A such that $P \vdash_* \mathit{Correct}(N)$. If these P-proofs of $\mathit{corr}(x, N, 0^{p(|x|)})$ can even be constructed from input x in polynomial time, then we say that A is* p-representable in P.

Furthermore, if every language $A \in \mathsf{C}$ is (p-)representable in P, then we say that C is (p-)representable in P.

Intuitively, representability of A in P means that we have short P-proofs of the promise condition of A (with respect to some C-machine for A). Given a proof system P for L and a promise class C which is expressible in L, it makes sense to consider the subclass of all languages or functions from C which are representable in P. This leads to the following definition:

Definition 4. *For a promise class C expressible in a language L and a proof system P for L, let $\mathsf{C}(P)$ denote the class of all $A \in \mathsf{C}$ which are representable in P.*

Note that for each $A \in \mathsf{C}$ there exists some proof system P for L such that $A \in \mathsf{C}(P)$, but in general $\mathsf{C}(P)$ will be a strict subclass of C which enlarges for stronger proof systems. It is, of course, interesting to ask whether these subclasses $\mathsf{C}(P)$ have sufficiently good properties. In particular, it is desirable that $\mathsf{C}(P)$ is closed under reductions. Therefore, we make the following definition:

Definition 5. *A promise class C is* provably closed under a reduction $\le_R$ in L *if C is expressible in L and for each proof system P for L there exists a proof system P' for L such that $P \le P'$ and for all $A \in \mathsf{C}$ and $B \in \mathsf{C}(P')$, $A \le_R B$ implies $A \in \mathsf{C}(P')$.*

We remark that provable closure of C under $\le_R$ is a rather weak notion as it does not even imply closure of C under $\le_R$ in the ordinary sense (because of the restriction $A \in \mathsf{C}$). Also we do not require each subclass $\mathsf{C}(P)$ to be closed under $\le_R$, but that for each proof system P this holds for some stronger system P'.

This is a sensible requirement, because proof systems for L can be defined quite arbitrarily, and closure of $\mathsf{C}(P)$ typically requires additional assumptions on P (cf. [2] where provable closure of the class of disjoint NP-pairs under different reductions is shown). In fact, it is not difficult to construct counterexamples:

Proposition 1. *Let C be a promise class which is expressible in a language L and let $\leq_R$ be a reduction for C. Let further P be a proof system for L such that there exist $A, B \in \mathsf{C} \setminus \mathsf{C}(P)$ with $A \leq_R B$. Then there exists a proof system $P' \geq P$ such that $\mathsf{C}(P')$ is not closed under $\leq_R$.*

4 Optimal Proof Systems and Easy Subsets

In this section we search for characterizations for the existence of optimal or even p-optimal proof systems for arbitrary languages L (Question Q1) and apply these results to concrete choices for L. We start with a criterion for the existence of p-optimal proof systems.

Theorem 1. *Let L be a language such that $PS(L)$ is expressible in L. Then L has a p-optimal proof system if and only if the P-easy subsets of L have a recursive P-presentation.*

Proof (Idea). For the forward direction, we observe that every P-easy subset of L has short proofs in some proof system for L. These proofs are translated into short proofs in the p-optimal proof system by some polynomial-time Turing transducer. Thus, by enumerating all polynomial-time clocked Turing transducers, we can construct a recursive P-presentation of all P-easy subsets of L.

Conversely, we construct a p-optimal proof system P_{opt} in the following way. A P_{opt}-proof of a is of the form $\langle \pi, P, \mathit{certificate} \rangle$, where P is a polynomial-time clocked transducer such that $P(\pi) = a$. The certificate assures that $P(\pi) \in L$. It follows from expressibility of $PS(L)$ in L that $\mathit{Correct}(P)$ is a P-easy subset of L if and only if P produces only elements from L (for any input). Hence, we can use P-presentability of the P-easy subsets of L to produce certificates. $\square$

By a similar argument we can provide two characterizations for the existence of optimal proof systems.

Theorem 2. *Let L be a language such that $PS(L)$ is expressible in L. Then the following conditions are equivalent:*

1. *There exists an optimal proof system for L.*
2. *The NP-easy subsets of L have a recursive NP-presentation.*
3. *The P-easy subsets of L have a recursive NP-presentation.*

Given these general results, it is interesting to ask for which languages L the set $PS(L)$ is expressible in L. Our next lemma provides sufficient conditions:

Lemma 2. *Let L be a language fulfilling the following two conditions:*

1. *Natural numbers can be encoded by elements of L, i.e., there exists an injective function $Num : \mathbb{N} \to L$ which is both computable and invertible in polynomial time.*
2. *L possesses an AND-function, i.e., there exists a function $AND : \Sigma^* \times \Sigma^* \to \Sigma^*$ which is both polynomial-time computable and polynomial-time invertible such that for all $x, y \in \Sigma^*$, $AND(x, y) \in L$ if and only if $x \in L$ and $y \in L$.*

Then $PS(L)$ is expressible in L.

Using this lemma we can show L-expressibility of $PS(L)$ for many interesting choices of L:

Proposition 2. *For any of the following languages L, the set $PS(L)$ is expressible in L:*

- SAT_i *for $i \in \mathbb{N}$ (the satisfiability problem for quantified propositional formulas with i quantifier alternations, starting with existential quantifiers),*
- TAUT_i *for $i \in \mathbb{N}$ (quantified propositional tautologies with i quantifier alternations, starting with universal quantifiers),*
- QBF *(quantified propositional tautologies),*
- *the graph isomorphism problem* GI*, its complement* $\overline{\mathrm{GI}}$*, and the complement* $\overline{\mathrm{GA}}$ *of the graph automorphism problem.*

For GI, which like any problem in NP has an optimal proof system, we obtain the following characterization on the existence of a p-optimal proof system.

Corollary 1. GI *has a p-optimal proof system if and only if there exists a recursive* P*-presentation of all polynomial-time computable subsets of* GI*.*

Let us remark that in Lemma 2, instead of an AND-function we could also use a padding function for L. In this way we obtain a similar result as Corollary 1 for GA (which is not known to possess an AND-function).

5 Complete Sets and Enumerations

In this section we consider the question Q2, asking whether language or function promise classes have complete sets or functions. There is a long history of equating complete sets and recursive enumerations of machines. The following result essentially stems from [13], but particular cases of the theorem have been been previously obtained, namely for $\mathsf{NP} \cap \mathsf{coNP}$ by Kowalczyk [14], for UP by Hartmanis and Hemachandra [12], and, more recently, for disjoint NP-pairs by Glaßer, Selman, and Sengupta [8]. We just formulate the theorem for language classes, but a similar result also holds for promise function classes.

Theorem 3 (Köbler, Messner, Torán [13]). *Let C be a promise class which is closed under many-one reductions. Then C has a many-one complete problem if and only if there exists a recursive enumeration $(N_i)_{i \geq 0}$ of C-machines such that $\mathsf{C} = \{L(N_i) \mid i \geq 0\}$.*

Let us note that in the proof of the forward implication of Theorem 3, the hypothesis that C is closed under many-one reductions seems indeed crucial. Namely, if C consists of all P-easy subsets of TAUT, then C trivially contains a many-one complete set. On the other hand, a recursive enumeration of C-machines as in Theorem 3 is rather unlikely to exist, as this would imply the existence of a p-optimal propositional proof system by Theorem 1. But of course, the P-easy subsets of TAUT are not closed under many-one reductions.

6 Optimal Proof Systems and Complete Sets

Now we are ready to analyse the relations between our central questions Q1 and Q2 on the existence of optimal proof systems for languages L and the existence of complete sets for promise classes C. While Köbler, Messner, and Torán [13] have shown that for many natural choices of L and C, a positive answer to Q1 implies a positive answer to Q2, we will provide here a number of characterizations involving both questions. In particular, these characterizations will also yield the above mentioned relation between Q1 and Q2 for concrete applications.

Our first result characterizes the existence of complete sets for a promise class C by the representability of C in a proof system.

Theorem 4. *Let C be a promise language (or function) class which can use nondeterminism and let L be a language such that C is provably closed under many-one reductions in L. Then C has a many-one complete language (or function) if and only if there exists a proof system for L in which C is representable.*

Proof (Idea). For the forward implication, we code a many-one complete language A for C into some proof system P for L. By provable closure under reductions, L has a proof system $P' \geq P$ such that $\mathsf{C}(P')$ is closed under many-one reductions. As $A \in \mathsf{C}(P')$ and A is many-one complete for C, we get $\mathsf{C}(P') = \mathsf{C}$.

Conversely, let P be a proof system for L in which C is representable. Using the universal machine for C, we construct a complete set for C by simulating C-machines N on their inputs. But before we start such a simulation, we check the promise of N by guessing short P-proofs for $\mathit{Correct}(N)$. For this last step we need that C can use nondeterminism. $\square$

For promise classes not using nondeterminism we obtain the following result:

Theorem 5. *Let C be a promise language (or function) class which is closed under many-one reductions and let L be a language such that C is expressible in L. Then C has a many-one complete language (or function) if and only if L has a proof system in which C is p-representable.*

Let us mention some applications of this result. The promise class DisjNP of disjoint NP-pairs and the class UP are expressible in TAUT, and the class $\mathsf{NP} \cap \mathsf{coNP}$ is expressible in QBF (cf. [2,13,17,19]). Hence we obtain the following corollary exemplifying our theorem.

Corollary 2

1. *Complete disjoint* NP*-pairs exist if and only if* TAUT *has a proof system in which* DisjNP *is p-representable (if and only if* TAUT *has a proof system in which* DisjNP *is representable).*
2. UP *has a complete language if and only if* TAUT *has a proof system in which* UP *is p-representable.*
3. $\mathsf{NP} \cap \mathsf{coNP}$ *has a complete language if and only if* QBF *has a proof system in which* $\mathsf{NP} \cap \mathsf{coNP}$ *is p-representable.*

Theorem 4 also allows to derive results which show that the question of the existence of complete problems for C does not depend on the strength of the underlying reduction. This can be done as in the following corollary:

Corollary 3. *Let $\leq$ and $\leq'$ be two reductions which are refined by many-one reductions. Assume further that* C *can use nondeterminism and is both provably closed under $\leq$ and $\leq'$ in some language L. Then* C *has a $\leq$-complete problem if and only if* C *has a $\leq'$-complete problem.*

In this way it can be shown, for example, that the question of the existence of complete disjoint NP-pairs is equivalent for reductions ranging from strong many-one reductions to smart Turing reductions (cf. [8,2]).

Our next result shows that question Q1 on the existence of p-optimal proof systems for a language L can be characterized by a "universally quantified" version of the condition from Theorem 5. Further, Q1 is even equivalent to the existence of complete sets for all promise classes representable in L:

Theorem 6. *Let L be a language such that $PS(L)$ is expressible in L. Then the following conditions are equivalent:*

1. *There exists a p-optimal proof system for L.*
2. *There exists a proof system for L in which any promise class which is expressible in L is p-representable.*
3. *There exists a proof system for L in which the class of all* P*-easy subsets of L is p-representable.*
4. *Every promise language and function class which is expressible in L has a many-one complete language or function.*

Proof (Sketch). The proof is structured into the implications 1 $\Rightarrow$ 2 $\Rightarrow$ 3 $\Rightarrow$ 1 and 2 $\Rightarrow$ 4 $\Rightarrow$ 1. For 1 $\Rightarrow$ 2, let P be a p-optimal proof system for L and let C be a promise class expressible in L. For each $A \in \mathsf{C}$ and each C-machine N for A we can construct a proof system P' with short P'-proofs of $Correct(N)$. Translating these proofs into the p-optimal system P, we obtain $A \in \mathsf{C}(P)$.

Implication 2 $\Rightarrow$ 3 follows from Lemma 1. For the direction 3 $\Rightarrow$ 1, we need to construct from item 3 a recursive P-presentation of all P-easy subsets of L as in Theorem 1. This in turn yields a p-optimal proof system for L.

The equivalence between items 2 and 4 is the mentioned "universally quantified" version of Theorem 5. Finally, for 4 $\Rightarrow$ 1 we use the assumption of expressibility of $PS(L)$ in L. As $PS(L)$ is a promise function class, item 4 guarantees the existence of a many-one complete function for $PS(L)$, which coincides with the notion of a p-optimal proof system for L. □

The next theorem contains a similar statement for optimal proof systems.

Theorem 7. *Let L be a language such that $PS(L)$ is expressible in L. Then the following conditions are equivalent:*

1. *There exists an optimal proof system for L.*
2. *L has a proof system P such that every promise class which is expressible in L is representable in the system P.*
3. *L has a proof system in which all P-easy subsets of L are representable.*

Combining Theorems 4 and 7 we obtain the following corollary which is essentially contained in [13].

Corollary 4. *Let L be a language. If L has an optimal proof system, then any promise language or function class C which is expressible in L and which can use nondeterminism has a complete language or function.*

As the proof of the backward implication of Theorem 4 does not use provable closure of C under reductions in L, we can formulate Corollary 4 without this assumption.

Comparing Theorem 6 and Corollary 4, it is apparent that while we could prove the equivalence of the existence of p-optimal proof systems for L and complete problems for all promise classes expressible in L (Theorem 6), we did not obtain this equivalence for optimal proof systems (cf. Corollary 4). The reason is that $PS(L)$, considered as a promise function class, does not seem to have the property to use nondeterminism, because otherwise, the existence of an optimal proof system for L would already imply the existence of a p-optimal proof system for L. We can even obtain a slightly stronger result:

Proposition 3. *If $PS(\mathrm{SAT})$ can use nondeterminism, then every language with an optimal proof system also has a p-optimal proof system.*

Proof. Assume that $PS(\mathrm{SAT})$ can use nondeterminism. By Proposition 2, the class $PS(\mathrm{SAT})$ is expressible in SAT. As SAT has an optimal proof system, Corollary 4 now yields a complete function for $PS(\mathrm{SAT})$ which coincides with the notion of a p-optimal proof system for SAT. From this we conclude that every language with an optimal proof system also has a p-optimal proof system by a result from [3]. □

7 Optimal Proof Systems with Advice

Whether or not there exist optimal proof systems or complete sets for promise classes remains unanswered by our results above. Hence, our central questions Q1 and Q2 remain open. As these problems have been open for more than twenty years by now, many researchers tend to believe in a negative answer (of course, this is arguable, but in the algorithmic world negative results are usually harder to obtain than positive ones).

Recently, Cook and Krajíček [6] have introduced the concept of propositional proof systems with advice which seems to yield a strictly more powerful model than the classical Cook-Reckhow setting. Surprisingly, Cook and Krajíček [6] have shown that in the presence of advice, optimal propositional proof systems exist (cf. also [4] for a generalization to arbitrary languages). Our next result shows that the relation between optimal proof systems and complete sets for promise classes can be transferred to the advice setting. Thus we derive from Cook and Krajíček's results the following strong information on complete problems in the presence of advice.

Theorem 8. *Let* C *be a promise complexity class and let* L *be a language such that* C *is expressible in* L *by a length-depending promise. Then* $\mathsf{C}/1$ *contains a problem (or function) using one bit of advice which is many-one hard for* C.

Proof (Sketch). Let $\langle\cdot,\dots,\cdot\rangle$ be a polynomial-time computable length-injective tupling function. We now define the problem (or function) A_{C} with one advice bit which will be many-one hard for C. Inputs are of the form $\langle x, 0^N, 0^m\rangle$ where x is the input, 0^N is the unary encoding of a Turing machine N, and 0^m is the time bound for N. At such an input, A_{C} first computes the string $\mathit{corr}(x, N, 0^m)$. Then A_{C} uses its advice bit to verify whether or not $\mathit{corr}(x, N, 0^m)$ is in L (for this step we could have also used the optimal proof system for L with one bit of advice, cf. [6,4]). If $\mathit{corr}(x, N, 0^m) \in L$, then A_{C} simulates N on input x for at most m steps and produces the corresponding output (in case the simulation does not terminate it rejects or outputs some fixed element). As $\langle\cdot,\dots,\cdot\rangle$ is length injective and corr is length depending, the element $\mathit{corr}(x, N, 0^m)$ is uniquely determined by $|\langle x, 0^N, 0^m\rangle|$ and therefore the advice bit of A_{C} can in fact refer to $\mathit{corr}(x, N, 0^m)$.

If A is a problem (or function) from C and N is a C-machine for A with polynomial running time p, then A many-one reduces to A_{C} via $x \mapsto \langle x, 0^N, 0^{p(|x|)}\rangle$. Hence A_{C} is many-one hard for C. □

Let us state a concrete application of this general result. As disjoint NP-pairs are expressible in TAUT by a length-depending promise [2], we obtain:

Corollary 5. *There exist a disjoint pair* (A, B) *and a sequence* $(a_n)_{n\in\mathbb{N}}$ *with the following properties:*

1. *A and B are computable in nondeterministic polynomial time with advice a_n for inputs of length n.*
2. *The set $\{\langle a_n, 0^n\rangle \mid n \in \mathbb{N}\}$ is computable in* coNP.
3. *Every disjoint* NP*-pair is polynomial-time many-one reducible to* (A, B).

Acknowledgements. We thank the anonymous referees for helpful comments and detailed suggestions on how to improve this paper.

References

1. Balcázar, J.L., Díaz, J., Gabarró, J.: Structural Complexity I. Springer, Heidelberg (1988)
2. Beyersdorff, O.: Classes of representable disjoint NP-pairs. Theoretical Computer Science 377(1-3), 93–109 (2007)
3. Beyersdorff, O., Köbler, J., Messner, J.: Nondeterministic functions and the existence of optimal proof systems. Theoretical Computer Science (in press)
4. Beyersdorff, O., Köbler, J., Müller, S.: Nondeterministic instance complexity and proof systems with advice. In: Proc. 3rd International Conference on Language and Automata Theory and Applications. LNCS, vol. 5457, pp. 164–175. Springer, Heidelberg (2009)
5. Cook, S.A.: Feasibly constructive proofs and the propositional calculus. In: Proc. 7th Annual ACM Symposium on Theory of Computing, pp. 83–97 (1975)
6. Cook, S.A., Krajíček, J.: Consequences of the provability of NP $\subseteq$ P/poly. The Journal of Symbolic Logic 72(4), 1353–1371 (2007)
7. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. The Journal of Symbolic Logic 44(1), 36–50 (1979)
8. Glaßer, C., Selman, A.L., Sengupta, S.: Reductions between disjoint NP-pairs. Information and Computation 200(2), 247–267 (2005)
9. Glaßer, C., Selman, A.L., Sengupta, S., Zhang, L.: Disjoint NP-pairs. SIAM Journal on Computing 33(6), 1369–1416 (2004)
10. Glaßer, C., Selman, A.L., Zhang, L.: Survey of disjoint NP-pairs and relations to propositional proof systems. In: Goldreich, O., Rosenberg, A.L., Selman, A.L. (eds.) Essays in Theoretical Computer Science in Memory of Shimon Even, pp. 241–253. Springer, Heidelberg (2006)
11. Glaßer, C., Selman, A.L., Zhang, L.: Canonical disjoint NP-pairs of propositional proof systems. Theoretical Computer Science 370(1-3), 60–73 (2007)
12. Hartmanis, J., Hemachandra, L.A.: Complexity classes without machines: On complete languages for UP. Theoretical Computer Science 58, 129–142 (1988)
13. Köbler, J., Messner, J., Torán, J.: Optimal proof systems imply complete sets for promise classes. Information and Computation 184(1), 71–92 (2003)
14. Kowalczyk, W.: Some connections between representability of complexity classes and the power of formal systems of reasoning. In: Chytil, M.P., Koubek, V. (eds.) MFCS 1984. LNCS, vol. 176, pp. 364–369. Springer, Heidelberg (1984)
15. Krajíček, J., Pudlák, P.: Propositional proof systems, the consistency of first order theories and the complexity of computations. The Journal of Symbolic Logic 54(3), 1063–1079 (1989)
16. Krajíček, J., Pudlák, P.: Some consequences of cryptographical conjectures for S_2^1 and EF. Information and Computation 140(1), 82–94 (1998)
17. Sadowski, Z.: On an optimal quantified propositional proof system and a complete language for NP $\cap$ co-NP. In: Chlebus, B.S., Czaja, L. (eds.) FCT 1997. LNCS, vol. 1279, pp. 423–428. Springer, Heidelberg (1997)
18. Sadowski, Z.: On an optimal propositional proof system and the structure of easy subsets of TAUT. Theoretical Computer Science 288(1), 181–193 (2002)
19. Sadowski, Z.: Optimal proof systems and complete languages. In: Local Proc. 4th Conference on Computability in Europe, pp. 407–414. University of Athens (2008)
20. Selman, A.L.: Much ado about functions. In: Proc. 11th Annual IEEE Conference on Computational Complexity, pp. 198–212 (1996)

k-SAT Is No Harder Than Decision-Unique-k-SAT

Chris Calabro and Ramamohan Paturi[⋆]

University of California, San Diego
La Jolla, CA, USA
{ccalabro,paturi}@cs.ucsd.edu

Abstract. We resolve an open question by [3]: the exponential complexity of deciding whether a k-CNF has a solution is the same as that of deciding whether it has exactly one solution, both when it is promised and when it is not promised that the input formula has a solution. We also show that this has the same exponential complexity as deciding whether a given variable is backbone (i.e. forced to a particular value), given the promise that there is a solution. We show similar results for True Quantified Boolean Formulas in k-CNF, k-Hitting Set (and therefore Vertex Cover), k-Hypergraph Independent Set (and therefore Independent Set), Max-k-SAT, Min-k-SAT, and 0-1 Integer Programming with inequalities and k-wide constraints.

Keywords: k-SAT, unique satisfiability, exponential complexity, quantified Boolean formulas, hitting set, independent set.

1 Introduction

For a problem $L = \{x \mid \exists y\, R(x, y)\}$ where R is some relation, define the *solutions* of x as $\mathrm{sol}(x) = \{y \mid R(x, y)\}$. Define Decision-Unique-L (or DU-L) to be the problem of deciding whether the input x has $|\mathrm{sol}(x)| = 1$; Unique-L (or U-L) to be DU-L but with the promise that $|\mathrm{sol}(x)| \leq 1$; and Satisfiable-Unique-L (or SU-L) to be DU-L but with the promise that $|\mathrm{sol}(x)| \geq 1$.

Note that these definitions depend on R being understood from context, since alternative formulations of a problem could lead to alternative definitions of DU-L, U-L, SU-L, but this will rarely cause ambiguity. E.g. DU-k-SAT will be the problem of deciding whether a given k-CNF has a unique satisfying assignment and DU-IS will the the problem of deciding whether a given graph has a unique independent set of size at most a given integer.

For a problem P parameterized by parameter n and solvable in time poly $(|x|)2^{O(n)}$ on input x by a randomized algorithm with error $\leq \frac{1}{3}$, define the *exponential complexity* c_P of P to be the infimum of those c such that P is solvable in time poly$(|x|)2^{cn}$ by a randomized algorithm with error $\leq \frac{1}{3}$. For each problem considered here except for the weighted satisfiability variants, $|x| \leq \mathrm{poly}(n)$,

[⋆] Research supported by NSF Award CCF-0515332.

so that the poly($|x|$) can usually be dropped from the above definition. Also, for problems involving formulas (graphs), we will implicitly take the parameter to be the number of variables (nodes). We can similarly define the *deterministic* exponential complexity dc_P of a problem P by eliminating the word 'randomized' in the above definition. Clearly $c_P \leq dc_P$. The reverse inequality is not known, and even a strong hypothesis like P = BPP does not obviously imply it since, for all we know, derandomization may square running time, assuming it can be done at all.

A Boolean formula is in *conjunctive normal form* (CNF) iff it is a conjunction of disjunctions of literals - i.e. an AND of ORs of variables or their negations. A CNF is a k-CNF iff each disjunction contains $\leq k$ literals. SAT is the problem of deciding whether a given CNF has an assignment to the variables that makes the formula true. k-SAT further restricts the input to k-CNF.

While $c_{k\text{-SAT}} \leq c_{\text{DU-}k\,+\,1\text{-SAT}}$ is obvious (add a new variable z to each clause and n new clauses $(z \to x_i)$ for each i), the inequality is probably not tight since, assuming the exponential time hypothesis (that $c_{3\text{-SAT}} > 0$), $c_{k\text{-SAT}}$ increases infinitely often as a function of k [6], and it would be surprising if it did not increase with each k. [3] conjectured that the deterministic exponential complexity of k-SAT and DU-k-SAT are the same, i.e. that

$$dc_{k\text{-SAT}} = dc_{\text{DU-}k\text{-SAT}} \ . \tag{1}$$

The current authors, in [1], incorrectly cited [3], claiming that they had shown (1) rather than merely conjectured it. Here we remedy this - we prove (1) by giving a deterministic polytime Turing reduction from k-SAT to SU-k-SAT such that on an input with n variables, the oracle is called only on formulas with $\leq n+O(1)$ variables. Since $dc_{\text{SU-}k\text{-SAT}} \leq dc_{\text{DU-}k\text{-SAT}}$, and self-reducibility implies $dc_{\text{DU-}k\text{-SAT}} \leq dc_{k\text{-SAT}}$, we have

$$dc_{k\text{-SAT}} = dc_{\text{SU-}k\text{-SAT}} = dc_{\text{DU-}k\text{-SAT}} \ .$$

Via standard error reduction techniques (lemma 1), such a reduction also shows

$$c_{k\text{-SAT}} = c_{\text{SU-}k\text{-SAT}} = c_{\text{DU-}k\text{-SAT}} \ . \tag{2}$$

It is not known whether $c_{k\text{-SAT}} = dc_{k\text{-SAT}}$, but it seems like the answer is 'no'. E.g. [7] gives a nice table contrasting the history of the development of deterministic vs. randomized k-SAT algorithms, with the more recent randomized algorithms with significantly less exponential complexity than their deterministic counterparts.

We also use the same technique to show similar results for True Quantified Boolean Formulas in k-CNF, k-Hitting Set, Vertex Cover, k-Hypergraph Independent Set, Independent Set, Max-k-SAT, Min-k-SAT, and 0-1 Integer Programming with inequalities and k-wide constraints.

Our main contribution is the following technique for efficiently reducing a problem P to SU-P: view an instance ϕ of P as a set of constraints $C_1, \ldots, C_m$ over a set of variables $x_1, \ldots, x_n$. Starting with the empty set of constraints, add one constraint of ϕ at a time to a list and maintain the invariant that we know a solution to the list. To add a new constraint C_i to the list, construct some gadget that conditionally turns on C_i in the case that the variables are assigned differently than the solution already known for the $i-1$ case. Then, using this gadget, call the oracle repeatedly to learn a solution for the i case. Knowing a solution to the $i-1$ case allows us to encode such gadgets significantly more efficiently than we would otherwise know how to do. Although this idea is the basis for each reduction we consider, there are significant problem-specific details that prevent us from factoring out large common parts of the proofs.

The current work does not completely resolve the question of the exponential complexity of satisfiability vs. that of unique satisfiability since it still does not show $c_{k\text{-SAT}} = c_{\text{U-}k\text{-SAT}}$, though we strongly suspect it, as equality holds in the limit as $k \to \infty$ [1].

The reduction of Valiant and Vazirani [10] falls a bit shorter in proving this statement about k-SAT, but is quite general: (the intersection of a nonempty set system of size about 2^m on n variables with the solutions to a system of approximately m random linear equations over GF_2) has size exactly 1 with probability $\Omega(1)$. This idea shows the NP-hardness of U-k-SAT under randomized reductions but not that it has the same exponential complexity as k-SAT because expressing an $m \times n$ random linear system in k-CNF seems to require a quadratic increase in the number of variables. However this idea can be used to show that $c_P = c_{\text{U-}P}$ for any problem P where intersecting such a linear system could be expressed with only a $1 + o(1)$ factor increase in the complexity parameter, such as in CircuitSAT.

Paper Organization. §2 gives the previous work on the problem. In §3, we define *efficient* reductions, which will allow a cleaner presentation. §4–7 prove the main theorems, organized according to problem type: §4 covers constraint satisfaction problems, §5 covers quantified Boolean formulas, §6 covers optimization problems where the objective function is the solution size, and §7 covers optimization problems where the objective function is the number of satisfied constraints. §8 concludes.

2 Previous Work

[3] show that $dc_{k(r)\text{-SAT}} \leq dc_{\text{DU-}k\text{-SAT}}$ where $k(r)$-SAT is k-SAT but where the input is restricted to have $\leq r$ false clauses at one or more of the assignments $1^n, 0^n$. Since there is nothing special about these 2 assignments, we might as well think of this variant as requiring that the input include some assignment at which the formula has $\leq r$ false clauses - i.e. that not only does the formula have a small satisfiability gap, but that the input include a witness to such a small gap.

But it's not immediately clear whether $k(r)$-SAT is a very important restriction of SAT. On the one hand, the standard Cook reduction from an arbitrary problem in NP to SAT via computation tableaux generates a k-CNF with a gap of 1, and one can even generate a witness for this gap in polytime. This is because the formula generated essentially encodes, "There is a y such that after computing the predicate $R(x, y)$, the result is true," and only a single clause actually encodes the part that says "the result is true". So $k(1)$-SAT is NP-complete, but the number of variables needed in a reduction to $k(r)$-SAT seems to be large, even when reducing from k-SAT, and so it doesn't seem to be very useful for upperbounding the exponential complexity of k-SAT.

On the other hand, the expected fraction of clauses satisfied in a k-CNF under a random assignment is $1 - 2^{-k}$, and so an assignment satisfying at least that many clauses can be found in polytime with high probability. But Håstad showed, using a proof based on PCPs [4], that with no extra restriction on the input formula, no larger a fraction can be guaranteed, unless P = NP. This leaves unclear just how much smaller is the exponential complexity of $k(r)$-SAT than that of k-SAT. We will show that they are equal.

3 Efficient Reductions

Let us say that a parameterized problem A *efficiently reduces* to parameterized problem B, and write $A \preceq B$, iff $\exists$ a polytime Turing reduction f from A to B so that for each instance x of A of parameter n and oracle call y that $f^B(x)$ makes, the parameter of y is $\leq n + O(1)$. Obviously, $\preceq$ is reflexive and transitive. We will also write $A \simeq B$ iff $A \preceq B$ and $B \preceq A$.

Lemma 1. *If $A \preceq B$ then $dc_A \leq dc_B$ and $c_A \leq c_B$.*

Proof. The first inequality is obvious. For the second, let M_B be a randomized poly$(|x|)2^{cn}$-time Turing machine solving B with error $\leq \frac{1}{16} = p$ (this can be constructed by one with error $\leq \frac{1}{3}$ by taking the majority answer from 21 independent calls). Suppose $f^B(x)$ runs in time $\leq t$. Define M_A as f, but replacing each call to the oracle by $r = 2 \lg t$ calls to M_B and take the majority answer. From the union bound, the probability that M_A errs is $\leq t$ times the probability that a binomial random variable with parameters (r, p) is $\geq \frac{1}{2}r$, and this is $\leq \sum_{i=\lceil \frac{r}{2} \rceil}^{r} \binom{r}{i} p^i (1-p)^{r-i} \leq p^{\frac{r}{2}} 2^r = 2^{-r}$. So M_A solves A, takes time $\leq 2t \lg t 2^{c(n+O(1))}$, and errs with probability $\leq t2^{-r} = \frac{1}{t}$. □

Note that lemma 1 would hold even if we significantly loosened the notion of an efficient reduction by allowing subexponential time, randomness with one-sided error, and oracle calls to problems with parameter as much as $n(1+o(1))$, but we will actually be demonstrating this stronger notion here. Also it should be noted that this form of reduction is more strict than the similar SERF reductions of [5], which allow a linear increase in the complexity parameter since they want the loosest possible notion of reduction under which SUBEXP is closed.

4 Constraint Satisfaction Problems

4.1 *k*-SAT

In this section, we show the following.

Theorem 2

$$k\text{-SAT} \simeq \text{SU-}k\text{-SAT} \simeq \text{DU-}k\text{-SAT}\ .$$

(1) and (2) then follow from lemma 1. Note that all problems discussed in this paper with a parameter k are solvable in polytime for $k < 2$, so we will assume throughout that $k \geq 2$. For each problem P, SU-$P \preceq$ DU-P via the identity map, so we won't bother to state this in the proofs. Also, for each problem P that we consider in this paper, DU-$P \preceq P$ by using self-reducibility to find a solution and then making n more queries to decide whether there is another, so we won't bother to formally state this in the proofs to follow either.

For example, if $P = k$-SAT and the input formula is ϕ, we can set a variable x_i, then ask the oracle whether the formula is still satisfiable to discover a correct setting of x_i, then set x_i correctly and continue similarly with the remaining variables to get a complete solution a. Then for each variable x_i, ask the oracle whether $F|x_i \neq a_i$ is satisfiable.

Proof. (k-SAT $\preceq$ SU-k-SAT) Let A be an oracle for SU-k-SAT. Also, let any predicate on $\leq k$ Boolean variables represent the k-clauses logically equivalent to them; e.g. $(x \rightarrow y = 0)$ will stand for the clause $\{\bar{x}, \bar{y}\}$.

Let $\phi = \{C_1, \ldots, C_m\}$ with variables $x_1, \ldots, x_n$ be our input formula and let z be a new variable. For i going from 1 to m, we will find a solution a to the first i clauses $\phi_i = \{C_1, \ldots, C_i\}$, if there is one. Finding an a that satisfies 0 clauses is trivial. Suppose that we have an a that satisfies ϕ_{i-1}. For each literal l in C_i, we ask A whether $\phi_{i-1} \cup \{(z \rightarrow x_j = a_j) \mid j \in [n]\} \cup \{(\bar{z} \rightarrow l)\}$ (which is satisfiable: set $x = a, z = 1$) is uniquely satisfiable. It answers yes iff there is no solution to ϕ_i with $l = 1$. If each of the $|C_i| \leq k$ queries gives an answer of yes, then ϕ_i, and hence ϕ, is not satisfiable. If the jth query answers no, then we can safely set the first $j-1$ literals of C_i to 0 and the jth literal to 1 in a partial assignment b. At this point, we know that b can be extended to a solution to ϕ_i, and we want to use similar calls to the oracle to find assignments to the remaining variables to get a new assignment that satisfies ϕ_i.

More specifically, suppose we have a partial assignment b' that we know can be extended to a solution to ϕ_i. Let b be a partial assignment that extends b' by setting a new variable x_r to an arbitrary value b_r. Then we use A and the following lemma to discover whether b can also be extended to a solution of ϕ_i, and if not, we simply flip b_r. We continue in this way until we have a full solution to ϕ_i.

Lemma 3. *Let $a \in \text{sol}(\phi_{i-1})$, b be a partial assignment, and $\psi = \phi_{i-1} \cup \{(z \rightarrow x_r = a_r) \mid r \in [n]\} \cup \{(\bar{z} \rightarrow x_r = b_r) \mid r \in$ domain of $b\}$. Then ψ has a solution, and it is unique iff b cannot be extended to a solution to ϕ_{i-1}.*

Proof. a together with assigning z to 1 is a solution to ψ. There is no other solution to ψ with $z = 1$, and $\psi\ |_{z=0}$ forces the partial assignment b. □

The reduction uses poly(n) time and makes $\leq mn$ oracle calls, each with $\leq n+1$ variables, so it is efficient. □

4.2 Integer Programming

Let k-BIP be the problem of deciding whether there is a solution to a given system of m linear inequalities in n Boolean variables, where each inequality involves $\leq k$ variables.

Corollary 4

$$k\text{-BIP} \simeq \text{SU-}k\text{-BIP} \simeq \text{DU-}k\text{-BIP} .$$

Proof Since the construction is almost exactly the same as for k-SAT in theorem 2, we only give an outline. To express $\alpha \to \beta$ in the construction, use the inequality $\alpha \leq \beta$. To express a negated variable $\bar{x}$, use $1 - x$. When adding a new constraint C_i, ask the oracle $\leq 2^k$ questions, one for each setting c of the $\leq k$ variables in C_i that satisfies C_i: is it possible to extend $b \cup c$ to a solution of the first $i-1$ constraints? This gives a polytime reduction that makes $\leq m(2^k - k + n)$ queries, each with $\leq n+1$ variables. $\square$

4.3 Backbone Variables

Backbone variables are a tool from statistical physics for understanding the nature of random k-SAT instances - see e.g. [8]. Dubois and Dequen [2] use such variables in a heuristic to refute large unsatisfiable hard random 3-SAT instances.

Given a nonempty set system $S \subseteq \mathcal{P}(U)$, $i \in U$ is a *backbone variable* iff $\forall a \in S\ i \in a \vee \forall a \in S\ i \notin a$. x_i is a backbone variable of formula ϕ iff x_i is a backbone variable of $\text{sol}(\phi)$. Define the k *backbone variable* promise problem (k-BB) to be to decide whether a given variable is backbone in a given k-CNF ϕ with the promise that ϕ is satisfiable.

Corollary 5

$$k\text{-SAT} \simeq k\text{-BB} .$$

Proof Immediate from theorem 2 since a satisfiable k-CNF has exactly 1 solution iff each of its variables is backbone. $\square$

5 Extending the Result Up the PH

Define (d,k)-TQBF to be those true quantified Boolean formulas of the form $Q_1\overrightarrow{w}_1 \cdots Q_d\overrightarrow{w}_d\ \phi$ where each $\overrightarrow{w}_i$ is a (possibly empty) tuple of Boolean variables, each quantifier $Q_i \in \{\exists, \forall\}$ is $\exists$ iff i is odd, each variable of $\phi \in k$-CNF is quantified, and the whole formula is true in the standard sense. The solutions are those assignments to $\overrightarrow{w}_1$ that make $Q_2\overrightarrow{w}_2 \cdots Q_d\overrightarrow{w}_d\ \phi$ true. Then DU-(d,k)-TQBF is essentially those (d,k)-TQBFs where the first quantifier is $\exists!$ ("there is a unique") instead of $\exists$. If we parameterize on the total number of variables n, then we have the following.

Theorem 6

$$(d,k)\text{-TQBF} \simeq \text{SU-}(d,k)\text{-TQBF} \simeq \text{DU-}(d,k)\text{-TQBF} \ .$$

Proof. ((d,k)-TQBF $\preceq$ SU-(d,k)-TQBF) It was shown in [9] that TQBF restricted to quantified 2-CNF formulas is in linear time, so we may assume wlog that $k \geq 3$. Let A be an oracle for SU-(d,k)-TQBF and $F = Q_1\overrightarrow{w}_1 \cdots Q_d\overrightarrow{w}_d\ \phi$ with variables $x_1, \ldots, x_n$ and clauses $\phi = \{C_1, \ldots, C_m\}$ be our input formula. Let $x_1, \ldots, x_q$ be the variables in $\overrightarrow{w}_1$, and y, z be variables not in F. If some clause has only $\forall$ quantified literals, then player $\forall$ can easily win and we reject. Otherwise, for i going from 1 to m, we will find a solution a, if there is one, to $F_i = Q_1\overrightarrow{w}_1 \cdots Q_d\overrightarrow{w}_d\phi_i$ where $\phi_i = \{C_1, \ldots, C_i\}$. Finding an a that satisfies F_0 is trivial. Suppose we have an a that satisfies F_{i-1}. We use A and the following lemma (with $b = \emptyset$) to decide whether F_i is satisfiable.

Lemma 7. *Let $a \in \text{sol}(F_{i-1})$, b be a partial assignment to $\overrightarrow{w}_1$. Suppose $l_k \in C_i$ is $\exists$ quantified under Q_j. Let $\psi = \phi_{i-1} \cup \{(z \to x_r = a_r) \mid r \in [q]\} \cup \{(\bar{z} \to x_r = b_r) \mid r \in \text{domain of } b\} \cup \{(z \to y), \{l_1, \ldots, l_{k-1}, y\}, \{\bar{y}, l_k, z\}\}$, and $G = Q_1 z, \overrightarrow{w}_1 \cdots Q_j y, \overrightarrow{w}_j \cdots Q_d\overrightarrow{w}_d\ \psi$. Then G has a solution, and it is unique iff b cannot be extended to a solution to F_i.*

Proof. a together with assigning z (and y, if $j = 1$) to 1 is a solution to G. There is no other solution to G with $z = 1$, and $G\mid_{z=0}$ forces the partial assignment b. A winning strategy for player $\exists$ for $G\mid_{z=0}$ is also a winning strategy for F_i: just ignore y. Conversely, a winning strategy for F_i is a winning strategy for $G\mid_{z=0}$ if, in addition, we set $y = l_k$, which is possible since they are both quantified under Q_j. □

If F_i is satisfiable, we find a solution b to F_i by starting with the empty partial assignment. Suppose we have a partial assignment b' that we know can be extended to a solution to F_i. Let b be a partial assignment that extends b' by setting a new variable x_r in $\overrightarrow{w}_1$ to an arbitrary value b_r. We use A and lemma 7 to discover whether b can be extended to a solution to F_i, and if not, we flip b_r. We continue in this way until we have a full solution to F_i. The reduction uses poly(n) time and makes $\leq m(q+1)$ oracle calls, each with $\leq n+2$ variables, so it is efficient. □

The same result also holds for (∞, k)-TQBF, i.e. without restricting the number of levels of alternation, but it is less exciting since it is obvious: just add 2 empty quantifiers at the beginning.

One could also ask whether the exponential complexity changes when !s are placed on some subset of existential quantifiers other than just the first one. But the above proof technique does not seem to generalize. One problem that arises when trying to prove that the exponential complexity of the unique case (where the ! is on a $\exists$ other than the first) is no more than that of the non-unique case is that, while before it was easy to store a solution (the a variable) in a polynomial amount of space, given a formula such as $\exists x\ \forall y\ \exists z\ \phi$, it now seems like we have to store a whole function that maps each y to an appropriate z. It is not obvious how to store such a function in a subexponential amount of space. But another problem arises when trying to prove even the reverse inequality: it was easy to show that a problem of the form $\exists! x\ \phi$ Turing reduces to problems of the form $\exists x\ \phi$ with only $o(n)$ more variables and in a subexponential amount of time, but how can one even Turing reduce a problem of the form $\forall x\ \exists! y\ \phi$ to problems of the form $\forall x\ \exists y\ \phi$ with only $o(n)$ more variables and in a subexponential amount of time, let alone other, more complex formulas with more quantifiers?

6 Solution Optimization Problems

The following problems involve optimizing the size of the solution.

6.1 *k*-Hitting Set

Define the *k-Hitting Set* problem (k-HS) as given a k-set system and an integer l, decide whether there is a hitting set of size $\leq l$; i.e. given (U, S, l) such that $S \subseteq \mathcal{P}(U)$ and $\forall s \in S\ |s| \leq k$, if we define the hitting sets as $\{h \subseteq U \mid \forall s \in S\ h \cap s \neq \emptyset\}$, decide whether there is a hitting set of size $\leq l$.

Theorem 8

$$k\text{-HS} \simeq \text{SU-}k\text{-HS} \simeq \text{DU-}k\text{-HS} .$$

In particular, taking $k = 2$ gives the result for vertex cover:

$$\text{VC} \simeq \text{SU-VC} \simeq \text{DU-VC} .$$

Proof. (k-HS $\preceq$ SU-k-HS) Let A be an oracle for SU-k-HS and let (U, S, l) with nodes $U = \{x_1, \ldots, x_n\}$ and sets $S = \{C_1, \ldots, C_m\}$ be our input instance. Let $z, \bar{z}$ be 2 nodes not in U. For i going from 1 to m, we will find a smallest hitting set a of $S_i = \{C_1, \ldots, C_i\}$. Initially, $a = \emptyset$ is a smallest hitting set of S_0. Suppose that we have a smallest hitting set a of S_{i-1}.

Let $L = (l_1, \ldots, l_n)$ be an ordering of U with the nodes of C_i first. Let sets b, c be initially empty. For j going from 1 to n, we will maintain the invariant that at step j, if S_i has a hitting set of size $\leq |a|$, then the lexicographically largest such (according to the order L, where l_1 is the most significant) contains all of b and none of c, and $b \cup c = \{l_1, \ldots, l_j\}$. Suppose the invariant holds for $j - 1$. We use A and the following lemma (with $b' = b \cup \{l_j\}$) to discover whether $b \cup \{l_j\}$ is contained in a hitting set of S_i of size $\leq |a|$.

Lemma 9. *Let a be a smallest hitting set of S_{i-1}, $b' \subseteq U$, and $T = S_{i-1} \cup \{\{z, \bar{z}\}\} \cup \{\{\bar{z}, x\} \mid x \in a\} \cup \{\{z, x\} \mid x \in b'\}$. Then T has a hitting set of size $\leq |a| + 1$, and it is unique iff S_i does not have a hitting set of size $\leq |a|$ that contains b'.*

Proof. $a \cup \{z\}$ is a hitting set of T of size $\leq |a| + 1$. There is no other containing z, and any without z contains b'. □

If the answer is yes (which corresponds to an oracle answer of no), we add l_j to b, otherwise to c. If c ever contains all of C_i then S_i has no hitting set of size $\leq |a|$, in which case, letting l be an arbitrary element of C_i, $a \cup \{l\}$ is a smallest hitting set of S_i. Otherwise, we continue applying the lemma, adding elements to either b or c, until we have a hitting set of S_i of size $\leq |a|$. Once we have a smallest hitting set a for S_m, we simply compare $|a|$ to l. The reduction uses poly(n) time and makes $\leq mn$ oracle calls, each with $\leq n+2$ nodes, so it is efficient. $\square$

6.2 k-Hypergraph Independent Set

The *k-Hypergraph Independent Set* problem (k-HIS) is, given a k-hypergraph (i.e. where each edge contains $\leq k$ vertices) and an integer l, decide whether there is a set I of vertices of size $\geq l$ such that no edge is contained in I.

Corollary 10

$$k\text{-HIS} \simeq \text{SU-}k\text{-HIS} \simeq \text{DU-}k\text{-HIS} .$$

In particular, taking $k = 2$ gives the result for independent set:

$$\text{IS} \simeq \text{SU-IS} \simeq \text{DU-IS} .$$

Proof. Follows immediately from theorem 8 by observing that a set of vertices is independent and of size $\geq l$ iff its complement is a hitting set of the edges and of size $\leq n - l$.

6.3 Limitations

Although Hitting-Set (HS) and Set-Cover (SC) are duals of one another, it is not as obvious how the exponential complexities of k-HS and k-SC are related. The technique used to show theorem 8 does not seem to work for SC since constraints are represented by the universe elements and not the sets, so adding linearly many constraints to construct the oracle queries increases the parameter from n to cn for some $c > 1$, causing the reduction to be inefficient.

The situation for k-Coloring is similar.[1] Though we can easily construct an oracle query with the right logical properties, using only the techniques here will cause it to have linearly many more vertices, and thus only show that $c_{k\text{-Coloring}} \leq O(c_{\text{DU-}k\text{-Coloring}})$, where the constant factor in the big-Oh does not depend on k.

7 Constraint Optimization Problems

The following problems involve optimizing the number of constraints that are satisfied.

Max-k-SAT (Min-k-SAT) is the problem of deciding whether a given k-CNF with an integer weight for each clause has an assignment that satisfies at least (at most) some given weight l.

[1] Here we take 'uniqueness' of a coloring solution to mean 'unique up to permutations of the colors'.

Theorem 11

$$\text{Max-}k\text{-SAT} \simeq \text{SU-Max-}k\text{-SAT} \simeq \text{DU-Max-}k\text{-SAT}\ .$$

Proof. (Max-k-SAT $\preceq$ SU-Max-k-SAT) Let A be an oracle for SU-Max-k-SAT and $\phi \in k$-CNF, $l \in \mathbb{Z}$ be our input where ϕ has variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ with (wlog) nonzero weights $w_1, \ldots, w_m$. Let z be a variable not in ϕ. For i going from 1 to m, we will find the maximum weight l' that can be satisfied in $\phi_i = \{C_1, \ldots, C_m\}$ and an assignment a that satisfies that weight. Initially, $l' = 0$ and a is arbitrary. Suppose we know the maximum weight l' that can be satisfied in ϕ_{i-1} and a satisfies it. We want to find the maximum weight l'' that can be satisfied in ϕ_i. We consider 2 cases: $w_i > 0$ and $w_i < 0$.

If $w_i > 0$, then $l'' \in [l', l' + w_i]$. For each literal $l \in C_i$ and $t' \in (l', l' + w_i]$, we can use A and the following lemma (with b the partial assignment that sets $l = 1$, and $t = t' - w_i$) to discover whether some assignment with $l = 1$ satisfies weight $\geq t'$ in ϕ_i.

Lemma 12. *Let l' be the largest weight that can be satisfied in ϕ_{i-1} and let a satisfy that weight. Let $|w_i| > 0$ and $l' - |w_i| < t \leq l'$. Let b be a partial assignment and $\psi = \phi_{i-1} \cup \{|w_i| \cdot (z \to x_r = a_r) \mid r \in [n]\} \cup \{|w_i| \cdot (\bar{z} \to x_r = b_r) \mid r \in \text{domain of } b\}$. Then ψ has an assignment that satisfies weight $\geq t + (n + |b|)|w_i|$, and it is unique iff b cannot be extended to an assignment that satisfies weight $\geq t$ in ϕ_{i-1}.*

Proof. a together with $z = 1$ satisfies weight $\geq l' + (n + |b|)|w_i| \geq t + (n + |b|)|w_i|$ in ψ. There is no other such assignment with $z = 1$, and any with $z = 0$ agrees with b. □

So we can use binary search to discover l'' using $\leq |C_i| \lg(w_i + 1)$ queries. If $l'' = l'$, then a satisfies weight l'' in ϕ_i and we are done. Otherwise, as a slight optimization, if the last query set the jth literal of C_i to 1, then we can safely set literal j to 1 and literals 1 through $j - 1$ to 0 in the partial assignment b, and at this point, we know that b can be extended to a full assignment satisfying weight l'' in ϕ_i *and* that b satisfies C_i (since $l'' > l'$). We can continue using the lemma to extend b to such an assignment using $n - j$ more queries to A.

If $w_i < 0$, then $l'' \in [l' + w_i, l']$. Let b be the partial assignment that makes C_i false. For each $t \in (l' + w_i, l']$ we can use 1 query to A and the lemma to discover whether there is an assignment satisfying weight $\geq t$ in ϕ_i. So we can use binary search to discover l'' using $\leq \lg(|w_i| + 1)$ queries. If $l'' = l' + w_i$, then a satisfies weight l''. Otherwise, we can continue using the lemma to extend b to a full assignment satisfying weight l'' in ϕ_i using $n - |C_i|$ more queries.

At the end, we simply compare l' to l. The reduction uses time polynomial in the size of the input and makes $\leq \sum_i (|C_i| \lg(|w_i| + 1) + n - |C_i|)$ oracle calls, each with $\leq n + 1$ variables, so it is efficient. □

Theorem 11 applies to integer weights, but the proof is robust and can easily be modified to accommodate rational weights, e.g. by first multiplying by the

LCM of the denominators. Since the weights used in the proof have the same size as those of the input, the same proof works for the problem restricted to unit weights. The same proof also works for nonnegative weights, or any combination of the above.

Negating the input weights together with the algorithm in the proof also gives an efficient reduction from Min-k-SAT to SU-Min-k-SAT. The only rub is that this reduction is not correct if we restrict to nonnegative weights. To handle this case, we use a different reduction below.

Theorem 13. *If we restrict to nonnegative integer weights,*

$$\text{Min-}k\text{-SAT} \simeq \text{SU-Min-}k\text{-SAT} \simeq \text{DU-Min-}k\text{-SAT} .$$

Two clauses are in conflict with each other, i.e. cannot be simultaneously unsatisfied, iff they share a variable, but positively in the one and negatively in the other. We want to find a maximum weight independent set in the graph with the clauses as vertices and edges between conflicting clauses. But even if we restricted to unit weights, we cannot simply invoke corollary 10 since the parameter here is variables, not clauses. Thus it seems like we need another proof.

It should be pointed out that the proof below does not work if we restrict to unit weights since the construction of ψ uses non-unit weights. There may be some more convoluted construction using the same ideas but that avoids this technical point.

Proof. (Min-k-SAT $\preceq$ SU-Min-k-SAT) Let A be an oracle for SU-Min-k-SAT and $\phi \in k\text{-CNF}, l \in \mathbb{Z}$ be our input where ϕ has variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$ with (wlog) positive weights $w_1, \ldots, w_m$. Let z be a variable not in ϕ. For i going from 1 to m, we will find the minimum weight l' that can be satisfied in $\phi_i = \{C_1, \ldots, C_i\}$ and an assignment a that satisfies that weight. Initially, $l' = 0$ and a is arbitrary. Suppose we know the minimum weight l' that can be satisfied in ϕ_{i-1} and a satisfies it. The minimum weight l'' that can be satisfied in ϕ_i is in the interval $[l', l' + w_i]$. Letting b be the partial assignment that makes C_i false, we can use the following lemma and binary search to discover l'' using $\lg(w_i + 1)$ calls to A.

Lemma 14. *Let l' be the smallest weight that can be satisfied in ϕ_{i-1} and let a satisfy that weight. Let $w_i > 0$ and $l' \le t < l' + w_i$. Let b be a partial assignment and $\psi = \phi_{i-1} \cup \{w_i \cdot (z \to x_r \neq a_r) \mid r \in [n]\} \cup \{w_i \cdot (\bar{z} \to x_r \neq b_r) \mid r \in$ domain of $b\} \cup \{(n - |b|) w_i \cdot (z)\}$. Then ψ has an assignment that satisfies weight $\le t + nw_i$, and it is unique iff b cannot be extended to an assignment that satisfies weight $\le t$ in ϕ_{i-1}.*

Proof. a together with $z = 1$ satisfies weight $\le l' + nw_i \le t + nw_i$ in ψ. There is no other such assignment with $z = 1$, and any with $z = 0$ agrees with b. □

If $l'' = l' + w_i$, then a satisfies weight l'' in ϕ_i. Otherwise, we use the lemma (with $t = l''$) $n - |C_i|$ times to extend b to a full assignment achieving weight l''

in ϕ_i. At the end, we simply compare l' to l. The reduction uses time polynomial in the size of the input and makes $\leq \sum_i (\lg(w_i + 1) + n - |C_i|)$ oracle calls, each with $\leq n + 1$ variables, so it is efficient. □

8 Conclusions

We show a simple technique to settle conjecture (1) as well as many questions relating the exponential complexity of similar parameterized constraint satisfaction problems to their unique-solution counterparts. Our problem list here is not intended to be exhaustive but demonstrative. Theorem 11 shows the robustness of the technique, allowing considerable variation in problem specification without disturbing the proof.

Relating the exponential complexities of such problems to their unique-solution counterparts under the *promise* of at most 1 solution appears to be harder. Current techniques [10,1] for that problem use oblivious hashing (i.e. not looking at the input but only its size), but fall short of such strong results as are here. It seems like new, non-oblivious techniques are needed.

References

1. Calabro, C., Impagliazzo, R., Kabanets, V., Paturi, R.: The complexity of Unique k-SAT: An isolation lemma for k-CNFs. Journal of Computer and System Sciences 74(3), 386–393 (2008)
2. Dubois, O., Dequen, G.: A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence, pp. 248–253 (2001)
3. Grandjean, E., Büning, H.: SAT-problems and reductions with respect to the number of variables. Journal of Logic and Computation 7(4), 457–471 (1997)
4. Håstad, J.: Some optimal inapproximability results. In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pp. 1–10. ACM Press, New York (1997)
5. Impagliazzo, R., Paturi, R., Zane, F.: Which Problems Have Strongly Exponential Complexity? Journal of Computer and System Sciences 63, 512–530 (2001)
6. Impagliazzo, R., Paturi, M.: On the complexity of k-SAT. Journal of Systems Sciences 62(2), 367–375 (2001)
7. Iwama, K., Tamaki, S.: Improved upper bounds for 3-SAT. ECCC Report TR03-053 (2003)
8. Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., Troyansky, L.: Determining computational complexity from characteristic phase transitions. Nature 400(8), 133–137 (1999)
9. Plass, M., Aspval, B., Tarjan, R.: A linear time algorithm for testing the truth of certain quantified Boolean formulas. Information Processing Letters 8, 121–123 (1979)
10. Valiant, L., Vazirani, V.: NP is as easy as detecting unique solutions. Theoretical Computer Science 47, 85–93 (1986)

Unique Decipherability in the Monoid of Languages: An Application of Rational Relations[⋆]

Christian Choffrut[1] and Juhani Karhumäki[2]

[1] L.I.A.F.A., Université Paris 7, 2 Pl. Jussieu – 75 251 Paris Cedex – France
[2] Dept. of Math. and TUCS, University of Turku, 20014, Turku – Finland

Abstract. We attack the problem of deciding whether a finite collection of finite languages is a code, that is, possesses the unique decipherability property in the monoid of finite languages. We investigate a few subcases where the theory of rational relations can be employed to solve the problem. The case of unary languages is one of them and as a consequence, we show how to decide for two given finite subsets of nonnegative integers, whether they are the n-th root of a common set, for some $n \geq 1$. We also show that it is decidable whether a finite collection of finite languages is a Parikh code, in the sense that whenever two products of these sets are commutatively equivalent, so are the sequences defining these products. Finally, we consider a nonunary special case where all finite sets consist of words containing exactly one occurrence of the specific letter.

Keywords: Unique decipherability, finite automata, regular languages.

1 Introduction

The question whether or not a given morphism $h : \Sigma^* \to \Delta^*$ is injective, that is, whether or not the encoded message can be uniquely decoded, is fundamental in the theory of message transmission. More precisely, the problem asks whether or not the given set of code words possesses the *unique decipherability property*. The issue for finite sets X was already affirmatively answered in 1950 with the so-called Sardinas and Patterson algorithm, see [19]. Later, it was extended, via syntactic monoids, to all rational sets, see [3] and its complexity was analyzed in [7].

A particularly illustrative way of solving this problem is to construct a finite two-tape automaton for all double representations (factorizations) of message sequences and to reduce the testing to the emptiness problem for rational relations. In [5] the same approach was used to show how to decide whether or not, given two finite sets X and Y, the two monoids X^* and Y^* they generate

[⋆] This research was done during the second author's visit at the LIAFA, CNRS UMR 7089, Université Paris Diderot, Novembre–December 2008. The second author was supported by the Academy of Finland under the grant 126211.

are isomorphic. This, however, works for finite sets only and the general case of rational sets is still open.

The unique decipherability problem can be formulated for any associative algebra. It has been studied, e.g., in the theory of trees, [15], or in the case of multivalued encodings, [18]. In the case of the monoid of finite languages, amazingly, little seems to be known, a splendid exception being the fact that the set of prefix languages under the operation of concatenation product is free, that is, any collection of finite prefix sets is a code, see [17]. A partial explanation to the lack of such results was revealed recently when it was shown in a number of papers, how powerful or difficult language equations are. To mention a few examples, it is shown in [12] that even the question whether or not, for given finite sets A, B, C, D, E, F, the equation $AB^iC = DE^iF$ holds for all $i \geq 0$, is recursively undecidable, see also [14]; the maximal set commuting with a given finite set A need not be recursive, see [13]; or the fact that two given finite sets A and B are conjugate, i.e., that there exists a set Z such that $AZ = ZB$ holds, is known to be decidable only in the case of bifix sets, see [4].

Nonetheless, there are two related research topics which have been studied in the literature. Research on decomposition of rational languages was initiated already in Conway's book, see [8]. Later, this research was pursued in, e.g., [11], where also prime decompositions are defined. In another direction, unambiguous products of languages were studied in [1] and [16]. As we shall see, it is this, or more precisely its negation, the ambiguity, which makes our problems difficult.

As already hinted, our goal is to tackle the unique decipherability decision problem for a finite collection of finite languages. More precisely, we want to apply the theory of rational relations to solve a few special cases, even if the general problem seems to be very much beyond the reach of our tools.

The structure of our presentation is as follows. In section 2, we fix the terminology, recall the basic tools we are using and prove a simple case of our problem to be decidable, namely we show how to decide, given two finite sets of integers, whether or not some of their nonnegative powers (actually subset sums since we are working in the additive structure) coincide. The problem can be formulated as a natural decision question in additive number theory: decide whether or not two finite sets of numbers are the (additive) roots of some common set.

In section 3 we extend the above proof to the unique decipherability property for unary languages, and later in section 4 a further extension is introduced where so-called Parikh codes are considered. We say that a collection of finite languages is a Parikh code, in the sense that whenever two products of these sets are commutatively equivalent, so are the two sequences defining these products. Being a Parikh code is necessary but not sufficient for a set of finite languages to be a code. Finally, in section 5, another approach of using rational languages is introduced. It allows us to consider the case when all words of all sets of the collection contain one and only one occurrence of a fixed letter. In this case, we can decide, given two such sets, whether or not some of their powers coincide; we also outline methods of deciding whether a given collection of special finite languages is uniquely decipherable, that is a code.

2 Preliminaries and an Example

In this section we fix the terminology, recall basic results and give a simple example. For a general reference to the field, we suggest [2] and [3].

We denote by Σ a finite alphabet, and by Σ^* the free monoid it generates. Elements and subsets of Σ^* are called *words* and *languages*, respectively. Other monoids considered here are submonoids of the additive monoid of nonnegative integers $\mathbb{N}$, and Cartesian products of these and of Σ^*. Our main concern is on finite languages and finite subsets of $\mathbb{N}$ and $\mathbb{N}^k$, and basic tools to deal with those rely on properties of rational, i.e., semilinear sets. We recall that a *rational* subset of a monoid is a subset obtained from finite subsets by applying finitely many times the operations of set union, product and Kleene iteration, also known as the star-operation. The result of applying the Kleene operation to the subset X is denoted by X^*. We shall also use the notation $X^+ = XX^* = X^*X$. A *linear* set, in turn, is a set of the form

$$\{a + \lambda_1 b_1 + \cdots \lambda_p b_p \mid \lambda_i \in \mathbb{N},\ \text{for } i = 1, \ldots, p\}$$

where $p \geq 0$ and $a, b_i \in \mathbb{N}^k$ for $1 \leq i \leq p$. A *semilinear* set, is a finite union of linear sets.

We recall that a subset of Σ^* (resp. the Cartesian product $\Sigma^* \times \Delta^*$) is rational if and only if it is recognized by some finite one-tape (resp. two-tape) automaton.

It is quite straightforward to check that the family of rational sets of $\mathbb{N}^k$ is identical to the family of semilinear sets. A fundamental, nontrivial property due to Ginsburg and Spanier is that this family is closed under complement. More precisely, we have, see [10], also [9].

Theorem 1. *The family of semilinear sets is an effective Boolean algebra.*

This means that not only the family is closed under the Boolean operations, but that from a specification of two semilinear sets we can compute a specification for the complement and the intersection. This theorem plays a crucial role in our considerations, as well as some other closure properties of semilinear sets such as the closure under morphic images and projections.

Now we state our basic problems. Let M be an associative algebra, that is an algebra with a single associative operation. A subset $X \subseteq M$ is *uniquely decipherable* in M if , whenever

$$x_1 \cdots x_p = y_1 \cdots y_q, \quad \text{with } x_i, y_j \in X$$

holds, then necessarily we have

$$p = q \ \text{ and } x_i = y_i, \quad \text{for } i = 1, \ldots, p\ .$$

This leads to the following decision issue: the UNIQUE DECIPHERABILITY PROBLEM for M, (UD-problem for short) asks to decide whether or not a given finite subset of M can be uniquely deciphered.

Two related simpler problems are:

The Power equality problem for M (PE for short) asks whether or not, for two given finite subsets X and Y of M, some of their powers coincide, that is whether or not $X^n = Y^m$ holds for some $n, m \geq 1$.
The Common root problem for M (CR for short) asks whether or not for two given finite subsets X and Y of M, they are distinct n-th powers of a set for a certain n, that is, whether or not $X^n = Y^n$ holds for some $n \geq 1$.

We conclude this section by illustrating our techniques with a simple example, which, we believe, is a natural problem in additive theory of numbers. By convention, we keep using the multiplicative notation though we work with the additive structure of the integers. In particular if X and Y are two subsets of integers, then XY stands for all the sums of the form $x + y$ with $x \in X$ and $y \in Y$ and the notation X^n stands for the expression $\overbrace{X + \cdots + X}^{n \text{ times}}$. E.g., with $X = \{0, 1, 2\}$ we have $X^2 = X + X = \{0, 1, 2, 3, 4\}$.

Theorem 2. *Given two finite subsets $X, Y \subseteq \mathbb{N}$, it is recursively decidable whether or not the equality $X^n = Y^m$ holds for some integers $n, m \geq 1$.*

Proof. We define a subset of $\mathbb{N}^3$

$$Z = \mathbb{N} \times (X \times 1)^+ \setminus (1 \times Y)^+ \times \mathbb{N}\,. \tag{1}$$

If $\pi_{1,3}$ is the projection of $\mathbb{N}^3$ onto $\mathbb{N}^2$ defined by $\pi_{1,3}(x, y, z) = (x, z)$, then we claim that the following holds

$$(n, m) \in \pi_{1,3}(Z) \text{ if and only if } X^n \not\subseteq Y^m\,.$$

Indeed, this follows from the construction: if $(n, m) \in \pi_{1,3}(Z)$ then there exist elements $x_1, \ldots, x_n \notin Y^m$, that is $X^n \not\subseteq Y^m$, and conversely. Consequently,

$$(n, m) \notin \pi_{1,3}(Z) \text{ if and only if } X^n \subseteq Y^m\,. \tag{2}$$

It follows that the set of pairs (n, m) satisfying the condition (2), which defines a rational relation on $\mathbb{N}^2$, characterizes the pairs of integers for which $X^n \subseteq Y^m$ holds. Similarly, the relation characterizing the set of pairs for which $X^n \supseteq Y^m$ holds, is rational. As the intersection is again rational and effective, the equality $X^n = Y^m$ holds for some integers n and m if and only if this rational relation is nonempty. □

Theorem 2, rather its proof, has the following immediate consequences.

Corollary 1. *The common root problem of finite subsets for $\mathbb{N}$ is decidable.*

Corollary 2. *It is recursively decidable whether or not two finite subsets X and Y of $\mathbb{N}$ are ultimately equivalent, i.e., whether or not there exists an integer N such that*

$$X^n = Y^n$$

holds for $n \geq N$.

Of course, in either of the above corollaries, in order to get the answer "yes", the maximal and minimal numbers of the two subsets must necessarily coincide. This condition holds for the second largest and smallest elements as well. For the others, the ambiguity comes into play, and makes the problem difficult to analyse.

A simple example from [6] showing that a square root of a set may not be unique is as follows: take $X = \{0, 2, 3, 7, 10, 12, 14, 15\}$ and $Y = \{0, 2, 3, 7, 12, 13, 14, 15\}$. Then $X^2 = Y^2 = [0, 30] \setminus \{1, 8, 11, 23\}$.

3 The Unary Case

In this section we extend our considerations of the previous section to cover the UD-problem for unary languages, that is we prove

Theorem 3. *The unique decipherability problem is decidable for unary languages*

Proof. Let $\Xi = \{X_1, \ldots, X_k\}$ be a collection of finite unary languages. We have to decide whether or not there exist two sequences $i_1, \ldots, i_p$ and $j_1, \ldots, j_q$ such that

$$X_{i_1} \ldots X_{i_p} = X_{j_1} \ldots X_{j_q} \text{ with } X_{i_\alpha}, X_{j_\beta} \in \Xi \text{ and } i_1, \ldots, i_p \neq j_1, \ldots, j_q$$

We fix some notations. For $i = 1, \ldots, k$ let $\pi_i : \mathbb{N}^k \to \mathbb{N}$ be the projection onto the i-th component. Furthermore, let $e_i \in \mathbb{N}^k$ be the vector having 1 in position i and 0 everywhere else. We modify the expression (1) in the proof of Theorem 2 by setting

$$Z = \mathbb{N}^k \times \Big(\bigcup_{j=1}^{k} (X_j \times e_j) \Big)^+ \setminus \Big(\bigcup_{j=1}^{k} (e_j \times X_j) \Big)^+ \times \mathbb{N}^k$$

Then we have $(z_1, x, z_2) \in Z$ with $z_1, z_2 \in \mathbb{N}^k$ and $x \in \mathbb{N}$ if and only if $x \in X(z_1) \setminus X(z_2)$, where $X(z_\alpha) = \prod_{i=1}^{k} X_i^{\pi_i(z_\alpha)}$, for $\alpha = 1, 2$. The last part of the proof mimics that of Theorem 2, the only additional feature being that at the end we have to intersect with the following rational subset of $\mathbb{N}^{2k}$

$$\{(x, y) \in \mathbb{N}^{2k} \mid x, y \in \mathbb{N}^k, x \neq y\} \qquad \square$$

4 Unique Parikh Decipherability

Our method allows us to go still a step further. In the previous section we were able to solve our problem for all unary languages. Here, we can solve the general problem at the price of substituting the condition of unique Parikh decipherability to that of unique decipherability.

We say that a collection $\Xi = \{X_1, \ldots, X_k\}$ of finite languages possesses the unique Parikh decipherabilty property if the condition

$$X_{i_1} \ldots X_{i_p} \sim_c X_{j_1} \ldots X_{j_q}$$

implies that

$$i_1 \ldots i_p \sim_c j_1 \ldots j_q$$

holds, where $\sim_c$ is used to denote the commutative equivalence of languages or words, respectively. We can formulate

Theorem 4. *The unique Parikh decipherability is recursively decidable for finite collections of finite languages in Σ^*.*

Sketch of the Proof. This result is actually a generalization of Theorem 2. Indeed, it can be shown that this latter theorem holds for subsets of $\mathbb{N}^m$ for arbitrary $m \geq 1$, not only for subsets of $\mathbb{N}$. Now, set $\Sigma = \{a_1, \ldots, a_m\}$ and consider the morphism $\phi : \Sigma^* \to \mathbb{N}^m$ which maps each word $w \in \Sigma^*$ to the m-tuple $(|w|_{a_1}, \ldots, |w|_{a_m})$ where $|w|_{a_i}$ denotes the number of occurrences of the letter a_i in w. Then our problem reduces to the unique decipherability problem for the finite collection $\phi(X_1), \ldots, \phi(X_k) \subseteq \mathbb{N}^m$. □

It is worthwhile emphasizing that all our results reported so far are based on strong closure properties of rational relations in the commutative case, in particular the closure under complement, and as a consequence under intersection. This leads to the following comments. First, the complexity of our algorithms are quite high, particularly due the the operation of complementation. Second, there is no hope to extend our approach at least in a naive way, since rational relations over free monoids with more than one generator are not closed under intersection. On the other hand, we do not see how to construct complicated examples of collections of finite sets which would satisfy the unique Parikh decipherability, but would not satisfy the unique decipherability property.

Two last observations. It can be readily shown as hinted in the proof of the previous theorem, that Theorem 2 carries over from $\mathbb{N}$ to $\mathbb{N}^m$ for $m \geq 1$ and actually also to $\mathbb{Z}^m$. Also, since the commutative image of a rational subset of a free monoid is a semilinear set of $\mathbb{N}^m$, Theorem 4 also holds for a finite collections of rational, not only finite, languages of a free monoid.

5 A Special Nonunary Case

In this section we consider the UD-problem, for languages over a general alphabet, but in quite a restricted setting, namely we assume that the finite languages are subsets of

$$(\Sigma \setminus \{b\})^* b (\Sigma \setminus \{b\})^* \text{ for some fixed letter } b \in \Sigma. \quad (3)$$

We show that the problem is decidable in this case. The solution is based on strong closure properties of rational languages.

Theorem 5. *The unique decipherability problem is decidable for any collection of finite sets of the form (3).*

Sketch of the Proof. We outline the main idea of the proof in a simple setting without going into technical details. The proof is based on strong closure properties of rational language.

We start by solving a different problem. Given two finite collections of finite languages of type (3)

$$\begin{aligned} &X_i \ \text{ for } i = 1, \ldots, N \text{ and,} \\ &Y_i \ \text{ for } i = 1, \ldots, N, \end{aligned}$$

where

$$\begin{aligned} X_i &= \{x_{i,s} \mid s = 1, \ldots, s(i)\} \text{ for } i = 1, \ldots, N \text{ and} \\ Y_i &= \{y_{i,r} \mid r = 1, \ldots, r(i)\} \end{aligned}$$

determine whether or not

$$X_w = Y_w \tag{4}$$

holds for some $w = i_1 \cdots i_k \in \{1, \ldots, N\}^+$. The notation X_w stands for the product

$$X_w = X_{i_1} \cdots X_{i_k}.$$

The idea is to determine the set of $w \in \{1, \ldots, N\}^+$ such that

$$X_w \subseteq Y_w$$

and to test whether of not its intersection with the set of w's such that $Y_w \subseteq X_w$ holds, is not empty. We show that these two languages are recognized by finite automata, therefore that the test is effective.

It suffices to prove the claim for the set of words for which the inclusion $X_w \subseteq Y_w$ holds. We define an automaton $\mathcal{A}$ as follows. Its states are words over $\Sigma \setminus \{b\}$ and inverses (considering the free monoid as embedded in the free group), the empty word 1 being both the initial and final state. Its alphabet is the set

$$J = \{(i, s) \mid i = 1, \ldots, N \text{ and } s = 1, \ldots, s(i)\}$$

The transitions are defined as follows: For two states α and β and $x_{i,s} \in X_i$ if

$$\alpha x_{i,s} = y_{i,r}\beta, \ \text{ for some } y_{i,r} \in Y_i,$$

then there is a transition

$$\alpha \xrightarrow{(i,s)} \beta \tag{5}$$

Actually, here are four different possibilities depending on whether α and/or β are words, as above, or there formal inverses. These modifications are obvious. Clearly, $\mathcal{A}$ is well defined (due to the form of sets X_i and Y_i) finite but nondeterministic automaton. Intuitively, it checks those products of X-words which can be decomposed into Y-words as well. Note that, again according to the form of our words, such decompositions are of the same length.

Set $I = \{1, \ldots, N\}$ and define the letter-to-letter substitution $\pi : J^* \mapsto I^*$ by posing $\pi(i, s) = i$. Let L be the language recognized by the automaton and set $P = \pi^{-1}\pi(L)$. Then the set of w's such that $X_w \subseteq Y_w$ holds is equal to $\pi(P) \setminus (\pi(P \setminus L))$. Since all these operations are effective and involve rational languages, the problem (4) is decidable.

We now turn to the proof of Theorem 5 is obtained by modifying the above construction. Since we want to decide the noncode property, that is that $X_w = X_{w'}$ with $w \neq w'$, we have to generate two different sequences of indices, and then test the inclusions

$$X_w \subseteq X_{w'} \quad \text{and} \quad X_{w'} \subseteq X_w .$$

This all can be done, by defining the starting automaton $\mathcal{A}$ having transitions of the form

$$\alpha \xrightarrow{((i,s),i,i',(i',s'))} \beta$$

instead of (5). The first (resp. the last) two components are used to check the inclusion $X_w \subseteq X_{w'}$ (resp. $X_{w'} \subseteq X_w$), and simultaneously the second and the third component are used to guarantee that $w \neq w'$. We omit the details. □

The first part of the previous proof can be reformulated as

Corollary 3. *For two finite languages $X, Y \subseteq (\Sigma\setminus\{b\})^* b (\Sigma\setminus\{b\})^*$, with $b \in \Sigma$, it is recursively decidable whether or not there exists an n such that $X^n = Y^n$ holds true.*

The above deserves a few comments. It relies very much on the special form of the X-sets, that is on the fact that there is just one "marker" symbol in all words of X sets. On the other hand, the marker need not be a symbol. That is to say that X should satisfy the conditions that each word in X contains exactly one occurence of a word u and each occurence of X^2 contains exactly two occurences of u.

References

1. Anselmo, M., Restivo, A.: Factorizing languages. In: IFIP Congress (1), pp. 445–450 (1994)
2. Berstel, J.: Transductions and context-free languages. B. G. Teubner (1979)
3. Berstel, J., Perrin, D.: The theory of codes, vol. 117. Academic Press, London (1985)
4. Cassaigne, J., Karhumäki, J., Salmela, P.: The conjugacy of biprefix sets (to appear)
5. Choffrut, C., Harju, T., Karhumäki, J.: A note on decidability questions on presentations of word semigroups. Theor. Comput. Sci. 183(1), 83–92 (1997)
6. Choffrut, C., Karhumäki, J.: On Fatou properties of rational languages. In: Martin-Vide, C., Mitrana, V. (eds.) Where Mathematics, Computer Science, Linguistics and Biology Meet, pp. 227–235. Kluwer, Dordrecht (2000)
7. Chrobak, M., Rytter, W.: Unique deciperability for partially commutative alphabet (extended abstract). In: Gruska, J., Rovan, B., Wiedermann, J. (eds.) MFCS 1986. LNCS, vol. 233. Springer, Heidelberg (1986)

8. Conway, J.H.: Regular algebras and finite machines. Chapman and Hall, Boca Raton (1974)
9. Eilenberg, S., Schützenberger, M.-P.: Rational Sets in Commutative Monoids. Journal of Algebra 13, 173–191 (1969)
10. Ginsburg, S., Spanier, E.H.: Semigroups, Presburger formulas, and languages. Pacific Journal of Mathematics 16, 285–296 (1966)
11. Han, Y.-S., Salomaa, A., Salomaa, K., Wood, D., Yu, S.: On the existence of prime decompositions. Theory Comput. Syst. 376, 60–69 (2007)
12. Karhumäki, J., Lisovik, L.P.: The equivalence problem of finite substitutions on ab*c, with applications. Int. J. Found. Comput. Sci. 14(4), 699 (2003)
13. Kunc, M.: The power of commuting with finite sets of words. Theory Comput. Syst. 40(4), 521–551 (2007)
14. Kunc, M.: The simplest language where equivalence of finite substitutions is undecidable. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) FCT 2007. LNCS, vol. 4639, pp. 365–375. Springer, Heidelberg (2007)
15. Mantaci, S., Restivo, A.: Codes and equations on trees. Theor. Comput. Sci. 255(1-2), 483–509 (2001)
16. Massazza, P., Bertoni, A.: On the square root of languages. In: Formal power series and algebraic combinatorics, Moscow, pp. 125–134 (2000)
17. Perrin, D.: Codes conjugués. Information and Control 20(3), 222–231 (1972)
18. Restivo, A.: A note on multiset decipherable codes. IEEE Transactions on Information Theory 35(3), 662–663 (1989)
19. Sardinas, A.A., Patterson, C.W.: A necessary and sufficient condition for the unique decomposition of coded messages. In: IRE Intern. Conv. Rec., vol. 8, pp. 104–108. Chapman and Hall, Boca Raton (1953)

Concurrently Non-malleable Black-Box Zero Knowledge in the Bare Public-Key Model*

Yi Deng[1], Giovanni Di Crescenzo[2], Dongdai Lin[1], and Dengguo Feng[1]

[1] State Key Laboratory of Information Security, Institute of Software, Chinese Academy of sciences, Beijing, 100190, China
{ydeng,ddlin,feng}@is.iscas.ac.cn
[2] Telcordia Technologies, Piscataway, NJ, USA
giovanni@research.telcordia.com

Abstract. In this paper we study the possibility of reducing the setup assumptions under which concurrent non-malleable zero knowledge protocol can be realized. A natural model choice is the bare public-key (BPK) model of [6], a model with very minimal setup assumptions. Our main contribution is to show in this model the following about constant-round concurrent non-malleable *black-box* zero-knowledge arguments.

– They can be constructed from any *one-way function* for any language in $\mathcal{NP}$. Here, our construction takes 5 rounds, and we can reduce it to a 4-round (round-optimal) argument under existence of one-way permutations.
– Under number-theoretic assumptions, they admit a *time-efficient* instantiation for some specific $\mathcal{NP}$ languages (e.g., all languages having efficient Σ protocols, for which we can implement our construction using only $\mathcal{O}(1)$ modular exponentiations).

Compared to the *non-black-box* construction in a concurrent work of [OPV, ICALP 2008] in this model, our protocol (even the construction from one-way function) is significantly more time- and round-efficient and can be based on more general assumptions.

1 Introduction

Zero-knowledge protocols, first introduced in [26], have received a significant amount of attention from the research community because of their useful applications to several cryptographic protocols in a variety of settings. As such protocols are often deployed in distributed and asynchronous networks like the Internet, the research on these protocols is moving towards extending the security properties of (stand-alone) zero-knowledge protocols to models with multiple parties, asynchronous message delivery, and adversarial modification to exchanged messages.

Concurrent Zero-Knowledge. The notion of concurrent zero-knowledge, first studied by [21], extends the zero-knowledge security notion to the case where multiple

* This work was supported in part by the National Natural Science Foundation of China Under Grant NO.60803128, and the 973 Program of China (2007CB311202), and the National Natural Science Foundation of China Under Grant NO. 60673069.

concurrent executions of the same protocol take place and a malicious adversary may control the scheduling of the messages and corrupt multiple provers or verifiers in order to violate the soundness or zero-knowledge properties (respectively). Unfortunately, concurrent zero-knowledge with black-box simulation requires a logarithmic number of rounds for languages outside $\mathcal{BPP}$ [7] and therefore their round-complexity is not efficient. In the Common Reference String model, in [10] it is showed that 3-round and time-efficient concurrent zero knowledge can be achieved. Surprisingly, using non-black-box techniques, Barak [1] constructed a constant round non-black-box bounded concurrent zero knowledge protocol whose time-complexity however is not efficient.

Non-Malleable Zero-Knowledge. The concept of non-malleable zero knowledge was put forward in [20]. The issue of malleability arises in the so-called man-in-the-middle setting, in which the adversary plays the role of the verifier in several proofs (left sessions) and at the same time acts as the prover in some other proofs (right sessions), having full control over the scheduling of the messages between parties. The serious problem in such scenario is that the information obtained from left interactions may help the adversary to cheat the verifier in one of the right sessions (malleability). A zero-knowledge protocol is considered non-malleable if it is immune against such problem. In [20], the authors give a $\mathcal{O}(\mathtt{log}n)$-round non-malleable zero-knowledge protocol in which the adversary interacts with only one prover. Achieving non-malleability non-interactively in the common random string model was studied in [13] and [36]. In [2], a constant-round coin-tossing protocol assuming the existence of hash functions that are collision-resistant against subexponential-time adversaries was presented, which can be used to transform non-malleable non-interactive zero-knowledge in the shared random string model into interactive non-malleable zero-knowledge in the plain model. A new constant-round non-malleable ZK with minimum assumptions was present in [33], but it failed to be extended to the *unbounded* concurrent model (but it can be viewed as a bounded-concurrent non-malleable zero knowledge argument). Note that in [2] and [33] non-black-box techniques were used, and thus the resulting protocols are very inefficient. As showed in [28], it is impossible to achieve concurrent non-malleability without set-up assumption when the inputs (statements) for honest parties are chosen *adaptively* by the adversary. Several works on this issue show the feasibility to achieve concurrent non-malleability efficiently in the common reference string model. In [22] Garay et al. defined the so-called Ω-protocol, a variant of Σ-protocol with straight line extractor, and then they show a technique to transform the Ω-protocol to a concurrently non-malleable ZK protocol. Gennaro [23] introduced multi-trapdoor commitments and presented a very efficient ZK protocol enjoying concurrent non-malleability.

The BPK Model. A growing area of research in Cryptography is that of reducing the setup assumptions under which certain cryptographic protocols can be realized. In an effort to reduce the setup assumptions required for efficient zero-knowledge arguments of knowledge that remain secure against concurrent man-in-the-middle attacks, we consider the Bare Public-Key (BPK) model of [6], a model with very relaxed set-up assumptions. Comparing with some previous model such as common reference string model and the preprocessing model this model seems to significantly reduce the set-up assumptions: it just assumes that each verifier deposits a public key pk in a public file before any interaction with the prover begins, with no need of trusted parties. Since its introduction, many papers focusing on resettable zero knowledge in this model appeared in recent years, but they do not address man-in-the-middle attacks. The BPK model has been further studied in many papers, including [29,11,17,18,39,15], where the main focus was to present constant-round concurrently sound and concurrent/resettable zero-knowledge protocols. Formally, the bare public-key model (BPK model) makes the following assumptions.

1. There are two types of entities: provers and verifiers, and the entire interaction between them can be divided into two stages; the first stage is called *preprocessing stage* and only needs to be run by verifiers; at the end of the preprocessing stage, the *proof stage* starts, where any pair of prover and verifier can interact. All algorithms have access to a public file. Provers, verifiers and the public file are defined below.
2. The *public file*, structured as a collection of records, is empty at the beginning and can be modified by the verifiers during the registration stage; the version of the public file F obtained at the end of the registration stage will be used during the proof stage.
3. An (honest) *prover* P is an interactive deterministic polynomial-time algorithm that operates in the proof stage, on input a security parameter 1^n, an n-bit string $x \in L$, an auxiliary input w, a public file F and a random tape r_p, where L is a language in $\mathcal{NP}$.
4. An (honest) *verifier* V is a pair of deterministic polynomial-time algorithms (V_1, V_2), where V_1 operates in the preprocessing stage and V_2 operates in the proof stage. On input a security parameter 1^n and a random tape r_{v1}, V_1 generates a key pair (pk, sk) and stores pk in the public file. On input pk, sk, an n-bit string x and a random tape r_{v2}, the interactive algorithm V_2 performs the interactive protocol with a prover, and outputs "accept x" or "reject x" at the end of this protocol.

Defining the Man-in-the-Middle Adversary in BPK Model. Before discussing the possibility of achieving concurrent non-malleability in the BPK model, we define the man-in-the-middle attack in this model. In a MIM attack in BPK model the adversary $\mathcal{A}$ is allowed to act as a malicious verifier in the registration stage and control all the communication between honest parties that take place in the proof stage:

Registration phase: $\mathcal{A}$, on input the security parameter 1^n, and a random string r, receives all public keys registered by *honest* verifiers and consequently registers up to a polynomial number of public keys. These keys registered by $\mathcal{A}$ are possibly different from those registered by honest verifiers. The public keys registered by $\mathcal{A}$ and honest verifiers form public file F, and $\mathcal{A}$ keeps some state information st that can be used in the proof phases.

Proof phase: Similarly as in the standard model, in this phase $\mathcal{A}$ is allowed to interact with polynomial number of honest provers (left sessions) and polynomial number of honest verifiers (right sessions), and to schedule the left sessions and the right sessions *concurrently*. Recall that in each session, the common input consists of the statement to be proven and the verifier's public key.

Let L be a language in $\mathcal{NP}$ and let R_L be the associated polynomial-time relation. Informally, we say that a system (P, V) is a *(black-box) concurrently non-malleable* *zero knowledge argument of knowledge for relation R_L* in the BPK model and in the presence of a MIM attack, if it satisfies the following requirements:

1. *Completeness and Concurrent Zero Knowledge* (as defined in [18]).
2. *Extraction:* For every left session i, if $\mathcal{A}$, mounting the MIM attack in the BPK model, convinces an honest verifier that the relevant statement $y_i \in L$ with probability p, then, there is a probabilistic polynomial-time algorithm $\mathcal{E}$ that, given access to $\mathcal{A}$, returns a witness w_i such that $(y_i, w_i) \in R_L$ with probability negligibly close to p.
3. *Simulatability of $\mathcal{A}$'s View:* There exists a probabilistic polynomial-time simulator $\mathcal{M} = [\mathcal{M}_P, \mathcal{M}_v]$ such that $\mathcal{A}$'s view (in both Registration phase and proof phase) in the real man-in-the-middle setting is computational indistinguishable from the view $\mathtt{View}[\mathcal{A}, \mathcal{M}_P, \mathcal{M}_V]$ simulated by $\mathcal{M}$.

We remark that the above extraction requirement, although sufficient for this version of our paper, is actually a simplified version of the one we use for our formal proof, which is obtained by adapting known definitions from [3,19].

Our Results. In this paper, we first formally define MIM attacks in the BPK model, and then

1. construct a 5-round concurrent non-malleable *black-box* zero-knowledge argument from any *one-way function* for any $\mathcal{NP}$ language. Under the existence of one-way permutations, we can further reduce it to a 4-round (round-optimal) argument.
2. present *efficient* instantiations of the above protocol for any $\mathcal{NP}$ language having an efficient Σ protocol under appropriate number-theoretic assumptions (e.g., the strong RSA problem), where efficiency metrics is, as done in the literature, with respect to round, communication and time efficiency. In particular, the protocol only requires $\mathcal{O}(1)$ (small constant) modular exponentiations.

We note that our results, as well as any results of security against man-in-the-middle attacks in the BPK model, crucially rely on the guaranteed authentication of the data stored in the common setup file in the BPK model, an issue apparently not specifically used in all previous works in the BPK model (not studying MIM attacks). More details on this issue can be found in [12].

Independent Works [4,31,19]. Barak et al. [4] show there are (non-constant round) concurrent non-malleable ZK arguments in plain model *if the statements to be proven by honest provers are fixed in advance*. Throughout this paper, we make no such assumption, that is, our (both positive and negative) results hold with respect to *input-adaptive* adversary which is allowed to choose statements for the honest provers adaptively (i.e., choose a statement based on the history of previous executions) during the whole interaction.

In [31], the authors study concurrent non-malleable witness-indistinguishable protocols in the standard model and use their *non-black-box-simulation* solution to this problem to present a constant-round concurrently non-malleable zero-knowledge argument of knowledge against MIM attacks in the BPK model. Compared to their construction, our main result avoids the inefficiency of all known non-black-box-simulation techniques by using only *black-box-simulation* techniques (which in fact have quite efficient instantiations), and relies on much weaker (in fact, minimal) complexity assumptions.

Finally, in [19] the authors present a formal definition of proofs of knowledge in the BPK model, they prove separation results for different notions of their witness extraction requirement, and prove that a previous protocol from [18] (as well as a generalization of it based on more general complexity assumptions) satisfies concurrent witness extraction in the BPK model. However, we note that in [19] the authors do not consider MIM attacks.

2 Our Main Protocol

In this section we present our main result: under general complexity assumptions, we present a constant-round concurrently non-malleable zero-knowledge argument of knowledge against MIM attacks for any polynomial-time relation in the BPK model. Formally, we obtain the following

Theorem 1. Let L be a language in $\mathcal{NP}$, let R_L be the associated polynomial-time relation. In the BPK model, if there exist one-way function families, there exists a constant-round (black-box) concurrently non-malleable zero-knowledge argument of knowledge against a MIM attack for R_L.

Remarks on Complexity Assumptions, Round and Time Efficiency. We note that the complexity assumption sufficient for our transformation is the weakest possible as it is also necessary for non-trivial relations, due to one of the many consequences of the main result in [32]. The above statement for our positive result focuses on strongest generality with respect to complexity assumptions and neglects round-optimality and time-efficiency of prover and verifier. However, we note that our transformation takes 5 rounds when implemented using arbitrary one-way function families, and can be implemented in only 4 rounds (which is optimal, due to a result in [29]), when implemented using one-way permutation families (see below for the assumptions and rounds required to implement Σ-protocols for $\mathcal{NP}$-complete languages). Furthermore, for some specific $\mathcal{NP}$ problem (e.g., the strong RSA problem), we can instantiate our protocol efficiently, which takes only $\mathcal{O}(1)$ (a *small* constant) modular exponentiations, as we show in Section 3.

Σ-Protocols Used in Our Construction. Σ-protocols are defined as 3-round public-coin proofs of knowledge with some nice properties: 1) *special soundness.* Let (a, e, z) be the three messages exchanged by prover P and verifier V in a session. From any statement x and any pair of accepting transcripts (a, e, z) and (a, e', z') where $e \neq e'$, one can efficiently compute w such that $(x, w) \in R$. 2) *Special honest-verifier zero-knowledge:* given the second message e and the statement x, we can compute an accepting transcript of form (a, e, z) that is computational indistinguishable from the real transcript between P and the honest V.

Many known efficient protocols, such as those in [27] and [37], are Σ-protocols. Furthermore, there is a Σ-protocol for the language of Hamiltonian Graphs [5], assuming that one-way permutation families exists; if the commitment scheme used by the protocol in [5] is implemented using the scheme in [30] from any pseudo-random generator family, then the assumption can be reduced to the existence of one-way function families, at the cost of adding one preliminary message from the verifier. We will refer to this modified protocol as a *4-round Σ-protocol*. We will also use *partially-witness-independent* Σ-protocols, where only the last message in these protocols depends on the witness of the proved statement, while all other messages only depend on (an upper bound on) the length of any such witness. Many Σ-protocols (including [5] for all of $\mathcal{NP}$) are partially-witness-independent.

Interestingly, Σ-protocols can be composed to proving the OR of atomic statements, and the resulting protocol, denoted as a Σ_{OR}-protocol, turns out to be witness indistinguishable. We refer readers to [14,8] for this type of composition.

In our main construction we will use two 4-round and partially-witness-independent Σ_{OR}-protocols for specific polynomial-time relations.

Informal Description of the Difficulties Solved by Our Transformation. The natural starting point for our transformation is the concurrently non-malleable zero-knowledge argument of knowledge in the CRS model from [22]. Informally speaking, this protocol goes as follows: the prover first generates a key pair (sk', vk') of a one-time strong signature scheme, then sends vk' and proves (using a Σ_{OR}-protocol Π_p) that either he knows a witness for the statement to be proved or he knows a valid signature of vk' under a signature verification key vk from the common reference string. In the last step, the prover signs on the whole transcript of this session using sk' and sends the signature to the verifier. Furthermore, the simulator for this protocol uses its knowledge of the secret key associated with the signature verification key vk in the common reference string.

A first way to adjust this protocol so that it might work in the BPK model is as follows. Instead of taking vk from the reference string (which is not available in the BPK model), we require the verifier to choose vk, and to give to the prover a proof (not necessarily a zero knowledge proof) that the verifier knows the secret key sk associated with vk (note that omitting this latter proof might make it easier for cheating verifiers to violate the zero-knowledge requirement). It turns out that several standard attempts to present a proof for this protocol actually fail, one major problem being in the fact that an algorithm trying to use a cheating prover to break the signature scheme seems to need itself knowledge of the signature secret key in order to be able to use the cheating prover's power. (This does not lead to a contradiction of the security of the signature scheme.)

A second adjustment to this protocol is as follows: the verifier chooses two signature verification keys vk_0, vk_1 (rather than one), and proves knowledge of at least one of the two associated secret keys to the prover. Analogously, the prover proves knowledge either of a witness for the statement to be proved or of a valid signature of vk' under any one of the two signature verification keys vk_0, vk_1. However, even after this additional fix we cannot rule out malleability interactions between the verifier's and the prover's subproofs in this protocol. (A similar situation was detailed in [18], where a specific message schedule was given, and it was showed that a malicious prover could use this schedule and malleability attacks to elude extraction attempts.) Here, we note that a fix based on the solution to this problem proposed by [18] would result in an inefficient protocol that would require $O(1)$ exponentiations for every bit of the security parameter.

Our final fix is that of requiring the prover to commit to a random string and prove the knowledge that this string is either a witness for the statement to be proved or a valid signature of vk' under one of the two signature verification keys vk_0, vk_1. We will show that with this combination of signatures and commitments we avoid the malleability attacks from [18] *efficiently* (i.e., we show instantiations of the overall transformation under appropriate number-theoretic assumptions that only require $O(1)$ additional exponentiations) and only using *general complexity assumptions*, such as the existence of one-way function families for the 5-round variant and of one-way permutation families for the 4-round variant. On the other hand, as we allow the man-in-the-middle adversary to register its own public keys, the analysis of security is more involved: our proof of the extraction property of this protocol makes a novel combined use of concurrent scheduling analysis and signature-based simulation arguments.

The formal description of our protocol uses two Σ_{OR}-protocols Π_v, Π_p, as described below. Protocol Π_v (resp., Π_p) is used by the verifier (resp., prover).

The protocol (P, V)

Security parameter: 1^n.
Common input: the public file F, n-bit string $x \in L$, an index i that specifies the i-th entry $pk_i = (ver\_k_0, ver\_k_1)$ in F, where (ver_k_0, ver_k_1) are two verification keys of two signature schemes (KG_0, Sig_0, Ver_0) and (KG_1, Sig_1, Ver_1) that are both secure against adaptive chosen message attack.
The Prover's private input: a witness w for $x \in L$.
V**'s Private input:** a secret key sk (sk is one of the signing keys corresponding to (ver_k_0, ver_k_1), i.e, $sk = sig\_k_0$ or sig_k_1.

P **Step 0:** Compute and send to V the first message of a 4-round and partially-witness-independent Σ_{OR}-protocol Π_v in which V will prove knowledge of sk that is one of the signing keys (sig_k_0, sig_k_1).
V **Step 1:** compute and send to P the second message of the 4-round Σ_{OR}-protocol Π_v; compute and send to P the first message f of a 4-round Σ_{OR}-protocol Π_p;
P **Step 1:** generate a key pair (sk', vk') for a one-time strong signature scheme (KG', Sig', Ver'); pick a random string r and compute the commitment $C = COM(w, r)$; compute and send to V the second message a of the 4-round Σ_{OR}-protocol Π_p in which P will prove to V that there exists (s, r) such that: 1) $C = COM(s, r)$, and 2) s is a witness for statement $x \in L$ *or* a valid signature of vk' corresponding to Ver_0 or Ver_1; send vk', C, a and a random string (i.e., the challenge of the Σ-protocol) as the third message of Π_v to P;
V **Step 2:** compute the fourth message of protocol Π_v according to the challenge sent by P in P **Step 1**; send this message to P; send a random challenge e of protocol Π_p to P;
P **Step 2:** check whether the transcript of protocol Π_v is accepting; if so, compute the last message z of protocol Π_p; let $tran$ denote the transcript of above interaction (i.e, the whole sequence of messages sent between parties, including z); compute the signature δ=$Sig'(sk', tran)$ and send z, δ to V;

V **Step 3:** accept if and only if (f, a, e, z) is an accepting transcript of Π_p and $Ver'(vk', \delta, tran) = 1$.

Sketch of Proofs. Due to space limitations, we just sketch the proofs of the properties of our protocol.

Completeness. The completeness property of our protocol easily follows from the completeness of the two Σ_{OR} subprotocols Π_v, Π_p used. The completeness of Π_v implies that V can run the prover's program in protocol Π_v, given any one of the secret signature keys sig_k_0, sig_k_1. The completeness of Π_p implies that P can run the prover's program in protocol Π_p, given the witness w for statement $x \in L$, and given the fact that if $x \in L$ the statement proved using Π_p is also true.

Concurrent Zero-Knowledge. The proof for the concurrent zero-knowledge property is obtained by carefully combining related proofs given in other papers (e.g., [6,22,18,15]), and we only briefly sketch it here. Briefly speaking, we show a (black-box) simulator that runs the following basic steps. First of all, upon receiving in a session a proof of knowledge according to protocol Π_v, the simulator runs the extractor associated with Π_v and obtains one of the two signature secret keys associated with the verifier's public key pk_i in public file F. This can be done by the simulator efficiently and regardless of the interleaving strategy used by the concurrent adversary, as both the verifier in protocol Π_v and the extractor associated with Π_v can be run in expected polynomial time (i.e., without need for a witness for the statement $x \in L$). Once this is achieved, the simulator can derive a witness for the statement proved using protocol Π_p, this witness being a signature secret key, and thus run on a 'straight-line mode' (i.e., without rewinding the verifier) for this particular session and all future sessions based on this public key, regardless of the interleaving strategy used by the concurrent adversary. As the only required verifier rewindings are those made by the extractor, and as at most one extractor procedure is required per public key, amounting to a total of polynomially many extractor procedures (which is a crucial advantage of using the BPK model), the entire simulation can be carried out in expected polynomial time. The above discussion assumes that the cheating verifier never aborts before completing a protocol execution. However, the difficulties arising when this is not the case were already handled in the sequential zero-knowledge, single-session setting [25], and the techniques used in this latter paper have been extended to the concurrent zero-knowledge, multi-session setting in the BPK model [18,15]. Here, the use of basic properties of the BPK model (in particular, that one can eventually simulate a given session into a straight-line mode) is once again crucial to guarantee that the simulation running time remains expected polynomial time.

In the rest of this section we focus on the most interesting property of *extraction*. (The property *simulatability of $\mathcal{A}$'s view* follows directly from the proof of extraction).

Extraction. Without loss of generality, we can assume that the MIM adversary $\mathcal{A}$ interacts with only one honest verifier V and interacts with many provers under any public keys chosen by $\mathcal{A}$ from the public file F. Then we can assume that $\mathcal{A}$ engages in at most polynomial number of left interactions and makes V accept at the end of the j-th right session on input statement $y_j \in L$ with a probability $p^j_{\mathcal{A}}(y_j)$. For such $\mathcal{A}$, we construct an extractor $\mathcal{E}$ and a polynomial-time algorithm that satisfy the extraction requirement.

The extractor in the registration phase. On input the security parameter 1^n, $\mathcal{E}$ generates two key pairs (sig_k_0, ver_k_0) and (sig_k_1, ver_k_1) for the signature scheme secure against adaptive chosen message attack; then, it registers the public key $pk = (ver\_k_0, ver\_k_1)$ on the public file and keeps sig_k_b as its secret key (it also stores sig_k_{1-b}, which will be important for the extractor to successfully extract a witness), where b is a random bit selected by $\mathcal{E}$. Then $\mathcal{E}$ runs $\mathcal{A}$'s key generation algorithm (which takes $\mathcal{E}$'s public key as input) and gets its outputs, i.e., a polynomial number of public keys that $\mathcal{A}$ registers on the public file during the registration phase. At the end of this stage, all parties are assumed to obtain the (same) public file.

The extractor in the proof phase. We first explain informally a high-level view of algorithm $\mathcal{E}$ in this phase. First of all, upon receiving in a left session an accepting conversation of a subprotocol Π_v where $\mathcal{A}$ plays as the prover within Π_v, $\mathcal{E}$ extracts the secret key associated to this execution of Π_v by multiple rewindings as in the extractor used in many other papers (e.g., the main protocol in [16]). Since there are at most a polynomial number of such secret keys, the expected number of rewindings is polynomial and this entire process takes at most expected polynomial time. These secret keys allow $\mathcal{E}$ to successfully simulate both the prover in the left sessions (as using a secret key it is possible to compute in polynomial time a witness for the statement proved using subprotocol Π_p) and the verifier in the right interactions (as sig_k_b is itself a witness for the statement proved using subprotocol Π_v) with $\mathcal{A}$ during the proof phase. A correct simulation of the interaction with $\mathcal{A}$ is necessary for $\mathcal{E}$ to later perform one more extraction in correspondence of the i-th right session, and thus obtain the desired witness.

The proof of the property of extraction consists of the following two steps:

1. With probability negligibly close to $p^j_{\mathcal{A}}(y_j)$, the above extractor extracts $w' = (s, r)$ such that $C = COM(s, r)$ and s is a witness for y_i ($(y_i, s) \in R_L$) or it is a valid signature on vk' corresponding to ver_k_0 or ver_k_1. This is guaranteed by the *special-soundness* of Π_p.
2. Show that the probability p that s is a valid signature on vk' corresponding to ver_k_0 or ver_k_1 is negligible. Informally, the proof of this step proceeds as follows. We first show that if p is non-negligible, then with non-negligible probability q, s is a valid signature on vk' corresponding to ver_k_{1-b} (which is the secret key that $\mathcal{E}$ does not use in the entire extraction process). Otherwise we are able to break either the witness indistinguishability of Π_v or the computational binding of COM; we then show that if q is non-negligible, we can construct an non-uniform algorithm to break the existential unforgeability of the signature scheme $(KG_{1-b}, Sig_{1-b}, Ver_{1-b})$. In this substep, we will use the fact that the transcript of j-th right session is different from any left session and (KG', Sig', Ver') is an one-time strong signature scheme.

3 Efficient Instantiation of Our Main Protocol

The goal of this section is to show that the concurrently non-malleable zero-knowledge argument of knowledge for any polynomial-time relation presented in Section 2 has efficient instantiations for a large subclass of languages. Specifically, if the polynomial-time relation associated with the original NP language has an efficient Σ protocol, then protocol (P, V) reduces the efficiency of the original Σ protocol only by a small constant factor. Here, consistently with most cryptography literature, by efficiency we mean the combination of round-efficiency (e.g., a 3-message Σ protocol), time-efficiency (i.e., the protocol requires prover and/or verifier to only run a small constant number of modular exponentiations) and communication-efficiency (i.e., the length of the protocol transcript is at most a constant times the input length).

Thus, for the rest of this section, we consider L to be a language in NP for which the associated polynomial-time relation admits an efficient Σ protocol; note that we do not specifically need the 4-round Σ_{OR} protocols used in Section 2 to obtain a result under minimal complexity assumptions. Then, consider the application of protocol from Section 2 to language L. By looking at its construction, we easily see that it is composed of two subprotocols:

– a Σ_{OR} protocol Π_v to prove the knowledge of one out of 2 signature secret keys associated with two independently generated signature public keys vk_0, vk_1;
– a Σ_{OR} protocol Π_p to prove that a given commitment key is a commitment to either a witness w for statement $x \in L$, or a signature of a given message based on one of the two public keys vk_0, vk_1.

First of all, one can immediately see that an efficient instantiation of both Π_v, Π_p would suffice to obtain an efficient instantiation of the entire protocol. Furthermore, since the transformations from [8,14] efficiently transform any Σ protocol into a Σ_{OR} protocol, and using the assumption that L has an efficient Σ protocol, we obtain that it would actually suffice to provide an efficient Σ protocol for the following two statements:

1. proving the knowledge of a signature secret key associated with a signature public key;
2. proving that a given commitment key is a commitment to a signature of a given message based on a given signature public key.

About item (2), we note that an efficient protocol for this task has been given in [9] with respect to a signature scheme based on the strong RSA problem. Then, a matching solution to item (1) can be obtained using the efficient proof of knowledge of the factorization of a product of 2 primes from [34].

4 Achieving Minimal Round Complexity

A result from [29] implies that, unless $\mathcal{NP} \subseteq \mathcal{BPP}$, at least 4 rounds are necessary to provide concurrently non-malleable zero-knowledge arguments of knowledge in the BPK model. As described, our protocol (P, V) from Section 2 requires 5 rounds. However, we note that by replacing the 4-round Σ_{OR} protocol based on any one-way function family with a 3-round Σ_{OR} protocol based on any one-way permutation family, we can show that the resulting protocol is a 4-round concurrently non-malleable zero-knowledge arguments of knowledge for any polynomial-time relation in the BPK model. Moreover, very recently, in [15], a technique is provided to construct a 3-round Σ_{OR} protocol based on any one-way function family in the BPK model. By using this latter protocol instead of the 4-round Σ_{OR} protocol currently used in (P, V), we obtain a round-optimal protocol under the minimal assumption of the existence of any one-way function family.

References

1. Barak, B.: How to go beyond the black-box simulation barrier. In: Proc. of IEEE Symposium on Foundations of Computer Science (FOCS 2001), pp. 106–115 (2001)
2. Barak, B.: Constant-round Coin Tossing with a Man in the Middle or Realizing the Shared Random String Model. In: Proc. of IEEE Symposium on Foundations of Computer Science (FOCS 2001), pp. 345–355 (2002)
3. Bellare, M., Goldreich, O.: On Defining Proofs of Knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
4. Barak, B., Prabhakaran, M., Sahai, A.: Concurrent Non-malleable Zero Knowledge. In: Proc. of IEEE Symposium on Foundations of Computer Science (FOCS 2006), pp. 345–354 (2006)
5. Blum, M.: How to Prove a Theorem so No One Else can Claim It. In: Proc. of International Congress of Mathematicians (ICM 1986), pp. 1444–1451 (1986)
6. Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable Zero Knowledge. In: Proc. of ACM Symposium on Theory of Computing (STOC 2000), pp. 235–244 (2000)
7. Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Concurrent Zero-Knowledge requires $\Omega(logn)$ rounds. In: Proc. of ACM Symposium on Theory of Computing (STOC 2001), pp. 570–579 (2001)
8. Cramer, R., Damgård, I., Schoenmakers, B.: Proof of partial knowledge and simplified design of witness hiding protocols. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 174–187. Springer, Heidelberg (1994)
9. Camenisch, J., Lysyanskaya, A.: A Signature Scheme with Efficient Protocols. In: Cimato, S., Galdi, C., Persiano, G. (eds.) SCN 2002. LNCS, vol. 2576, pp. 268–289. Springer, Heidelberg (2003)
10. Damgård, I.: Efficient Concurrent Zero-Knowledge in the Auxiliary String Model. In: Preneel, B. (ed.) EUROCRYPT 2000. LNCS, vol. 1807, pp. 174–187. Springer, Heidelberg (2000)
11. Deng, Y., Lin, D.: Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model under Standard Assumption. In: Pei, D., Yung, M., Lin, D., Wu, C. (eds.) Inscrypt 2007. LNCS, vol. 4990, pp. 123–137. Springer, Heidelberg (2008)
12. Deng, Y., Di Crescenzo, G., Lin, D.: Concurrently Non-Malleable Zero Knowledge in the Authenticated Public-Key Model. CoRR abs/cs/0609057 (2006)
13. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust Non-Interactive Zero Knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
14. De Santis, A., Di Crescenzo, G., Persiano, G., Yung, M.: On Monotone Formula Closure of SZK. In: Proc. of IEEE Symposium on Foundations of Computer Science (FOCS 1994), pp. 454–465 (1994)
15. Di Crescenzo, G.: Minimal Assumptions and Round Complexity for Concurrent Zero-Knowledge in the Bare Public-Key Model. In: Proc. of COCOON 2009. LNCS. Springer, Heidelberg (2009)
16. Di Crescenzo, G., Ostrovsky, R.: On Concurrent Zero Knowledge with Preprocessing. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 485–502. Springer, Heidelberg (1999)
17. Di Crescenzo, G., Persiano, G., Visconti, I.: Constant Round Resettable Zero Knowledge with Concurrent Soundness in the Bare Public-Key Model. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 237–253. Springer, Heidelberg (2004)
18. Di Crescenzo, G., Visconti, I.: Concurrent Zero Knowledge in the Public-Key Model. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 816–827. Springer, Heidelberg (2005)

19. Di Crescenzo, G., Visconti, I.: On Defining Proofs of Knowledge in the Public-Key Model. In: Proc. of Italian Conference on Theoretical Computer Science (ICTCS 2007). World Scientific, Singapore (2007)
20. Dolev, D., Dwork, C., Naor, M.: Non-malleable Cryptography. SIAM J. on Computing 30(2), 391–437 (2000)
21. Dwork, C., Naor, M., Sahai, A.: Concurrent Zero-Knowledge. In: Proc. of ACM Symposium on Theory of Computing (STOC 1998), pp. 409–418 (1998)
22. Garay, J., MacKenzie, P., Yang, K.: Strengthening Zero-Knowledge Protocols Using Signatures. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 177–194. Springer, Heidelberg (2003)
23. Gennaro, R.: Multi-trapdoor Commitments and their Applications to Non-Malleable Protocols. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 220–236. Springer, Heidelberg (2004)
24. Goldreich, O.: Foundation of Cryptography-Basic Tools. Cambridge University Press, Cambridge (2001)
25. Goldreich, O., Kahan, A.: How to Construct Constant-Round Zero-Knowledge Proof Systems for NP. Journal of Cryptology 9(3), 167–190 (1996)
26. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SIAM J. Computing 18(1), 186–208 (1989)
27. Guillou, L., Quisquater, J.: A practical zero-knowledge protocol fitted to security microprocessors minimizing both transmission and memory. In: Günther, C.G. (ed.) EUROCRYPT 1988. LNCS, vol. 330, pp. 123–128. Springer, Heidelberg (1988)
28. Lindell, Y.: Lower Bounds for Concurrent Self-Composition. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 203–222. Springer, Heidelberg (2004)
29. Micali, S., Reyzin, L.: Soundness in the Public-Key Model. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 542–565. Springer, Heidelberg (2001)
30. Naor, M.: Bit Commitment using Pseudorandomness. Journal of Cryptology 4(2), 151–158 (1991)
31. Ostrovsky, R., Persiano, G., Visconti, I.: Constant-Round Concurrent Non-Malleable Zero Knowledge in the Bare Public-Key Model. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 548–559. Springer, Heidelberg (2008)
32. Ostrovsky, R., Wigderson, A.: One-way Functions are Essential for Non-Trivial Zero-Knowledge. In: Proc. of 2nd ISTCS, pp. 3–17 (1993)
33. Pass, R., Rosen, A.: New and Improved Constructions of Non-Malleable Cryptographic Protocols. In: Proc. of ACM Symposium on Theory of Computing (STOC 2005), pp. 533–542 (2005)
34. Poupard, G., Stern, J.: Short proofs of knowledge for factoring. In: Imai, H., Zheng, Y. (eds.) PKC 2000. LNCS, vol. 1751, pp. 147–166. Springer, Heidelberg (2000)
35. Rompel, J.: One-way Functions are Necessary and Sufficient for Secure Signatures. In: Proc. of ACM Symposium on Theory of Computing (STOC 1990), pp. 387–394 (1990)
36. Sahai, A.: Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In: Proc. of IEEE Symp. on FOCS 1999, pp. 543–553 (1999)
37. Schnorr, C.: Efficient Signature Generation for Smart Cards. Journal of Cryptology 4(3), 239–252 (1991)
38. Visconti, I.: Efficient Zero Knowledge on the Internet. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 22–33. Springer, Heidelberg (2006)
39. Yung, M., Zhao, Y.: Generic and Practical Resettable Zero-Knowledge in the Bare Public-Key Model. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 129–147. Springer, Heidelberg (2007)

Approximability Distance in the Space of H-Colourability Problems

Tommy Färnqvist[⋆], Peter Jonsson[⋆⋆], and Johan Thapper

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden
{tomfa,petej,johth}@ida.liu.se

Abstract. A graph homomorphism is a vertex map which carries edges from a source graph to edges in a target graph. We study the approximability properties of the *Weighted Maximum H-Colourable Subgraph* problem (MAX H-COL). The instances of this problem are edge-weighted graphs G and the objective is to find a subgraph of G that has maximal total edge weight, under the condition that the subgraph has a homomorphism to H; note that for $H = K_k$ this problem is equivalent to MAX k-CUT. To this end, we introduce a metric structure on the space of graphs which allows us to extend previously known approximability results to larger classes of graphs. Specifically, the approximation algorithms for MAX CUT by Goemans and Williamson and MAX k-CUT by Frieze and Jerrum can be used to yield non-trivial approximation results for MAX H-COL. For a variety of graphs, we show near-optimality results under the Unique Games Conjecture. We also use our method for comparing the performance of Frieze & Jerrum's algorithm with Håstad's approximation algorithm for general MAX 2-CSP. This comparison is, in most cases, favourable to Frieze & Jerrum.

Keywords: optimisation, approximability, graph homomorphism, graph H-colouring, computational complexity.

1 Introduction

Let G be a simple, undirected and finite graph. Given a subset $S \subseteq V(G)$, a *cut* in G with respect to S is the set of edges from vertices in S to vertices in $V(G) \setminus S$. The MAX CUT-problem asks for the size of a largest cut in G. More generally, a k-cut in G is the the set of edges going from S_i to S_j, $i \neq j$, where $S_1, \ldots, S_k$ is a partition of $V(G)$, and the MAX k-CUT-problem asks for the size of a largest k-cut. The problem is readily seen to be identical to finding a largest k-colourable subgraph of G. Furthermore, MAX k-CUT is known to be **APX**-complete for every $k \geq 2$ and consequently does not admit a polynomial-time approximation scheme (PTAS).

In the absence of a PTAS, it is interesting to determine the best possible approximation ratio c within which a problem can be approximated or alternatively, the smallest

[⋆] Supported by the *National Graduate School in Computer Science* (CUGS, Sweden).

[⋆⋆] 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.

c for which it can be proved that no polynomial-time approximation algorithm exists (typically under some complexity-theoretic assumption such as **P** $\neq$ **NP**). Since the 1970s, the trivial approximation ratio 1/2 was the best known for MAX CUT. It was not until 1995 that Goemans and Williamson [16], using semidefinite programming (SDP), achieved a ratio of .878567. Until very recently no other method than SDP was known to yield a non-trivial approximation ratio for MAX CUT. Trevisan [34] broke this barrier by using algebraic graph theory techniques to reach an approximation guarantee of .531. Frieze and Jerrum [15] determined lower bounds on the approximation ratios for MAX k-CUT using SDP techniques. Sharpened results for small values of k have later been obtained by de Klerk et al. [9]. Under the assumption that the *Unique Games Conjecture* (UGC) holds, Khot et al. [24] showed the approximation ratio for $k = 2$ to be essentially optimal and also provided upper bounds on the approximation ratio for $k > 2$. Håstad [19] has shown that SDP is a universal tool for solving the general MAX 2-CSP problem over any domain, in the sense that it establishes non-trivial approximation results for all of those problems. Assuming UGC, Raghavendra's SDP algorithms have optimal performance for every MAX CSP [30], but the exact approximation ratios are not yet known. In fact, even though an algorithm (doubly exponential in the domain size) for computing these ratios for specific MAX CSP problems has emerged [31], this should be contrasted to the infinite classes of graphs our method gives new bounds for.

Here, we study approximability properties of a generalised version of MAX k-CUT called MAX H-COL for undirected graphs H. This is a specialisation of the MAX CSP problem. Jonsson et al. [20] have shown that whenever H is loop-free, MAX H-COL does not admit a PTAS, and otherwise MAX H-COL is trivial. Langberg et al. [26] have studied the approximability of MAX H-COL when H is part of the input. We present approximability results for MAX H-COL where H is taken from different families of graphs. Many of these results turn out to be close to optimal under UGC. Our approach is based on analysing approximability algorithms applied to problems which they are not originally intended to solve. This vague idea will be clarified below.

Denote by $\mathcal{G}$ the set of all simple, undirected and finite graphs. A *graph homomorphism* from G to H is a vertex map which carries the edges in G to edges in H. The existence of such a map will be denoted by $G \rightarrow H$. If both $G \rightarrow H$ and $H \rightarrow G$, the graphs G and H are said to be *homomorphically equivalent* (denoted $G \equiv H$). For a graph $G \in \mathcal{G}$, let $\mathcal{W}(G)$ be the set of *weight functions* $w : E(G) \rightarrow \mathbb{Q}^+$ assigning weights to edges of G. For a $w \in \mathcal{W}(G)$, we let $\|w\| = \sum_{e \in E(G)} w(e)$ denote the total weight of G. Now, *Weighted Maximum H-Colourable Subgraph* (MAX H-COL) is the maximisation problem with

Instance: An edge-weighted graph (G, w), where $G \in \mathcal{G}$ and $w \in \mathcal{W}(G)$.
Solution: A subgraph G' of G such that $G' \rightarrow H$.
Measure: The weight of G' with respect to w.

We remark that we consider instances where the weight functions w are given explicitly. Given an edge-weighted graph (G, w), denote by $mc_H(G, w)$ the measure of the optimal solution to the problem MAX H-COL. Denote by $mc_k(G, w)$ the (weighted) size of a largest k-cut in (G, w). This notation is justified by the fact that $mc_k(G, w) = mc_{K_k}(G, w)$. In this sense, MAX H-COL generalises MAX k-CUT.

Let $\mathcal{G}_{\equiv}$ denote the set of equivalence classes of $\mathcal{G}$ under $\equiv$. The relation $\rightarrow$ is defined on $\mathcal{G}_{\equiv}$ in the obvious way and $(\mathcal{G}_{\equiv}, \rightarrow)$ is a lattice denoted by $\mathcal{C}_S$. For a more in-depth treatment of graph homomorphisms and the lattice $\mathcal{C}_S$, see [17]. In this paper, we endow $\mathcal{G}_{\equiv}$ with a metric d defined in the following way: for $M, N \in \mathcal{G}$, let

$$d(M,N) = 1 - \inf_{\substack{G \in \mathcal{G} \\ w \in \mathcal{W}(G)}} \frac{mc_M(G,w)}{mc_N(G,w)} \cdot \inf_{\substack{G \in \mathcal{G} \\ w \in \mathcal{W}(G)}} \frac{mc_N(G,w)}{mc_M(G,w)}. \tag{1}$$

We will show that d satisfies the following property: if $M, N \in \mathcal{G}$ and mc_M can be approximated within α, then mc_N can be approximated within $\alpha \cdot (1 - d(M,N))$ and conversely, if it is **NP**-hard to approximate mc_N within β, then mc_M is not approximable within $\beta / (1 - d(M,N))$ unless **P** = **NP**. Hence, we can use d for extending known (in)approximability bounds on MAX H-COL to new and larger classes of graphs. For instance, we can apply the algorithm of Goemans and Williamson (which is intended for solving MAX K_2-COL) to MAX C_{11}-COL (i.e. the cycle on 11 vertices) and analyse how well the problem is approximated (it will turn out that Goemans and Williamson's algorithm approximates MAX C_{11}-COL within 0.79869). Furthermore, we present a linear program for $d(M,N)$ and show that the computation of $d(M,N)$ can be drastically simplified whenever M or N is edge-transitive.

The metric d is related to a well-studied graph parameter known as *bipartite density* $b(H)$ [1,3,6,18,27]: if H' is a bipartite subgraph of H with maximum number of edges, then $b(H) = \frac{e(H')}{e(H)}$, where $e(G)$ is the number of edges in a graph G. Lemma 5 shows that $b(H) = 1 - d(K_2, H)$ for edge-transitive graphs H. We note that while d is invariant under homomorphic equivalence, this is not true for bipartite density. There is also a close connection to work by Šámal on *cubical colourings* [32,33]. In fact, it turns out that for a graph H, the cubical colouring number $\chi_q(H) = 1/(1 - d(K_2, H))$.

The paper comprises two main sections. Section 2 is used for proving the basic properties of d, showing that it is well-defined on $\mathcal{G}_{\equiv}$, and that it is a metric. After that, we describe how to construct the linear program for d. In section 3, we use d for studying the approximability of MAX H-COL and investigate optimality issues, for several classes of graphs. This is done by exploiting inapproximability bounds that are consequences of the Unique Games Conjecture. Comparisons are also made to the bounds achieved by the general MAX 2-CSP-algorithm by Håstad [19]. Our investigation covers a spectrum of graphs, ranging from graphs with few edges and/or containing long shortest cycles to dense graphs containing $\Theta(n^2)$ edges. The techniques used in this paper seem to generalise to larger sets of problems. This and other questions are discussed in Section 4. Due to space considerations, some proofs have been omitted.

2 Approximation via the Metric *d*

In this section we start out by proving basic properties of the metric d, that $(\mathcal{G}_{\equiv}, d)$ is a metric space, and that proximity of graphs M, N in this space lets us interrelate the approximability of MAX M-COL and MAX N-COL. Sections 2.2 and 2.3 are devoted to showing how to compute d.

2.1 The Space $(\mathcal{G}_{\equiv}, d)$

We begin by noting that $d(M,N) = 1 - s(N,M) \cdot s(M,N)$ if we define $s(M,N)$ (for $M, N \in \mathcal{G}$) as the infimum of $\frac{mc_M(G,w)}{mc_N(G,w)}$ over all $G \in \mathcal{G}$ and $w \in \mathcal{W}(G)$. We now see that the relation $mc_M(G,w) \geq s(M,N) \cdot mc_N(G,w)$ holds for all $G \in \mathcal{G}$ and $w \in \mathcal{W}(G)$. Using this observation, one can show that $s(M,N)$ and thereby $d(M,N)$ behaves well under graph homomorphisms and homomorphic equivalence.

Lemma 1. *Let $M, N \in \mathcal{G}$ and $M \to N$. Then, for every $G \in \mathcal{G}$ and every weight function $w \in \mathcal{W}(G)$, $mc_M(G,w) \leq mc_N(G,w)$.*

Corollary 2. *If M and N are homomorphically equivalent graphs, then $mc_M(G,w) = mc_N(G,w)$. Let $M_1 \equiv M_2$ and $N_1 \equiv N_2$ be two pairs of homomorphically equivalent graphs. Then, for $i,j,k,l \in \{1,2\}$, $s(N_i, M_j) = s(N_k, M_l)$.*

Corollary 2 shows that s and d are well-defined as functions on the set $\mathcal{G}_{\equiv}$ and it is routine work to show that d is indeed a metric on this space.

We say that a maximisation problem Π can be approximated within $c < 1$ if there exists a randomised polynomial-time algorithm A such that $c \cdot Opt(x) \leq \mathbf{E}(A(x)) \leq Opt(x)$ for all instances x of Π. Proximity of graphs G and H in d allows us to determine bounds on the approximability of MAX H-COL from known bounds on the approximability of MAX G-COL:

Lemma 3. *Let M, N, K be graphs. If mc_M can be approximated within α, then mc_N can be approximated within $\alpha \cdot (1 - d(M,N))$. If it is* **NP***-hard to approximate mc_K within β, then mc_N is not approximable within $\beta / (1 - d(N,K))$ unless* **P** $=$ **NP**.

Proof. Let $A(G,w)$ be the measure of the solution returned by an algorithm which approximates mc_M within α. We know that for all $G \in \mathcal{G}$ and $w \in \mathcal{W}(G)$ we have the inequalities $mc_N(G,w) \geq s(N,M) \cdot mc_M(G,w)$ and $mc_M(G,w) \geq s(M,N) \cdot mc_N(G,w)$. As a consequence, $mc_N(G,w) \geq mc_M(G,w) \cdot s(N,M) \geq A(G,w) \cdot s(N,M) \geq mc_M(G,w) \cdot \alpha \cdot s(N,M) \geq mc_N(G,w) \cdot \alpha \cdot s(N,M) \cdot s(M,N) = mc_N(G,w) \cdot \alpha \cdot (1 - d(M,N))$. For the second part, assume to the contrary that there exists a polynomial-time algorithm B that approximates mc_N within $\beta/(1-d(N,K))$. According to the first part mc_K can then be approximated within $(1-d(N,K)) \cdot \beta/(1-d(N,K)) = \beta$. This is a contradiction unless **P** $=$ **NP**. □

2.2 Exploiting Symmetries

We will now consider general methods for computing s and d. In Lemma 4, we show that certain weight functions provide a lower bound on $mc_M(G,w)/mc_N(G,w)$, and in Lemma 5, we provide a simpler expression for $s(M,N)$ which depends directly on the automorphism group and thereby the symmetries of N. This expression becomes particularly simple when N is edge-transitive. An immediate consequence of this is that $s(K_2, H) = b(H)$ for edge-transitive graphs H.

We describe the solutions to MAX H-COL alternatively as follows: let G and $H \in \mathcal{G}$, and for any vertex map $f : V(G) \to V(H)$, let $f^{\#} : E(G) \to E(H)$ be the (partial) edge map induced by f. In this notation $h : V(G) \to V(H)$ is a graph homomorphism precisely when $(h^{\#})^{-1}(E(H)) = E(G)$ or, alternatively, when $h^{\#}$ is a total function. The set of solutions to an instance (G, w) of MAX H-COL can then be taken to be the set of vertex maps $f : V(G) \to V(H)$ with the measure $w(f) = \sum_{e \in (f^{\#})^{-1}(E(H))} w(e)$.

In the remaining part of this section, we will use this description of a solution. Let $f : V(G) \to V(H)$ be an optimal solution to the instance (G, w) of MAX H-COL. Define the weight $w_f \in \mathcal{W}(H)$ in the following way: for each $e \in E(H)$, let $w_f(e) = \sum_{e' \in (f^{\#})^{-1}(e)} \frac{w(e')}{mc_H(G,w)}$. The next result is now fairly obvious:

Lemma 4. *Let $M, N \in \mathcal{G}$ be two graphs. Then, for every $G \in \mathcal{G}$, every $w \in \mathcal{W}(G)$, and any optimal solution f to (G, w) of* MAX N-COL, $\frac{mc_M(G,w)}{mc_N(G,w)} \geq mc_M(N, w_f)$.

Let M and $N \in \mathcal{G}$ be graphs and let $A = \mathrm{Aut}^*(N)$ be the (edge) automorphism group of N. We will let $\pi \in A$ act on $\{u, v\} \in E(N)$ by $\pi \cdot \{u, v\} = \{\pi(u), \pi(v)\}$. The graph N is edge-transitive if and only if A acts transitively on the edges of N. Let $\hat{\mathcal{W}}(N)$ be the set of weight functions $w \in \mathcal{W}(N)$ which satisfy $\|w\| = 1$ and for which $w(e) = w(\pi \cdot e)$ for all $e \in E(N)$ and $\pi \in \mathrm{Aut}^*(N)$.

Lemma 5. *Let $M, N \in \mathcal{G}$. Then, $s(M, N) = \inf_{w \in \hat{\mathcal{W}}(N)} mc_M(N, w)$. In particular, when N is edge-transitive, $s(M, N) = mc_M(N, 1/e(N))$.*

Proof. Clearly, $s(M, N) \leq \inf_{w \in \hat{\mathcal{W}}(N)} \frac{mc_M(N,w)}{mc_N(N,w)} = \inf_{w \in \hat{\mathcal{W}}(N)} mc_M(N, w)$. For the first part of the lemma, it will be sufficient to prove that the following inequality holds for some $w' \in \hat{\mathcal{W}}$: $\alpha = \frac{mc_M(G,w)}{mc_N(G,w)} \geq mc_M(N, w')$. By taking the infimum over graphs G and weight functions $w \in \mathcal{W}(G)$ in the left-hand side of this inequality, we see that $s(M, N) \geq mc_M(N, w') \geq \inf_{w \in \hat{\mathcal{W}}(N)} mc_M(N, w)$.

Let $A = \mathrm{Aut}^*(N)$ be the automorphism group of N. Let $\pi \in A$ be an arbitrary automorphism of N. If f is an optimal solution to (G, w) as an instance of MAX N-COL, then so is $f_\pi = \pi \circ f$. Let $w_\pi = w_{\pi \circ f}$. By Lemma 4, $\alpha \geq mc_M(N, w_\pi)$. Summing π in this inequality over A gives $|A| \cdot \alpha \geq \sum_{\pi \in A} mc_M(N, w_\pi) \geq mc_M(N, \sum_{\pi \in A} w_\pi)$ (the straightforward proof for the last inequality is omitted). The weight function $\sum_{\pi \in A} w_\pi$ can be determined as follows.

$$\sum_{\pi \in A} w_\pi(e) = \sum_{\pi \in A} \frac{\sum_{e' \in (f^{\#})^{-1}(\pi \cdot e)} w(e')}{mc_N(G, w)} = \frac{|A|}{|Ae|} \cdot \frac{\sum_{e' \in (f^{\#})^{-1}(Ae)} w(e')}{mc_N(G, w)},$$

where Ae denotes the orbit of e under A. Thus, $w' \sum_{\pi \in A} w_\pi / |A| \in \hat{\mathcal{W}}(N)$ and w' satisfies $\alpha \geq mc_M(N, w')$ so the first part follows.

For the second part, note that when the automorphism group A acts transitively on $E(N)$, there is only one orbit $Ae = E(N)$. Then, the weight function w' is given by

$$w'(e) = \frac{1}{e(N)} \cdot \frac{\sum_{e' \in (f^{\#})^{-1}(E(N))} w(e')}{mc_N(G, w)} = \frac{1}{e(N)} \cdot \frac{mc_N(G, w)}{mc_N(G, w)}.$$

□

2.3 Computing Distances

From Lemma 5 it follows that in order to determine $s(M, N)$, it is sufficient to minimise $mc_M(N, w)$ over $\hat{\mathcal{W}}(N)$. We will now use this observation to describe a linear program for computing $s(M, N)$. For $i \in \{1, \ldots, r\}$, let A_i be the orbits of $\text{Aut}^*(N)$ acting on $E(N)$. The measure of a solution f when $w \in \hat{\mathcal{W}}(N)$ is equal to $\sum_{i=1}^{r} w_i \cdot f_i$, where w_i is the weight of an edge in A_i and f_i is the number of edges in A_i which are mapped to an edge in M by f. Note that given a w, the measure of a solution f depends only on the vector $(f_1, \ldots, f_r) \in \mathbb{N}^r$. Therefore, take the solution space to be the set of such vectors: $F = \{ (f_1, \ldots, f_r) \mid f \text{ is a solution to } (N, w) \text{ of } \textsc{Max } M\textsc{-Col} \}$. Let the variables of the linear program be $w_1, \ldots, w_r$ and s, where w_i represents the weight of each element in the orbit A_i and s is an upper bound on the solutions.

$$\begin{array}{ll} \min s & \\ \sum_i f_i \cdot w_i \leq s & \text{for each } (f_1, \ldots, f_r) \in F \\ \sum_i |A_i| \cdot w_i = 1 & \text{and } w_i, s \geq 0 \end{array}$$

Given a solution w_i, s to this program, a weight function which minimises $mc_M(G, w)$ is given by $w(e) = w_i$ when $e \in A_i$. The measure of this solution is $s = s(M, N)$.

Example 6. The *wheel graph* on k vertices, W_k, is a graph that contains a cycle of length $k-1$ plus a vertex v not in the cycle such that v is connected to every other vertex. We call the edges of the $k-1$-cycle *outer edges* and the remaining $k-1$ edges *spokes*. It is easy to see that for odd k, the wheel graphs are homomorphically equivalent to K_3. We will now determine $s(K_3, W_n)$ for even $n \geq 6$ using the previously described construction of a linear program. Note that the group action of $\text{Aut}^*(W_n)$ on $E(W_n)$ has two orbits, one which consists of all outer edges and one which consists of all the spokes. If we remove one outer edge or one spoke from W_k, then the resulting graph can be mapped homomorphically onto K_3. Therefore, it suffices to choose $F = \{f, g\}$ with $f = (k-1, k-2)$ and $g = (k-2, k-1)$ since all other solutions will have a smaller measure than at least one of these. The program for W_k looks like this:

$$\begin{array}{l} \min s \\ (k-1) \cdot w_1 + (k-2) \cdot w_2 \leq s \\ (k-2) \cdot w_1 + (k-1) \cdot w_2 \leq s \\ (k-1) \cdot w_1 + (k-1) \cdot w_2 = 1 \\ w_i, s \geq 0 \end{array}$$

The solution is $w_1 = w_2 = 1/(2k-2)$ with $s(K_3, W_k) = s = (2k-3)/(2k-2)$.

In some cases, it may be hard to determine the distance between H and M or N. If we know that H is homomorphically sandwiched between M and N so that $M \to H \to N$, then we can provide an upper bound on the distance of H to M or N by using the distance between M and N. The following result can readily be proved from the definition of s:

Lemma 7. *Let $M \to H \to N$. Then, $s(M, H) \geq s(M, N)$ and $s(H, N) \geq s(M, N)$.*

3 Approximability of MAX H-COL

Let A be an approximation algorithm for MAX H-COL. Our method basically allows us to measure how well A performs on other problems MAX H'-COL. In this section, we will apply the method to various algorithms and various graphs. We do two things for each kind of graph under consideration: compare the performance of our method with that of some existing, leading, approximation algorithm and investigate how close to optimality we can get. Let $v(G), e(G)$ denote the number of vertices and edges in G, respectively. Our main algorithmic tools will be the following:

Theorem 8. *mc_2 can be approximated within $\alpha_{GW} \approx 0.878567$ [16] and mc_k can be approximated within $\alpha_k \sim 1 - \frac{1}{k} + \frac{2 \ln k}{k^2}$ [15]. Let H be a graph. There is an absolute constant $c > 0$ such that mc_H can be approximated within $1 - \frac{t(H)}{d^2} \cdot (1 - \frac{c}{d^2 \log d})$ where $d = v(H)$ and $t(H) = d^2 - 2 \cdot e(H)$ [19].*

Here, the relation $\sim$ indicates two expressions whose ratio tends to 1 as $k \to \infty$. We note that de Klerk et al. [9] have presented the sharpest known bounds on α_k for small values of k; for instance, $\alpha_3 \geq 0.836008$. We will compare the performance of Håstad's algorithm on MAX H-COL with the performance of the algorithms for mc_2 and mc_k in Theorem 8 analysed using Lemma 3 and estimates of the distance d. For this purpose, we introduce two functions, FJ_k and *Hå*, such that, if H is a graph, $FJ_k(H)$ denotes the best bound on the approximation guarantee when Frieze and Jerrum's algorithm for MAX k-CUT is applied to the problem mc_H, while *Hå*(H) is the guarantee when Håstad's algorithm is used to approximate mc_H. We note that the comparison is not entirely fair since Håstad's algorithm was probably not designed with the goal of providing optimal results—the goal was to beat random assignments. However, it is the currently best algorithm, with known bounds, that can approximate MAX H-COL for arbitrary $H \in \mathcal{G}$. This is in contrast with the algorithms of Raghavendra [30].

To be able to investigate the eventual near-optimality of our approximation method we will rely on the Unique Games Conjecture by Khot [23]. Thus, we assume henceforth that UGC is true, which gives us the following inapproximability results:

Theorem 9 (Khot et al. [24]). *For every $\varepsilon > 0$, it is NP-hard to approximate mc_2 within $\alpha_{GW} + \varepsilon$. It is NP-hard to approximate mc_k within $1 - \frac{1}{k} + \frac{2 \ln k}{k^2} + O(\frac{\ln \ln k}{k^2})$.*

3.1 Sparse Graphs

In this section, we investigate the performance of our method on graphs which have relatively few edges, and we see that the *girth* of the graphs plays a central role. The girth of a graph is the length of a shortest cycle contained in the graph. Similarly, the odd girth of a graph gives the length of a shortest odd cycle in the graph.

Before we proceed we need some facts about cycle graphs. Note that the odd cycles form a chain in the lattice $\mathcal{C}_S$ between K_2 and $C_3 = K_3$ in the following way: $K_2 \to \cdots \to C_{2i+1} \to C_{2i-1} \to \cdots \to C_3 = K_3$. Note that $C_{2k+1} \not\to K_2$ and $C_{2k+1} \not\to C_{2m+1}$. However, after removing one edge from C_{2k+1}, the remaining subgraph is isomorphic to the path P_{2k+1} which in turn is embeddable in both K_2 and C_{2m+1}. Since C_{2k+1} is edge-transitive, Lemma 5 gives us the following result:

Lemma 10. *Let $0 < k < m$ be odd integers. Then, $s(K_2, C_k) = s(C_m, C_k) = \frac{k-1}{k}$.*

Proposition 11. *Let $k \geq 3$ be odd. Then, $FJ_2(C_k) \geq \frac{k-1}{k} \cdot \alpha_{GW}$ and Hå$(C_k) = \frac{2}{k} + \frac{c}{k^2 \log k} - \frac{2c}{k^3 \log k}$. For any $\varepsilon > 0$, mc_{C_k} cannot be approximated within $\frac{k}{k-1} \cdot \alpha_{GW} + \varepsilon$.*

Proof. From Lemma 10 we see that $s(K_2, C_k) = \frac{k-1}{k}$ which implies (using Lemma 3) that $FJ_2(C_k) \geq \frac{k-1}{k} \cdot \alpha_{GW}$. Furthermore, mc_2 cannot be approximated within $\alpha_{GW} + \varepsilon'$ for any $\varepsilon' > 0$. From the second part of Lemma 3, we get that mc_{C_k} cannot be approximated within $\frac{k}{k-1} \cdot (\alpha_{GW} + \varepsilon')$ for any ε'. With $\varepsilon' = \varepsilon \cdot \frac{k-1}{k}$ the result follows. Finally, the bound on Hå(C_k) can be obtained by noting that $e(C_k) = k$. □

Håstad's algorithm does not perform particularly well on sparse graphs; this is reflected by its performance on cycle graphs C_k where the approximation guarantee tends to zero when $k \to \infty$. We will see that this trend is apparent for all graph types studied in this section. Using results of Lai & Liu [25] and Dutton & Brigham [10], we continue with a result on a class of graphs with large girth:

Proposition 12. *Let $n > k \geq 4$. If H is a graph with odd girth $g \geq 2k+1$ and minimum degree $\geq \frac{2n-1}{2(k+1)}$, where $n = v(H)$, then $FJ_2(H) \geq \frac{2k}{2k+1} \cdot \alpha_{GW}$ and mc_H cannot be approximated within $\frac{2k+1}{2k} \cdot \alpha_{GW} + \varepsilon$ for any $\varepsilon > 0$. Asymptotically, Hå(H) is bounded by $\frac{c}{n^2 \log n} + \frac{2(n^{g/(g-1)})^3}{n^4 n^{1/(g-1)}} - \frac{2n^{g/(g-1)} n^{1/(g-1)} c}{n^4 \log n}$.*

Stronger results are possible if we restrict ourselves to planar graphs: Borodin et al. [7] have proved that if H is a planar graph with girth at least $\frac{20k-2}{3}$, then H is $(2 + \frac{1}{k})$-colourable, i.e. there exists a homomorphism from H to C_{2k+1}. By applying our method, the following can be proved:

Proposition 13. *Let H be a planar graph with girth at least $g = \frac{20k-2}{3}$. If $v(H) = n$, then $FJ_2(H) \geq \frac{2k}{2k+1} \cdot \alpha_{GW}$ and Hå$(H) \leq \frac{6}{n} - \frac{12}{n^2} + \frac{c}{n^2 \log n} - \frac{6c}{n^3 \log n} + \frac{12c}{n^4 \log n}$. mc_H cannot be approximated within $\frac{2k+1}{2k} \cdot \alpha_{GW} + \varepsilon$ for any $\varepsilon > 0$.*

Proposition 13 can be further strengthened and extended in different ways: one is to consider a result by Dvořák et al. [11]. They have proved that every planar graph H of odd-girth at least 9 is homomorphic to the Petersen graph P. The Petersen graph is edge-transitive and it is known (cf. [3]) that the bipartite density of P is $4/5$ or, in other words, $s(K_2, P) = 4/5$. Consequently, mc_H can be approximated within $\frac{4}{5} \cdot \alpha_{GW}$ but not within $\frac{4}{5} \cdot \alpha_{GW} + \varepsilon$ for any $\varepsilon > 0$. This is better than Proposition 13 for planar graphs with girth strictly less than 13. Another way of extending Proposition 13 is to consider graphs embeddable on higher-genus surfaces. For instance, the lemma is true for graphs embeddable on the projective plane, and it is also true for graphs of girth *strictly* greater than $\frac{20k-2}{3}$ whenever the graphs are embeddable on the torus or Klein bottle. These bounds are direct consequences of results in Borodin et al. [7].

We conclude the section by looking at a class of graphs that have small girth. Let $0 < \beta < 1$ be the approximation threshold for mc_3, i.e. mc_3 is approximable within β but not within $\beta + \varepsilon$ for any $\varepsilon > 0$. Currently, we know that $\alpha_3 \leq 0.836008 \leq \beta \leq \frac{102}{103}$ [9,21]. The wheel graphs from Section 2.3 are homomorphically equivalent to K_3 for odd k and we conclude (by Lemma 3) that mc_{W_k} has the same approximability properties as mc_3 in this case. For even $k \geq 6$, the following result says that $FJ_3(W_k) \to \alpha_3$ when $k \to \infty$, and *Hå*(W_k) tends to 0.

Proposition 14. *For $k \geq 6$ and even, $FJ_3(W_k) \geq \alpha_3 \cdot \frac{2k-3}{2k-2}$ but mc_{W_k} is not approximable within $\beta \cdot \frac{2k-2}{2k-3}$. Hå$(W_k) = \frac{4}{k} - \frac{4}{k^2} + \frac{c}{k^2 \log k} - \frac{4c}{k^3 \log k} + \frac{4c}{k^4 \log k}$.*

3.2 Dense and Random Graphs

We will now study *dense* graphs, i.e. graphs H containing $\Theta(v(H)^2)$ edges. For a graph H on n vertices, we obviously have $H \to K_n$. Let $\omega(G)$ denote the size of the largest clique in G and $\chi(G)$ denote the chromatic number of G. If we assume that $\omega(H) \geq r$, then we also have $K_r \to H$. Thus, if we determine $s(K_r, K_n)$, then we can use Lemma 7 to bound $FJ_n(H)$. According to Turán [35], there exists a family of graphs $T(n,r)$ such that $v(T(n,r)) = n$, $e(T(n,r)) = \lfloor (1 - \frac{1}{r}) \cdot \frac{n^2}{2} \rfloor$, $\omega(T(n,r)) = \chi(T(n,r)) = r$, and if G is a graph such that $e(G) > e(T(v(G), r))$, then $\omega(G) > r$.

Lemma 15. *Let r and n be positive integers. Then, $s(K_r, K_n) = e(T(n,r))/e(K_n)$.*

Proposition 16. *Let $v(H) = n$ and pick $r \in \mathbb{N}$, $\sigma \in \mathbb{R}$ such that $\left\lfloor \left(1 - \frac{1}{r}\right) \cdot \frac{n^2}{2} \right\rfloor \leq \sigma \cdot n^2 = e(H) \leq \frac{n(n-1)}{2}$. Then, $FJ_n(H) \geq \alpha_n \cdot s(K_r, K_n) \sim 1 - \frac{1}{r} - \frac{1}{n} + \frac{2 \ln n}{n(n-1)}$ and Hå$(H) = 2\sigma + \frac{(1-2\sigma)c}{n^2 \log n}$.*

Note that when r and n grow, $FJ_n(H)$ tends to 1. This means that, asymptotically, we cannot do much better. If we compare the expression for $FJ_n(H)$ with the inapproximability bound for mc_n (Theorem 9), we see that all we could hope for is a faster convergence towards 1. As σ satisfies $\left(1 - \frac{1}{r}\right) \cdot \frac{1}{2} \leq \sigma \leq \left(1 - \frac{1}{n}\right) \cdot \frac{1}{2}$, we conclude that *Hå*$(H)$ also tends to 1 as r and n grow. To get a better grip on how *Hå*(H) behaves we look at two extreme cases.

For a maximal $\sigma = \left(1 - \frac{1}{r}\right) \cdot \frac{1}{2}$, *Hå*$(H)$ becomes $1 - \frac{1}{n} + \frac{c}{n^3 \log n}$. On the other hand, this guarantee, for a minimal $\sigma = \left(1 - \frac{1}{r}\right) \cdot \frac{1}{2}$ is $1 - \frac{1}{r} + \frac{c}{rn^2 \log n}$. At the same time, it is easy to see that Frieze and Jerrum's algorithm makes these points approximable within α_n (since, in this case, $H \equiv K_n$) and α_r (since Turán's theorem tells us that $H \to K_r$ holds in this case), respectively. Our conclusion is that Frieze and Jerrum's and Håstad's algorithms perform almost equally well on these graphs asymptotically.

Another way to study dense graphs is via random graphs. Let $\mathcal{G}(n,p)$ denote the random graph on n vertices in which every edge is chosen randomly and independently with probability $p = p(n)$. We say that $\mathcal{G}(n,p)$ has a property A *asymptotically almost surely* (a.a.s.) if the probability it satisfies A tends to 1 as n tends to infinity. Here, we let $p = c$ for some $0 < c < 1$. For $G \in \mathcal{G}(n,p)$ it is well known that a.a.s. $\omega(G)$ assumes one of at most two values around $\frac{2 \ln n}{\ln(1/p)}$ [5,29]. It is also known that, almost surely $\chi(G) \sim \frac{n}{2 \ln(np)} \ln\left(\frac{1}{1-p}\right)$, as $np \to \infty$ [4,28]. Let us say that $\chi(G)$ is concentrated in width s if there exists $u = u(n,p)$ such that a.a.s. $u \leq \chi(G) \leq u + s$. Alon and Krivelevich [2] have shown that for every constant $\delta > 0$, if $p = n^{-1/2-\delta}$ then $\chi(G)$ is concentrated in width $s = 1$. That is, almost surely, the chromatic number takes one of two values.

Proposition 17. *Let $H \in \mathcal{G}(n,p)$. When $np \to \infty$, $FJ_m(H) \sim 1 - \frac{2}{m} + \frac{2\ln m}{m^2} + \frac{1}{m^2} - \frac{2\ln m}{m^3}$, where $m = \omega(H)$. $Hå(H) = p - \frac{p}{n} + (1-p)\cdot\frac{c}{n^2\log n} + \frac{pc}{n^3 \log n}$.*

We see that, in the limiting case, $Hå(H)$ tends to p, while $FJ_m(H)$ tends to 1. Again, this means that, for large enough graphs, we cannot do much better. With a better analysis, one could possibly reach a faster convergence rate for $FJ_m(H)$.

It is interesting to look at what happens for graphs $H \in \mathcal{G}(n,p)$ where np does not tend to ∞ when $n \to \infty$. We have the following result by Erdős and Rényi [14]: let c be a positive constant and $p = \frac{c}{n}$. If $c < 1$, then a.a.s. no component in $\mathcal{G}(n,p)$ contains more than one cycle, and no component has more than $\frac{\ln n}{c-1-\ln c}$ vertices. Now we see that if $np \to \varepsilon$ when $n \to \infty$ and $0 < \varepsilon < 1$, then $\mathcal{G}(n,p)$ almost surely consists of components with at most one cycle. Thus, each component resembles a cycle where, possibly, trees are attached to certain cycle vertices, and each component is homomorphically equivalent to the cycle it contains. Since we know from Section 3.1 that Frieze and Jerrum's algorithm performs better than Håstad's algorithm on cycle graphs, it follows that the same relationship holds in this part of the $\mathcal{G}(n,p)$ spectrum.

4 Conclusions and Open Problems

We have defined a metric on graphs that measures how well one graph can be embedded in another. While not apparent from its definition, which involves taking infima over the set of all edge-weighted graphs, we have shown that the metric can be computed practically by using linear programming. Given a graph H and known approximability properties for MAX H-COL, this metric allows us to deduce bounds on the corresponding properties for graphs close to H. In other words, the metric measures how well an algorithm for MAX H-COL works on problems MAX H'-COL, for graphs H' close to H, it also translates inapproximability results between these problems. In principle, given a large enough set of graphs with known approximability results for MAX H-COL, our method could be used to derive good bounds on the approximability of MAX H-COL for all graphs. If the known results were in fact tight and the set of graphs dense in $\mathcal{G}_{\equiv}$ (in the topology induced by d), then we would have tight results for all graphs. In this paper we have considered the graphs with known properties to be the complete graphs. We have shown that this set of graphs is sufficient for achieving new bounds on several different classes of graphs, i.e. applying Frieze and Jerrum's algorithm to MAX H-COL gives comparable to or better results than when applying Håstad's MAX 2-CSP algorithm for the classes of graphs we have considered. One possible explanation for this is that the analysis of the MAX 2-CSP algorithm only aims to prove it better than a random solution on expectation, which may leave room for strengthening of the approximation guarantee. At the same time, we are probably overestimating the distance between the graphs. It is likely that both results can be improved. This immediately suggests two clear directions of research. On the one hand, we need approximability/inapproximability result pairs for MAX H-COL on a substantially larger class of graphs. This can be seen considering for example MAX C_5-COL. The closest complete graph to C_5 is K_2, which gives us the inconsequential inapproximability bound $\alpha_{GW} \cdot 5/4 > 1$. On the other hand, we do not measure the actual distance from each graph to the closest complete graph. Instead, we embed each graph between K_2 and a cycle or between its largest clique and K_k, where k is greater than or equal to the chromatic number. In the first case, Erdős [13] has proved that for any positive integers k and l there exists a graph of chromatic number k and girth at least l. It is obvious that such graphs cannot be sandwiched between K_2 and a cycle as was the case of the graphs of high girth in Section 3.1. Additionally, there are obviously graphs with an arbitrarily large gap between largest clique and chromatic number. A different idea is thus required to deal with these graphs. In general, to apply our method more precisely, we need a better understanding of the structure of $\mathcal{C}_S$ and how this interacts with our metric d. Clearly, progress in either one of the directions will influence what type of result to look for in the other direction. In light of this discussion, two interesting candidates for research are the circular complete graphs and the Kneser graphs, see for example [17]. Both of these classes generalise the complete graphs and have been subject to substantial previous research. Partial results for d on 3-colourable circular complete graphs have been obtained by Engström [12].

We conclude the paper by considering two other possible ways to extend our results. Firstly, Kaporis et al. [22] have shown that mc_2 is approximable within .952 for any given average degree d and asymptotically almost all random graphs G in $\mathcal{G}(n, m = \lfloor \frac{d}{2} n \rfloor)$, where $\mathcal{G}(n, m)$ is the probability space of random graphs on n vertices and m edges selected uniformly at random. In a similar vein, Coja-Oghlan et al. [8] give an algorithm that approximates mc_k within $1 - O(1/\sqrt{np})$ in expected polynomial time, for graphs from $\mathcal{G}(n, p)$. It would be interesting to know if these results could be carried further, to other graphs G, so that better approximability bounds on MAX H-COL, for H such that $G \rightarrow H$, could be achieved.

Secondly, the idea of defining a metric on a space of problems which relates their approximability can be extended to more general cases. It should not prove too difficult to generalise the framework introduced in this paper to MAX CSP over directed graphs or even languages consisting of a single, finitary relation. How far can this generalisation be carried out? Could it provide any insight into the approximability of MAX CSP on arbitrary constraint languages?

References

1. Alon, N.: Bipartite subgraph. Combinatorica 16, 301–311 (1996)
2. Alon, N., Krivelevich, M.: The concentration of the chromatic number of random graphs. Combinatorica 17, 303–313 (1997)
3. Berman, A., Zhang, X.-D.: Bipartite density of cubic graphs. Discrete Mathematics 260, 27–35 (2003)
4. Bollobás, B.: The chromatic number of random graphs. Combinatorica 8(1), 49–55 (1988)
5. Bollobás, B., Erdös, P.: Cliques in random graphs. Mathematical Proceedings of the Cambridge Philosophical Society 80(419), 419–427 (1976)
6. Bondy, J., Locke, S.: Largest bipartite subgraphs in triangle-free graphs with maximum degree three. Journal of Graph Theory 10, 477–504 (1986)
7. Borodin, O., Kim, S.-J., Kostochka, A., West, D.: Homomorphisms from sparse graphs with large girth. Journal of Combinatorial Theory, ser. B 90, 147–159 (2004)
8. Coja-Oghlan, A., Moore, C., Sanwalani, V.: MAX k-CUT and approximating the chromatic number of random graphs. Random Structures and Algorithms 28, 289–322 (2005)

9. de Klerk, E., Pasechnik, D., Warners, J.: Approximate graph colouring and MAX-k-CUT algorithms based on the θ function. Journal of Combinatorial Optimization 8, 267–294 (2004)
10. Dutton, R., Brigham, R.: Edges in graphs with large girth. Graphs and Combinatorics 7(4), 315–321 (1991)
11. Dvořák, Z., Škrekovski, R., Valla, T.: Planar graphs of odd-girth at least 9 are homomorphic to the Petersen graph. To appear in SIAM Journal on Discrete Mathematics
12. Engström, R.: Approximability distances between circular complete graphs. Master's thesis, Linköping University. LITH-IDA-EX-A–08/062–SE (2008)
13. Erdös, P.: Graph theory and probability. Canadian Journal of Mathematics 11, 34–38 (1959)
14. Erdös, P., Rényi, A.: On the evolution of random graphs. Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5, 17–61 (1960)
15. Frieze, A., Jerrum, M.: Improved approximation algorithms for MAX k-CUT and MAX BISECTION. Algorithmica 18(1), 67–81 (1997)
16. Goemans, M., Williamson, D.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM 42, 1115–1145 (1995)
17. Hell, P., Nešetřil, J.: Graphs and Homomorphisms. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, Oxford (2004)
18. Hophkins, G., Staton, W.: Extremal bipartite subgraphs of cubic triangle-free graphs. Journal of Graph Theory 6, 115–121 (1982)
19. Håstad, J.: Every 2-CSP allows nontrivial approximation. In: Proceedings of the 37th Annual ACM Symposium on the Theory of Computing (STOC 2005), pp. 740–746 (2005)
20. Jonsson, P., Krokhin, A., Kuivinen, F.: Ruling out polynomial-time approximation schemes for hard constraint satisfaction problems. In: Diekert, V., Volkov, M.V., Voronkov, A. (eds.) CSR 2007. LNCS, vol. 4649, pp. 182–193. Springer, Heidelberg (2007), `http://www.dur.ac.uk/andrei.krokhin/papers/hardgap.pdf`
21. Kann, V., Khanna, S., Lagergren, J., Panconesi, A.: On the hardness of approximating MAX k-CUT and its dual. Chicago Journal of Theoretical Computer Science 1997(2) (1997)
22. Kaporis, A., Kirousis, L., Stavropoulos, E.: Approximating almost all instances of MAX-CUT within a ratio above the Håstad threshold. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 432–443. Springer, Heidelberg (2006)
23. Khot, S.: On the power of unique 2-prover 1-round games. In: Proceedings of the 34th Annual ACM Symposium on the Theory of Computing (STOC 2002), pp. 767–775 (2002)
24. Khot, S., Kindler, G., Mossel, E., O'Donnel, R.: Optimal inapproximability results for MAX-CUT and other two-variable CSPs? SIAM Journal of Computing 37(1), 319–357 (2007)
25. Lai, H.-J., Liu, B.: Graph homomorphism into an odd cycle. Bulletin of the Institute of Combinatorics and its Applications 28, 19–24 (2000)
26. Langberg, M., Rabani, Y., Swamy, C.: Approximation algorithms for graph homomorphism problems. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX 2006 and RANDOM 2006. LNCS, vol. 4110, pp. 176–187. Springer, Heidelberg (2006)
27. Locke, S.: A note on bipartite subgraphs of triangle-free regular graphs. Journal of Graph Theory 14, 181–185 (1990)
28. Łuczak, T.: The chromatic number of random graphs. Combinatorica 11(1), 45–54 (1991)
29. Matula, D.: The employee party problem. Notices of the American Mathematical Society 19, A – 382 (1972)
30. Raghavendra, P.: Optimal algorithms and inapproximability results for every CSP? In: Proceedings of the 40th annual ACM symposium on the Theory of Computing (STOC 2008), pp. 245–254 (2008)
31. Raghavendra, P., Steurer, D.: How to round any CSP (2009) (manuscript)

32. Šámal, R.: Fractional covering by cuts. In: Proceedings of the 7th International Colloquium on Graph Theory (ICGT 2005), pp. 455–459 (2005)
33. Šámal, R.: On XY mappings. PhD thesis, Charles University in Prague (2006)
34. Trevisan, L.: Max Cut and the smallest eigenvalue. CoRR, abs/0806.1978 (2008)
35. Turán, P.: On an extremal problem in graph theory. Matematicko Fizicki Lapok 48, 436–452 (1941)

On Random Ordering Constraints

Andreas Goerdt

Technische Universität Chemnitz, Fakultät für Informatik
Straße der Nationen 62, 09107 Chemnitz, Germany
goerdt@informatik.tu-chemnitz.de
http://www.tu-chemnitz.de/informatik/TI/

Abstract. Ordering constraints are analogous to instances of the satisfiability problem in conjunctive normalform, but instead of a boolean assignment we consider a linear ordering of the variables in question. A clause becomes true given a linear ordering iff the relative ordering of its variables obeys the constraint. The naturally arising satisfiability problems are **NP**-complete for many types of constraints.

The present paper seems to be one of the first looking at random ordering constraints. Experimental evidence suggests threshold phenomena as in the case of random k-SAT instances. We prove first that random instances of the cyclic ordering and betweenness constraint have a sharp threshold for unsatisfiability. Second, random instances of the cyclic ordering constraint are satisfiable with high probability if the number of clauses is $\leq 1 \times \sharp$variables.

Keywords: Algorithms, logic, random structures, probabilistic analysis, ordering.

1 Introduction

1.1 Results

Let V always be a set of n variables. A 3-clause over V is an *ordered* 3-tuple (x, y, z) consisting of three different variables. A formula, also called ordering constraint is a set of clauses. Given a linear ordering of all n variables a clause evaluates to true if its variables satisfy a given constraint with respect to the ordering. A formula becomes true when all its clauses are true. This is the satisfiability problem the present paper deals with.

We consider random ordering constraints. The random instance $F(V, p)$ or the corresponding probability space is obtained by picking each of the $n(n-1)(n-2)$ clauses independently with probability p. Thus $F(V, p)$ is analogous to the well known random graph $G(n, p)$. As common in the theory of random structures this paper deals with properties holding with high probability, that is $1 - o(1)$ when n becomes large and $p = p(n)$ is a given function. We deal with the case $p = a/n^2$ for constant a. The definition of the random instance $F(V, p)$ is analogous to random 3-SAT formulas. A selection of literature on random 3-SAT is [3] ,[8], [20].

The clause (x, y, z) interpreted as *cyclic ordering constraint* is true iff $x < y < z$ or $z < x < y$ or $y < z < x$: There is a cyclic permutation of (x, y, z) which is monotonously increasing with respect to the ordering. Clauses which are cyclic permutations of each other are equivalent, syntactically we distinguish them. (Our results do not depend on this.) The associated satisfiability problem is **NP**-complete [13]. In case of the *betweenness* problem the clause (x, y, z) is true iff y is between x and z, that is we have $x < y < z$ or $z < y < x$. The satisfiability problem is **NP**-complete, too [21]. Basic observations which follow readily from the literature as shown in Section 2 collects

Proposition 1. *For random instances $F(V, p)$ with $p = a/n^2$ the following events have high probability:*

(a) For $a < 0.8$ the cyclic ordering and betweenness instance is satisfiable.
(b) For $a > 9 \cdot \ln 3 \approx 9.88$ the cyclic ordering instance is unsatisfiable.
(c) For $a > 4 \cdot \ln 2 \approx 2.77$ the betweenness instance is unsatisfiable.

The expected number of clauses of $F(V, p)$ with $p = a/n^2$ is an. Moreover, the number of clauses is asymptotically equal to an with high probability. Techniques as detailed on pages 34/35 of [5] show that analogous results hold for the random instance obtained by picking a random set of exactly an clauses. This applies to all of our results.

Experiments show that the random betweenness constraint is satisfiable for an clauses with $a < 1.5$. It usually is unsatisfiable for $a > 1.6$. Our experiments are for $n \leq 300$. For the cyclic ordering constraint we observe the same threshold phenomenon for approximately $3n$ clauses (and $n \leq 150$.) No theoretical results concerning these observations or even random ordering constraints in general seem to exist. We prove that the transition from satisfiability to unsatisfiability is by a sharp threshold.

Theorem 2. *There exists $C = C(n)$, $0.8 \leq C \leq 9.88$ (2.77) such that for each constant $\varepsilon > 0$ the cyclic ordering constraint (betweenness constraint) $F(V, p)$ is unsatisfiable with high probability for $p = (C + \varepsilon)/n^2$ and satisfiable for $p = (C - \varepsilon)/n^2$.*

We have only bounds within which the threshold value C may vary depending on n. The situation is analogous to random 3-SAT or k-colourability of random graphs, see [11]. As in these cases the proof of Theorem 2 is an application of the threshold criterion of Friedgut [10] which gives no information of the value.

Given Proposition 1, it seems non-trivial to show that $F(V, a/n^2)$ is satisfiable with high probability for any a substantially larger than 0.8. Analyzing a heuristic algorithm which tries to find a satisfying ordering we make first progress and prove

Theorem 3. *For $p = a/n^2$ with constant $a \leq 1$ the random cyclic ordering instance $F(V, p)$ is satisfiable with probability $\geq \varepsilon$ for a constant $\varepsilon > 0$.*

Theorem 3 together with Theorem 2 implies a high probability result.

Corollary 4. *The random cyclic ordering constraint with $p = a/n^2$ and $a < 1$ is satisfiable with high probability.*

1.2 On the Literature

Ordering constraints differ from traditional constraints like k-SAT or more general constraints in that the underlying assignment must be an ordering of all variables. This means that each variable can receive one out of n values. But, each of the n values can only be used once. Altogether we have $n! \approx 2^{n \log n} >> 2^n$ many assignments.

Beyond random k-SAT there is much work on random constraints with finite domain. Only a selection of the literature, in part due to Michael Molloy is [18], [19], [16] . The paper [18] points out that thresholds may have algorithmic relevance: Random instances at thresholds often have some algorithmic hardness which makes them attractive as test cases for algorithms. No systematic experimental studies of random ordering constraints seem to exist. Our experiments appear to confirm that instances closer to the threshold become harder.

Many real world notions involve ordering. Therefore knowledge representation formalisms often offer ordering constraints: In [15] the cyclic ordering constraint occurs. In [6] an extended betweenness constraint is used to describe some biological situation.

Worst case aspects of ordering constraints are investigated in [14]. The authors try to classify ordering constraints by their complexity. For some cases a dichotomy, either polynomial time solvable or **NP**-hard is shown. The recent [4] has more comprehensive results. The optimization version of the betweenness constraint is treated in [1] and [7].

2 Observations

We prove Proposition 1. The 2-core of a constraint F is the (unique) largest subformula of F in which each variable which occurs at all occurs at least twice, that is has degree ≥ 2. The 2-core is non-trivial iff it is not empty. The next proposition is easy to see: The satisfying order can be built one by one.

Proposition 5. *Let F be a constraint which has only the trivial 2-core. F interpreted as a cyclic ordering (betweenness) constraint is satisfiable.*

Satisfiable constraints with non-empty 2-core are easy to find: Consider the ordering $x < y < z < u$ and the constraint (x, y, z), $(x, z, u), (y, z, u)$.

Molloy's Theorem 1.2, on page 666 of [20], for graphs and hypergraphs is a threshold theorem for the appearance of a non-trivial 2-core in $F(V,\ a/n^2)$:

Proposition 6. *Let* $t = \min_{0<f<1}(-\ln(1-f)/(3 \cdot f^2))$.
(a) If $a < t$ the 2-core of $F(V,\ a/n^2)$ is empty with high probability.
(b) If $a > t$ the random constraint $F(V\ , a/n^2)$ has a 2-core of linear size with high probability.

First insight into the curve $-\ln(1-f)/(3 \cdot f^2)$ can be obtained by looking at the logarithm series $-\ln(1-f) = f + f^2/2 + f^3/3 + \cdots$ for $0 \leq f < 1$. We observe that

$0.8 < t < 0.82$ Therefore the cyclic ordering (betweenness) constraint $F(V, a/n^2)$ is satisfiable with high probability for $a \leq 0.8$ proving Proposition 1 (a) .

Concerning unsatisfiability we can use a standard first moment argument (known from random k-SAT): We consider the cyclic ordering constraint. Given a fixed linear ordering we have asymptotically $n^3/2$ clauses which are false under a given ordering. As clauses are picked independently the probability that a random $F(V, \ a/n^2)$ is true under this ordering is $(1-a/n^2)^{n^3/2} < \exp(-an/2)$. The expected number of linear orderings satisfying $F(V, a/n^2)$ is $< \ n! \times \exp(-an/2)$. This approaches 0 only if $a \geq \ln n$. And (by Markov) $F(V, \ a/n^2)$ is unsatisfiable with high probability. To prove Proposition 1 (b) and (c) we need

Lemma 7. *(a) Let F be a satisfiable cyclic ordering constraint. There exists a partition of the variables into 3 sets K, L, M, each with $n/3$ variables, such that F has no clause from $K \times M \times L$ or its cyclic permutations.*

(b) Let F be a satisfiable betweenness constraint. There exists a partition of the variables into two sets K, L, each with $n/2$ variables, such that F has no clause from $K \times L \times K$ or $L \times K \times L$.

Proof. (a) Let K be the first third of the variables of the satisfying ordering, L the second third, and M the last third of the variables. (b) As (a) using the first and second half of a satisfying ordering instead.

Proposition 1 (b) and (c) can now be proved by bounding the expected number of partitions as in Lemma 7. We have only $< 3^n$ ($< 2^n$ for the betweenness constraint) partitions altogether. We can proceed similarly as above.

3 The Threshold

We refer to [11] which we apply to the random formulas here. They can be seen as random 3-uniform directed hypergraphs. Unsatisfiability of a cyclic ordering (betweenness) constraint is a monotone property, as it is preserved under the addition of clauses. Moreover, unsatisfiability is a property which is invariant under permutation of the variables. Therefore it satisfies the symmetry properties necessary for the application of the criterion from [11].

For a formula F we abbreviate the property that F is unsatisfiable by UNSAT(F), in case of the random formula $F = F(V, p)$ we also write UNSAT(V, p). SAT(F) means: F is satisfiable. We say that the event UNSAT(V, p) has a coarse threshold iff we have a critical probability $p_c = p_c(n)$ and constants $\varepsilon, \varepsilon'$ and δ such that $\varepsilon <$ Pr[UNSAT(V, p_c)] $< 1 - \varepsilon$ and Pr[UNSAT($V, (1 + \delta)p_c$)] $< 1 - \varepsilon'$. In our case we always have $p_c = a/n^2$. This implies that a threshold is coarse iff adding an arbitrarily small, but linear number of clauses to $F(V, p_c)$ does *not* yield unsatisfiability with high probability (cf. the remark after Proposition 1.)

A variety of types of random formulas is needed: The random instance $F(V, p) \cup F(V, q)$ is constructed by first picking $F(V, p)$ and second and independently adding $F(V, q)$ to the formula picked. For a given formula M the random formula M^* is a random copy of M (formula isomorphic to M) on variables from V. A formula M is balanced iff its average degree ($= \sum_x$ degree of x in M / $\sharp$variables of M) is at least as large as that of any subformula of M. The formula $F(V,p) \cup M^*$ is the union of two independent instances of $F(V,p)$ and M^*. There exist several formulations of Friedgut's threshold criterion. Most convenient to apply is

Fact 8 (cf. Corollary 2.3 of [12]). *If UNSAT(V,p) has a coarse threshold then there exist*

- *a (critical) probability $p_c = p_c(n)$,*
- *a balanced formula M and a constant $b > 0$, and*
- *a constant $\varepsilon > 0$*

such that for infinitely many n we have

- $\varepsilon <$ *Pr [UNSAT(V, p_c)]* $< 1 - \varepsilon$,
- *the expected number of copies of M in $F(V, p_c)$ is $\geq b$, and*
- *Pr [UNSAT$(F(V, p_c) \cup M^*)$] $-$ Pr [UNSAT$(F(V, p_c) \cup F(V, \varepsilon p_c))$]* $\geq \varepsilon$.

Fact 8 says that in case of a coarse threshold there exists a fixed M such that adding M^* to $F(V, p_c)$ is more likely to make $F(V, p_c)$ unsatisfiable than adding a (small but) linear number of random clauses. We show that any M as in Fact 8 with Pr[UNSAT$(F(V, p_c) \cup M^*)$] $\geq$ Pr[UNSAT$(F(V, p_c))$] $+ \varepsilon$, necessary for the last inequality to hold, cannot satisfy the last inequality.

The subsequent Lemma 9 is crucial towards the threshold proof. We split the set of variables V into two disjoint subsets U and W. U is a set of k variables where k is a fixed constant independent of n, and W contains the remaining $n - k$ variables. The random instance $F(U, W, p)$ is obtained by picking each clause with at least one variable from U independently with probability p.

For fixed formulas F over W and F^* over U (think of F^* as an instance of the random formula M^*) we consider the random formula $F \cup F^* \cup F(U, W, p)$, over $V = W \cup U$, with $p = a/n^2$, where $a_1 < a < a_2$ for constants $a_1, a_2 > 0$ $a = a(n)$. We assume that F and F^* are both satisfiable as cyclic ordering (betweenness) constraints. Let $x_1 < x_2 < \ldots < x_{n-k}$ be an ordering satisfying F and $y_1 < y_2 < \ldots < y_k$ for F^* (where $\{x_1, \ldots, x_{n-k}\} = W$, $\{y_1, \ldots, y_k\} = U$.) Let UNSAT abbreviate the event UNSAT$(F \cup F^* \cup F(U, W, p))$ as cyclic ordering (betweenness) constraint. Recall that $F(W, p)$ is the usual random instance with variables from W. Let n be sufficiently large, $\varepsilon > 0$ a constant inpendent of n. We have

Lemma 9. *If Pr[UNSAT] $> \varepsilon$ then for any constant $\delta > 0$ the random instance $F \cup F(W, \delta \cdot p)$ interpreted as a cyclic ordering (betweenness) constraint is unsatisfiable with* high *probability.*

Proof. Pr[UNSAT] $> \varepsilon$ means that Pr[UNSAT$(F \cup F^* \cup F(U, W, p))$] $> \varepsilon$. As F and F^* are satisfiable and refer to disjoint sets of variables, unsatisfiability can only be caused by $F(U, W, p)$ (which connects F and F^*.) If $F' = F(U, W, p)$

causes unsatisfiability then $F \cup F^* \cup F'$ is in particular false under all orderings with $W_1 < U < W_2$ for $W_1 \cup W_2 = W$. That is the variables of U are adjacent. This implies: When we substitute the variables from U in F' with an arbitrary variable x from W we get an unsatisfiable formula over W. (Clauses with 2 variables form U occur only with probability $o(1)$ as k is a constant.) The random instance $F(W, \delta \cdot p)$ contains clauses which are obtained by substituting U with x *and* cause unsatisfiability with probability bounded below by a constant > 0 (smaller than ε.) This is the case as Prob[UNSAT] $> \varepsilon$ and $|U| = k$ is fixed. Thus the expected number of variables $x \in W$ such that the clauses with x from $F(W, \delta \cdot p)$ cause unsatisfiability is (small but) linear in $|W|$. For two variables $x \neq y$ the clauses from $F(W, \delta \cdot p)$ with x are almost independent from those with y. We get concentration at the expectation with Tschebycheff's inequality. And with high probability $F \cup F(W, \delta \cdot p)$ is unsatisfiable.

Proof sketch of Theorem 2. First, we observe that the formula M from Fact 8 must be satisfiable. The assumptions in this fact imply that M only has an empty 2-core. Second, we observe that the very last item of Fact 8 implies the existence of a satisfiable formula F with the property: The addition of M^* to F is more likely to yield an unsatisfiable formula than the addition of each clause with probability εp_c. Such an F contradicts Lemma 9.

4 The Cyclic Ordering Constraint

We come to the proof of Theorem 3.

4.1 The Reduction

In addition to 3-clauses as above we need 2-clauses which are ordered pairs (x, y) consisting of two different variables. Their interpretation is $x < y$ throughout, 3-clauses are interpreted as cyclic ordering constraints from now on. The graph associated to the set of 2-clauses E is called G_E. It is obtained by viewing each 2-clause (x, y) as the directed edge $x \to y$. The random instance $F(V, p, q)$ is the union of two independent instances of $F(V, 3, p)$ and $F(V, 2, q)$, with $F(V, 3, p) = F(V, p)$, and $F(V, 2, q)$ picks each 2-clause with q. For $A, B, C \subseteq V$ we use the notation $(A, B, C) = A \times B \times C$ and $(A, B) = A \times B$. The reduction used is

Definition 10. *Let $F = D \cup E$ be a constraint with 2-clauses E and 3-clauses D. Let A, B be a partition of the set of variables V into two disjoint sets with $A \cup B = V$. If E contains clauses from both, (A, B) and (B, A) the constraints F_A and F_B are not defined. Otherwise the constraint F_A over A is defined as:*
The 2-clauses of E which belong to (A, A) are in F_A.
Let $(x, y, z) \in D$ with at least two variables belonging to A, then

- $(x, y, z) \in (A, A, A) \Rightarrow (x, y, z) \in F_A$, $(x, y, z) \in (A, A, B) \Rightarrow (x, y) \in F_A$,
- $(x, y, z) \in (A, B, A) \Rightarrow (z, x) \in F_A$, $(x, y, z) \in (B, A, A) \Rightarrow (y, z) \in F_A$.

Constraint F_B is defined by exchanging the roles of A and B.

The 2-clauses from F belonging to (A, B) (resp. (B, A)) get lost when constructing F_A and F_B. When G_E has a cycle the whole cycle must either belong to A or to B in order that F_A and F_B are defined. In this case F is unsatisfiable anyway. The next Lemma states the satisfiability properties preserved by the reduction. It follows directly from Definition 10.

Lemma 11. *F without 2-clauses from (B, A) is satisfiable by a linear ordering with $A < B$ iff F_A and F_B are satisfiable.*

The following notions are the natural graph theoretic ones as induced by G_E, the graph associated to the set of 2-clauses E. The outdegree of a variable x is $\mathrm{Odeg}_E(x) = \sharp\text{clauses } (x, -) \in E$. The set of neighbors of x is $N_E(x) = \{y|(x, y) \in E\}$.

Definition 12. *(a) The boundary $B_0 = B_{0,E}$ is $B_0 = \{x|\ Odeg(x) = 0\}$. The boundary $B_1 = B_{1,E}$ is $B_1 = \{x|\ N_E(x) \subseteq B_0$ and $N_E(x)$ is not empty$\}$.*

(b) $B = B_E = B_0 \cup B_1$ is the boundary. The interior is $Int = Int_E = V \setminus B$.

(c) When F consists of 3-clauses D and 2-clauses E we let $B_F := B_E$ .... and so on for all these notions.

The proof of Theorem 3 is an analysis of the reduction as visualized in Figure 1. F is a formula over V consisting only of 3-clauses. A' is a fixed set of one half of the variables from V and A'' is the other half.

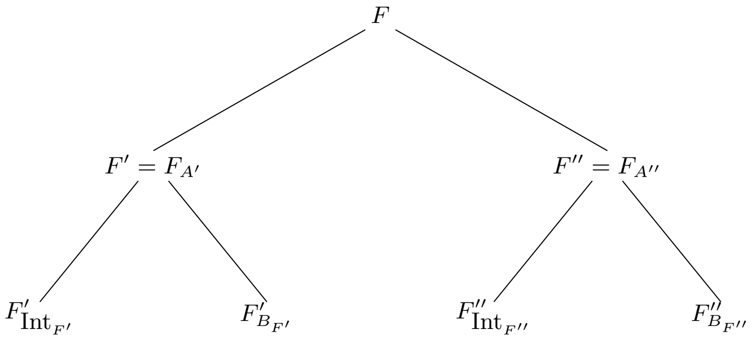


Fig. 1.

Note that all formulas of the tree comply with Definition 10, as any formula F' has no 2-clauses from $(B_{F'}, \mathrm{Int}_{F'})$ by Definiton 12. We prove for $F = F(V, 3, 1/n^2)$: The probability that *all* formulas at the leaves are simultaneously satisfiable does not go to 0. Then F is satisfiable by applying Lemma 11 twice.

Theorem 13 concerns the first reduction of Figure 1. The proof relies on the fact that F' and F'' are independent instances of $F(A,\ (1/4) \cdot a/(n/2)^2,\ (3/4) \cdot a/(n/2))$, with $|A| = n/2$. This holds as A' and A'' are sets fixed beforehand.

Theorem 13. *Let a be a constant. $F(V,3,a/n^2)$ is satisfiable with probability $> \varepsilon^2 > 0$ if $F(V,b/n^2,c/n)$ with $b = (1/4)a$ and $c = (3/4)a$ is satisfiable with probability $> \varepsilon > 0$.*

4.2 The Second Step of the Reduction

The subsequent Theorem 14 together with Theorem 13 implies Theorem 3.

Theorem 14. *For $F = F(V,b/n^2,c/n)$ with $b = 1/4$ and $c = 3/4$ we have*
$Pr[SAT(F_{Int})$ and $SAT(F_B)]$ is bounded strictly above 0.

To prove Theorem 14 we associate a (multi-)graph to F_B and to F_{Int}.

Definition 15. *Let F be a constraint of 2- and 3-clauses. The directed (multi-)graph associated to F is denoted by G_F. It has as vertices the variables of F. Its edges are the 2-clauses of F, and for each 3-clause $C = (x,y,z)$ of F the three edges (x,y), (y,z) and (x,z), each labelled with C.*

By a cycle of length s in G_F (or F) we mean set of labelled edges (x_1,x_2), $(x_2,x_3), \dots, (x_s,x_1)$ in G_F with $x_1,\dots,x_s$ all different. Always $s \geq 2$. Obviously, F is satisfiable if G_F is cycle free. In this case a satisfying ordering can be found by topologically sorting the vertices of G_F. We proceed to show that F_B and F_{Int} are both cycle free with probability not going to 0. Note that the edges of the associated (multi-)graphs are not necessarily independent. Moreover, B and Int and thus F_B and F_{Int} are dependent.

Following [9] who treat the undirected case, [22] seems to be the first work on cycles in the classical directed random graph $G(n,c/n)$ and shows: For constant $0 \leq c \leq 1$ $\Pr[G(n,c/n)$ has a directed cycle $] = c(1+o(1))$. In [17] a giant component threshold is shown: For constant $c > 1$ $G(n,c/n)$ has a strongly connected component of linear size. For constant $c < 1$ strongly connected components are only of logarithmic size. These results provide some intuition to us.

Usually one thinks of $F = F(V,p,q)$ as being generated by picking each 3-clause with its probability p first and then each 2-clause with q. We use a different generation process which in the end yields the right distribution. The process is motivated by the following formula. Given disjoint sets B_0 and B_1 and considering $E = F(V,2,q)$ we see that $\Pr[B_{0,E} = B_0$ and $B_{1,E} = B_1] =$

$$= \prod_{x \in B_0} (1-q)^{n-1} \times \prod_{x \in B_1} \Pr[N_E(x) \subseteq B_0 \text{ and } |N_E(x)| \geq 1] \times$$
$$\times \prod_{x \in \text{Int}} \Pr[N_E(x) \cap (V \setminus B_0) \text{ is not empty}]. \qquad (1)$$

This is the case as edges of G_E, the directed graph of E, starting at different vertices are independent.

We have $x \in B_{0,E}$ with probability $(1-q)^{n-1}$. Let B_0 be a fixed set, $b_0 = |B_0|$. Condition on $B_{0,E} = B_0$ and assume that the 2-clauses E arrive. For any $x \notin B_0$ we get $x \in B_{1,E}$ with $(1-(1-q)^{b_0})(1-q)^{n-1-b_0}$. In the same way we get for $x \notin B_0$ that $x \in \mathrm{Int}_E$ with $1-(1-q)^{n-1-b_0}$. This suggests the following generation process of $F = F(V,p,q)$. The process has five independent steps, and the probabilities multiply. (Recall Definition 12 (c).)

1. For $x \in V$ decide $x \in B_{0,F}$ with probability $(1-q)^{n-1}$ and $x \notin B_{0,F}$ with probability $1-(1-q)^{n-1}$. We abbreviate $b_0 = |B_{0,F}|$.

2. For $x \notin B_{0,F}$ decide independently $x \in B_{1,F}$ with

$$\frac{(1-q)^{n-1-b_0} \cdot (1-(1-q)^{b_0})}{1-(1-q)^{n-1}}, \text{ and } x \notin B_{1,F} \text{ with } \frac{1-(1-q)^{n-1-b_0}}{1-(1-q)^{n-1}}.$$

3. We generate the 2-clauses in the boundary $B = B_0 \cup B_1$. For each $x \in B_1$ we consider all clauses in (x, B_0). For $b_0 \geq k \geq 1$ every set of k such clauses has probability $q^k(1-q)^{b_0-k} \;/\; (1-(1-q)^{b_0})$. Add such a random set.

4. We generate the 2-clauses with at least one variable from $\mathrm{Int} = V \setminus B$. For $x \notin B$ we consider all clauses in $(x, V \setminus B_0)$. Each set of $n-1-b_0 \geq k \geq 1$ of these clauses has probability $q^k(1-q)^{n-1-b_0-k} \;/\; (1-(1-q)^{n-1-b_0})$. We add such a random set.

5. We add each clause from (Int, B_0) with probability q independently.

Formula (1) implies that this process generates the 2-clauses from $F(V,p,q)$. The 3-clauses are added independently. To generate F conditional on B_0, B_1 we start the process with Step 3. The subsequent Lemma holds as the clauses contributing to F_B are disjoint from those contributing to F_{Int}.

Lemma 16. *Let $F = F(V,p,q)$ and let B_0, B_1 be given disjoint sets of variables. Conditional on the event $B_{0,F} = B_0$ and $B_{1,F} = B_1$ the constraints F_{Int} and F_B are independent of each other.*

Notation: $\beta_0 = \exp(-c)$, $\beta = \exp(-c(1-\beta_0)) = \exp\left(-c(1-\exp(-c))\right)$,

$$\beta_1 = \beta - \beta_0, \text{ and } \gamma = 1-\beta.$$

Concentration results:

Lemma 17. *For $F = F(V, b/n^2, c/n)$ we have with high probability the following asymptotic (that is, a suitable factor of $1+o(1)$ must be appended) equalities:*

$$|B_{0,F}| = \beta_0 n, \;\; |B_F| = \beta n, \;\; |B_{1,F}| = \beta_1 n, \;\; |Int_F| = \gamma n.$$

Proof. The first equation follows from independence. The second equation follows from independence, conditioning on the fact that $B_{0,F}$ is a set of variables satisfying the first equation. We only need a standard second moment argument for the concentration. The rest is by definition of B_1, Int .

Given a formula F, NOCYC(F) means that F (or G_F) has no cycle.

Lemma 18. *We consider $F(V, b/n^2, c/n)$ with $b = 1/4, c = 3/4$.
Let $|B_0| = \beta_0 n(1+o(1))$ and $B_1 \cap B_0 = \emptyset$ with $|B_1| = \beta_1 n(1+o(1))$ variables. Let F be a random instance conditional on $B_{0,F} = B_0$ and $B_{1,F} = B_1$. The following conditional probabilities are both $> \varepsilon > 0$, a constant:*

(a) $Pr[NOCYC(F_{Int})]$ and (b) $Pr[NOCYC(F_B)]$.

Given the preceding three lemmas and the remark after Definition 15, Theorem 14 follows by combining them. Before presenting the proof of Lemma 18 some values (by pocket calculator): $\beta_0 = \exp(-c) \approx 0.4723$, $c(1-\beta_0) \approx 0.3957$,
$\beta = \exp(-c(1-\beta_0)) \approx 0.67319$, $\beta_1 \approx 0.2$, $\gamma \approx 0.3268$.

Proof of Lemma 18 (a). Let G_{Int} be the multigraph of F_{Int}. For $x, y \in$Int the edge (x, y) is induced by $3(n-1)(n-2)$ 3-clauses and by the 2-clause (x, y) (Definitions 10 and 15). The expected number of ways this edge occurs in G_{Int} is equal to

$$\frac{b}{n^2} \cdot 3(n-1)(n-2) + \frac{c}{n} \cdot \frac{1}{1-(1-c/n)^{n-1-|B_0|}} = \left(3b + \frac{c}{\gamma}\right) \frac{1}{n}(1+o(1)). \quad (2)$$

We abbreviate $d := 3b + (c/\gamma) = (3/4) \cdot (1 + 1/(1-\beta))$. The bound $d\gamma = (3/4) \cdot (2-\beta) < 1$ is crucial: Our β above yields $d\gamma \approx 0.9951$.

W.l.o.g. we restrict attention to cycles which do not contain two edges induced by the same 3-clause from F. This is easy to see, and the edges of a cycle are independent. Therefore the expected number of cycles of length $n \geq s \geq 2$ is $\leq (\gamma n)^s/s \cdot (d/n)^s = (d\gamma)^s/s$. We have enlarged d by a small constant, to get rid of the $1+o(1)-$factor from (2). The expected number of cycles of any length is $\leq -\ln(1-d\gamma) - d\gamma$ (a constant) by the logarithm series. We are in a situation similar to the directed random graph $G(n, c/n)$ with $c < 1$. The proof in [22] shows: Different cycles are essentially independent. From this the result mentioned after Definition 15 follows. This is tedious, and we use the Lovasz Local Lemma from pages 53/54 of [2], instead.

First, as $d\gamma < 1$ the expected number of cycles of length $> S$ gets arbitrarily small for large, constant S. We restrict attention to cycles of length $2 \leq s \leq S$. Some notational preparation: For $c_s = \sharp$candidate cycles of length s, we have $c_s \leq (\gamma n)^s/s$. For $1 \leq j \leq c_s$ let $C_{s,j}$ be the event that the j'th cycle of length s is present. The $C_{s,j}$ correspond to the events A_i in the Lovasz Local Lemma. We set $x_s := (d/n)^s$ – enlarging d slightly over its original definition above. The event $C_{s,j}$ has stochastic dependencies only with those events $C_{t,-}$ whose cycle has variables in common with the cycle of $C_{s,j}$. As $s, t \leq S$, there are $O((\gamma n)^{t-1})$ such events and

$$x_s \cdot \prod_{t=2}^{S} (1-x_t)^{O(n^{t-1})} = x_s(1-o(1)).$$

As $\Pr[C_{s,-}] \leq x_s(1-o(1))$ for all s, the assumptions of the Local Lemma hold. We conclude using, the logarithm series, that $\Pr[\bigwedge_{i,s} \neg C_{s,i}] \geq$

$$\geq \prod_{s=2}^{S} (1-x_s)^{c_s} \geq \exp\left(-\sum_{s=2}^{S} (d\gamma)^s/s\right) + o(1) > (1-d\gamma) \cdot \exp(d\gamma).$$

This is a constant > 0 independent of S and the proof is finished.

Proof sketch of Lemma 18 (b). Let $x, y \in B$. The expectation of the edge $(x, y) \notin (B_1, B_0)$ is $(3/4)(1/n)$. For $(x, y) \in (B_1, B_0)$ we get

$$\frac{3}{4} \cdot \frac{1}{n} \cdot \left(1 + \frac{1}{1 - (1 - c/n)^{|B_0|}}\right) = \frac{3}{4} \cdot \frac{1}{n} \cdot d \cdot (1 + o(1)),$$

where $d := 1 + (1/(1 - \exp(-c\beta_0)))$. We have $(3/4)d \approx 3.3255$ which seems large as we need constants < 1. But (B_1, B_0) has only $\approx 0.1n^2$ many candidate edges. This is important.

The number of candidate cycles through B of length s and with exactly $k \leq s/2$ edges from (B_1, B_0) is

$$\leq \binom{s}{k} \cdot 1/s \cdot (\beta_1 n \beta_0 n)^k \cdot (\beta n)^{s-2k}. \tag{3}$$

Its probability is $((1/n)(3/4))^s \cdot d^k$, recall from (a) that edges of a cycle are independent. Summing over k the expected number of cylces of length s is

$$\leq \frac{1}{s} \left(\frac{3}{4}\right)^s \cdot \left(\beta + \frac{\beta_1 \beta_0 d}{\beta}\right)^s.$$

We show that the base under the exponent s is < 1. Then we can finish analogously to (a) using the Local Lemma with parameters s and k, based on (3) and the subsequent probability. Recalling the definitions of the β's we get

$$\beta + \frac{\beta_1 \beta_0}{\beta} \cdot d = \beta_0 \cdot (2 - \exp(-c\beta_0) + \exp(c\beta_0)).$$

We bound (by calculator) $\beta_0 \leq 0.48$, then $c \cdot \beta_0 \leq 0.36$ and
$\exp(c\beta_0) \leq 144/100$ and $\exp(-c\beta_0) \geq 100/144$. Using these bounds we get

$$\beta_0 \cdot (2 - \exp(-c\beta_0) + \exp(c\beta_0)) < 4/3$$

finishing the proof sketch.

5 Conclusion

The main result of this paper, besides the threshold, is that cyclic ordering constraints are satisfiable beyond the 2-core threshold for 3-uniform hypergraphs. We did not really strive for an optimal constant and possibly our constant 1 in Theorem 3 can be improved based on the techniques presented. The proof of this Theorem is constructive in that it implies a polynomial time algorithm (perform the reductions from Figure 1 and sort topologically) working with probability not going to 0.

Acknowledgement. The author thanks André Lanka for helpful hints and thoughtful questions and Anja Lau for experimental work and proof reading.

References

1. Ailon, N., Alon, N.: Hardness of fully dense problems. Information and Computation 205(8), 1117–1129 (2007)
2. Alon, N., Spencer, J.H.: The Probabilistic Method. Wiley-Interscience Publications, Hoboken (1992)
3. Beame, P., Karp, R., Pitassi, T., Saks, M.: The efficiency of resolution and Davis-Putnam procedures. SIAM Journal on Computing 31(4), 1048–1075 (2002)
4. Bodirsky, M., Kára, J.: The complexity of temporal constraint satisfaction problems. In: Proceedings of SToC 2008, pp. 29–38. ACM, New York (2008)
5. Bollobas, B.: Random Graphs. Academic Press, London (1985)
6. Christof, T., Jünger, M., Kececioglu, J., Mutzel, P., Reinelt, G.: A branch-and-cut approach to physical mapping with end-probes
7. Chor, B., Sudan, M.: A geometric approach to betweenness. SIAM Journal on Disctete Mathematics 11(4), 511–523 (1998)
8. Dubois, O., Boufkhad, Y., Mandler, J.: Typical random 3-SAT formulae and the satisfiability threshold. In: Proceedings 11th SODA, pp. 126–127 (2000)
9. Erdös, P., Renyi, A.: On the evolution of random graphs. Publications of the Mathematical Institute of the Hungarian Academy of Science 5, 17–61 (1960)
10. Friedgut, E.: Necessary and sufficient conditions for sharp thresholds of graph properties and the k-SAT problem. Journal of the American Mathematical Society 12, 1017–1054 (1999)
11. Friedgut, E.: Hunting for sharp thresholds. Random Structures and Algorithms 26(1-2), 37–51 (2005)
12. Friedgut, E., Krivelevich, M.: Sharp Thresholds for Ramsey Properties of Random Graphs. Random Structures and Algorithms 17(1), 1–19 (2000)
13. Galil, Z., Meggido, N.: Cyclic ordering is NP-complete. Theoretical Computer Science 5(2), 179–182 (1977)
14. Guttmann, W., Maucher, M.: Variations on an ordering theme with constraints. In: Proceedings Fourth IFIP International Conference on Theoretical Computer Science, TCS 2006. IFIP, vol. 209, pp. 77–90. Springer, Heidelberg (2006)
15. Isli, A., Cohen, A.G.: An Algebra for cyclic ordering of 2D orientation. In: Proceedings of the 15th American Conference on Artificial Intelligence (AAAI), pp. 643–649. MIT Press, Cambridge (1998)
16. Hatami, H., Molloy, M.: Sharp Thresholds for Constraint Satisfaction Problems and Homomorphisms. Random Structures and Algorithms 33(3), 310–332 (2008)
17. Karp, R.M.: The transitive closure of a random digraph. Random Structures and Algorithms 1(1), 73–93
18. Molloy, M.: Models and thresholds for random constraint satisfaction problems. In: Proceedings SToC 2002, pp. 209–217. ACM, New York (2002)
19. Molloy, M.: When does the giant component bring unsatisfiability? Combinatorica 28, 693–735 (2008)
20. Molloy, M.: The pure literal rule threshold and cores in random hypergraphs. In: Proceedings SoDA 2004, pp. 665–674 (2004)
21. Opatrny, J.: Total ordering problem. SIAM Journal on Computing 8(1), 111–114 (1979)
22. Palasti, I.: On the threshold distribution function of cycles in a directed random graph. Studia Scientiarum Mathematicarum Hungarica 6, 67–73 (1971)

Depth Reduction for Circuits with a Single Layer of Modular Counting Gates

Kristoffer Arnsfelt Hansen

Department of Computer Science
Aarhus University*
arnsfelt@cs.au.dk

Abstract. We consider the class of constant depth AND/OR circuits augmented with a layer of modular counting gates at the bottom layer, i.e $\mathbf{AC^0} \circ \mathbf{MOD}_m$ circuits. We show that the following holds for several types of gates $\mathcal{G}$: by adding a gate of type $\mathcal{G}$ at the output, it is possible to obtain an equivalent probabilistic depth 2 circuit of quasipolynomial size consisting of a gate of type $\mathcal{G}$ at the output and a layer of modular counting gates, i.e $\mathcal{G} \circ \mathbf{MOD}_m$ circuits. The types of gates $\mathcal{G}$ we consider are modular counting gates and threshold-style gates. For all of these, strong lower bounds are known for (deterministic) $\mathcal{G} \circ \mathbf{MOD}_m$ circuits.

1 Introduction

A long standing problem in Boolean circuit complexity is to understand the computational power of constant depth AND/OR circuits augmented with modular counting (MOD_m) gates, i.e $\mathbf{ACC^0}$ circuits. One approach would be to consider restrictions on the occurrences of the MOD_m gates. Restricting circuit to contain MOD_m gates only at the layer below the output or to only contain *few* MOD_m gates have successfully resulted in lower bounds [10,23,14,9]. We believe that proving lower bounds for $\mathbf{ACC^0}$ circuits containing MOD_m only in a single layer would be an important next step towards understanding $\mathbf{ACC^0}$ circuits. The restriction we will study in this work is even stricter: we require that all MOD_m gates occur at the bottom layer. This still gives a class of circuits for which no strong lower bounds are known. In fact, no good lower bounds are known for depth 3 $\mathbf{ACC^0}$ circuits and this is true even when the MOD_m gates can occur only at the bottom layer.

More precisely, while strong lower bounds *are* known for $\mathbf{AND} \circ \mathbf{OR} \circ \mathbf{MOD}_m$ circuits, no strong lower bounds are known for $\mathbf{OR} \circ \mathbf{AND} \circ \mathbf{MOD}_m$ circuits. We remark that for these statements the precise definition of MOD_m gates is crucial[1]. Grolmusz proved that $\mathbf{MAJ} \circ \mathbf{OR} \circ \mathbf{MOD}_m$ circuits require size $2^{\Omega(n)}$

* Supported by a postdoc fellowship from the Carlsberg Foundation. Part of this research was done at The University of Chicago supported by a Villum Kann Rasmussen postdoc fellowship.

[1] Two definitions are commonly used in the literature, one being the complement of the other. This also means that the lower bounds we review below are stated differently than their original statement.

to compute the *inner product modulo 2* function IP_2 [12]. For the same class of circuits, Beigel and Maciel [4] proved that MOD_q requires size $2^{\Omega(n)}$, when $q \nmid m$, and that IP_p requires size $2^{\Omega(\sqrt{n})}$. They also managed to show that $\mathbf{MAJ} \circ \mathbf{AND} \circ \mathbf{MOD}_{p^k}$ circuits require size $2^{\Omega(n)}$ to compute the MOD_q function, but only when p is a prime not dividing q. Also, Jukna uses *graph complexity* [16] to derive lower bounds for $\mathbf{AND} \circ \mathbf{OR} \circ \mathbf{MOD}_2$ circuits; this lower bound is easily extended to $\mathbf{AND} \circ \mathbf{OR} \circ \mathbf{MOD}_m$ circuits.

One of the strongest lower bounds obtained in Boolean circuit complexity is the lower bound for $\mathbf{AC^0}[p^k]$ circuits by Razborov [18] and Smolensky [19]. This result is proved in two steps. First a *depth reduction* is invoked, resulting in *probabilistic* $\mathbf{MOD}_p \circ \mathbf{AND}_{\log^{O(1)} n}$ circuits. Then a lower bound for these are derived from counting arguments. Depth reduction results for the entire class $\mathbf{ACC^0}$ obtained by Yao [24] and Beigel and Tarui [6] gave hope that a similar two step approach could be used to obtain lower bounds for $\mathbf{ACC^0}$. Indeed by results of Håstad and Goldmann [15] it is then sufficient to obtain strong lower bounds for multiparty communication complexity for $\log^{O(1)} n$ players in the "number on the forehead" model, but such a result currently seems out of reach.

We believe that it should be explored if a two step approach using depth reduction can be employed for subclasses of $\mathbf{ACC^0}$. Indeed, the depth reduction by Beigel and Tarui results in a class that is arguably too powerful. They show that any $\mathbf{ACC^0}$ circuit is simulated by a *deterministic* $\mathbf{SYM} \circ \mathbf{AND}_{\log^{O(1)}}$ circuit. Beigel, Tarui and Toda proved that this latter class of circuits can even simulate *probabilistic* $\mathbf{EMAJ} \circ \mathbf{ACC^0}$ circuits [7].

In this paper we derive a number of depth reduction results for $\mathbf{AC^0} \circ \mathbf{MOD}_m$ circuits. Let $\mathcal{G}$ denote a class of modular counting gates (modulo a prime p), exact threshold gates, majority gates or threshold gates, i.e MOD_p, ETHR, MAJ or THR gates. Then by adding a gate of type $\mathcal{G}$ at the output of the $\mathbf{AC^0} \circ \mathbf{MOD}_m$ circuit allows one to obtain a depth reduction to probabilistic $\mathcal{G} \circ \mathbf{MOD}_m$ circuits.

For each of these classes strong lower bounds are known for *deterministic* circuits. For $\mathbf{MOD}_p \circ \mathbf{MOD}_m$ circuits lower bounds was obtained by Barrington, Straubing and Thérien [3] (See also [2,20,13,17]). Lower bounds for $\mathbf{MAJ} \circ \mathbf{MOD}_m$ circuits was obtained by Goldmann [11] and finally lower bounds for $\mathbf{THR} \circ \mathbf{MOD}_m$ circuits was obtained by Krause and Pudlák [17].

Our depth reduction reduction proof will use two ingredients. First, as previous results of this kind we will use constructions of probabilistic polynomials. Secondly we will use representations of Boolean functions as Fourier sums. We will present these in Section 2 and Section 3, respectively. Finally in Section 4 we combine these to obtain our main results. In the remainder of this section we briefly review the necessary circuit definitions.

1.1 Constant Depth Circuits

We consider circuits built from families of unbounded fanin gates. Inputs are allowed to be Boolean variables and their negations as well as the constants 0 and 1. In addition to AND, OR and NOT we consider MOD_m gates and threshold style gates. Let $x_1, \ldots, x_n$ be n Boolean inputs. For a positive integer m, let

MOD_m be the function that outputs 1 if and only if $\sum_{i=1}^n x_i \not\equiv 0 \pmod m$. The majority function, MAJ, is 1 if and only if $\sum_{i=1}^n x_i \geq \frac{n}{2}$. Similarly, the exact majority function, EMAJ, is 1 if and only if $\sum_{i=1}^n x_i = \frac{n}{2}$. Let $w \in \mathbf{R}^n$ and let t be any real number. The threshold function with weights w and threshold t, $\text{THR}_{w,t}$ is 1 if and only if $\sum_{i=1}^n w_i x_i \geq t$. Similarly, the exact threshold function with weights w and threshold t, $\text{ETHR}_{w,t}$ is 1 if and only if $\sum_{i=1}^n w_i x_i = t$.

Let **AND** and **OR** denote the families of unbounded fanin AND and OR gates. Let $\mathbf{MOD}_m$, **EMAJ**, **MAJ**, **ETHR**, **THR** denote the families of MOD_m, EMAJ, MAJ, $\text{ETHR}_{w,t}$ and $\text{THR}_{w,t}$ gates, for arbitrary w and t. If $\mathcal{G}$ is a family of Boolean gates and $\mathcal{C}$ is a family of circuits we let $\mathcal{G} \circ \mathcal{C}$ denote the class of circuits consisting of a $\mathcal{G}$ gate taking circuits from $\mathcal{C}$ as inputs.

By the *size* of a circuit we mean the number of *wires* in the circuit. As is usual we will always have a family of circuits in mind, containing a circuit for each input length. In this way the size becomes a function of the input length.

$\mathbf{AC^0}$ is the class of functions computed by polynomial size constant depth circuits built from AND, OR and NOT gates. $\mathbf{AC^0}[m]$ is the class of functions computed by polynomial size constant depth circuits built from AND, OR and MOD_m gates. $\mathbf{ACC^0}$ is the union of the classes $\mathbf{AC^0}[m]$. We will also use the the terms $\mathbf{AC^0}$, $\mathbf{AC^0}[m]$ and $\mathbf{ACC^0}$ in general to denote families of circuits whose size is not bounded by a polynomial; in such cases we will always specify a specific size bound.

We will also consider families of *probabilistic* Boolean circuits. For our purposes we simply define a probabilistic circuit to be a family containing for each input length a distribution over Boolean circuits of that input length. Let f be a Boolean function and let C be a probabilistic circuitc. We say that C computes f with error ϵ if for every $x \in \{0,1\}^n$ we have $\Pr[C(x) \neq f(x)] \leq \epsilon$. We say that C computes f with one-sided positive error ϵ, if C computes f with error ϵ and whenever $f(x) = 0$ we have $\Pr[C(x) = 0] = 1$. Similarly we say that C computes f with one-sided negative error ϵ, if C computes f with error ϵ and whenever $f(x) = 1$ we have $\Pr[C(x) = 1] = 1$.

2 Probabilistic Polynomials

Like the case of circuits we simply define probabilistic polynomials to be distributions over polynomials. We can then define when a polynomial compute a Boolean function with error, one-sided positive error and one-sided negative error completely analogously. As a further notion, when P is an integer polynomial we say that P computes f with zero-sided error ϵ if P computes f with error ϵ and for all x we have $\Pr[P(x) \in \{0,1\} \wedge P(x) \neq f(x)] = 0$. Note that if P computes f with zero-sided error, then by considering $P(x)(2P(x) - 1)$ and $P(x)(3 - 2P(x))$ we get probabilistic polynomials P_1 and P_2 that compute f with zero-sided error and satisfies $P_1(x) \geq 0$ and $P_2(x) \leq 1$ for all x.

Razborov [18] and Smolensky [19] (cf. [1]) gave a simple construction of probabilistic polynomials over $\mathbf{Z}_p$ computing the OR function.

Theorem 1 (Razborov, Smolensky). *For any prime p and any $\epsilon > 0$ there is a probabilistic polynomial over $\mathbf{Z}_p$ of degree $O(\log(\frac{1}{\epsilon}))$ that compute the* OR *function with one-sided positive error at most ϵ.*

This also gives a probabilistic polynomial that compute the AND function with one-sided negative error.

Fermat's little theorem gives a polynomial over $\mathbf{Z}_p$ of constant degree $p-1$, computing the MOD_p function and the following extension gives the same for the MOD_{p^k} function (see e.g [6] for a proof).

Lemma 2. *Let $q = p^k$ for a prime p. Then the MOD_q function is computed by polynomial over $\mathbf{Z}_p$ of degree $q-1$.*

Combining Theorem 1 and Lemma 2 and composing polynomials then gives the following.

Theorem 3 (Razborov, Smolensky). *Let $q = p^k$ for a prime p. Let C be a depth h $\mathbf{AC^0}[q]$ circuit of size S and let $\epsilon > 0$. Then there is a family of probabilistic polynomials of degree $O(\log(\frac{S}{\epsilon})^h)$ that compute the output of C with error at most ϵ.*

Based on a theorem by Valiant and Vazirani [22], Beigel et al. [5] and Tarui [21] gave probabilistic polynomials over the integers computing the OR function, thereby generalizing Theorem 1, albeit at the expense of a slightly larger degree. As with Theorem 1 it also gives probabilistic polynomials computing the AND function.

Theorem 4 (Beigel et al., Tarui). *For any $\epsilon > 0$ there is a family of probabilistic polynomials over $\mathbf{Z}$ of degree $O(\log(\frac{1}{\epsilon})\log n)$ and having coefficients of absolute value $2^{O(\log(\frac{1}{\epsilon})\log(n))}$ that compute the* OR *function with one-sided positive error at most ϵ.*

Let $P(x)$ denote a polynomial from this family. Tarui[2] considered the family of polynomials given by $Q(x) = 1-(x_1+\cdots+x_n+1)(P(x)-1)^2$ he obtained a family of polynomials computing the OR function with zero-sided error. With these polynomials Tarui obtained probabilistic polynomials computing the output of $\mathbf{AC^0}$ circuits with zero-sided error as well. Beigel et al. subsequently gave a simpler construction for obtaining this, that we will review next.

Theorem 5 (Tarui). *Let C be a depth h $\mathbf{AC^0}$ circuit of size S and let $\epsilon > 0$. Then there is a family of probabilistic polynomials over $\mathbf{Z}$ having degree $O((\log(\frac{S}{\epsilon})\log(S))^h)$ and coefficients of absolute value $2^{O((\log(\frac{S}{\epsilon})\log(S))^h)}$ that compute the output of C with zero-sided error at most ϵ.*

Proof. **(Beigel et al.)** By composing the polynomials given by Theorem 4 we get a family of polynomials of degree $O((\log(\frac{S}{\epsilon})\log(S))^h)$ and having coefficients

[2] Tarui actually stated his results in terms of the NOR function making the polynomials slightly different.

of absolute value at most $2^{O((\log(\frac{S}{\epsilon})\log(S))^h)}$ that probabilistically compute the output of C with error at most ϵ. Let F denote a member of this family. Let g be any gate of C taking inputs $g_1,\ldots,g_m$. Let P_g denote a member of the family of polynomials computing g in variables $y_1,\ldots,y_m$. If g is an OR gate define E_g by $E_g(y)=(y_1+\cdots+y_m)(P(y)-1)$. We then have that $E_g(y)=0$ if and only if $P_g(y)=\mathrm{OR}(y)$. When g is an AND gate then similarly we define $E_g(y)=(y_1+\cdots+y_n-n)P(y)$ and we have that $E_g(y)=0$ if and only if $P_g(y)=\mathrm{AND}(y)$.

Now, define $E(x)=\sum_{g\in C}(E_g(x))^2$. Then $E(x)=0$ whenever all gates in C are computed correctly. Then finally we have that the family of polynomials given

$$G(x)=F(x)-((F(x))^2+1)E(x)$$

compute the output of C with zero-sided error at most ϵ.

Clearly these polynomials are of degree $O((\log(\frac{S}{\epsilon})\log(S))^h)$ and have coefficients of absolute value $2^{O((\log(\frac{S}{\epsilon})\log(S))^h)}$ as well.

3 Fourier Sum Representation

In this section we will derive representations of circuits of the form $\mathcal{G}\circ\mathbf{AND}_d\circ\mathbf{MOD}_m$ for several choices of a family of Boolean gates $\mathcal{G}$ by Fourier sums over a field with an mth root of unity. Conversely we will derive $\mathcal{G}\circ\mathbf{MOD}_m$ circuits computing the Boolean functions represented by such representations. Combining these two types of results then implies that the layer of $\mathbf{AND}_d$ gates can be eliminated.

When $\mathcal{G}$ is a family of modular counting gates the appropriate setting will be Fourier sums over a finite field. When $\mathcal{G}$ is a family of threshold style gates the appropriate setting will instead be Fourier sums over the field of complex numbers.

3.1 Modular Counting Gates

Representations of $\mathbf{MOD}_p\circ\mathbf{AND}_d\circ\mathbf{MOD}_m$ circuits by Fourier sums over a finite field was introduced in the work of Barrington, Straubing and Thérien [3] and is made entirely explicit by Barrington and Straubing [2] and further results were obtained by Straubing and Thérien [20]. All these works actually consider depth $d+1$ $(\mathbf{MOD}_{p^k})^d\circ\mathbf{MOD}_m$ circuits, which are converted into $\mathbf{MOD}_p\circ\mathbf{AND}_{O(d)}\circ\mathbf{MOD}_m$ circuits as the first step in constructing the representation. We will next review these results.

Let m be a positive integer and let p be a prime that does not divide m. Choose k such that m divides p^k-1. Then the finite field $F=\mathrm{GF}(p^k)$ contains an mth root of unity ω. We will consider expressions in variables $x_1,\ldots,x_n$ of the form

$$\sum_{i=1}^{S}c_i\omega^{a_{i,1}x_1+\ldots a_{i,n}x_n}\ ,$$

where c_i are elements of F and $a_{i,j}$ are elements of $\mathbf{Z}_m$. We will call that a Fourier sum over F of size S. We say such an expression $E(x)$ computes a Boolean function f if $E(x) = f(x)$ for all $x \in \{0,1\}^n$ and we say $E(x)$ represents a Boolean function f if $E(x) \in \mathbf{Z}_p$ for all x and moreover $E(x) = 0$ if and only if $f(x) = 0$ for all $x \in \{0,1\}^n$.

When $x_1, \ldots, x_n$ are variables from $\mathbf{Z}_m$ then these expressions can in fact be viewed as Fourier transforms of functions $f : (\mathbf{Z}_m)^n \to F$, thereby justifying our terminology. For details about this we refer to the works of Barrington et al. [3,2,20]. We have the following.

Lemma 6. *A* MOD_m *gate can be computed by a Fourier sum of size* $2^{|F|-1}$*.*

Proof. A MOD_m gate with inputs $x_1, \ldots, x_n$ can be computed by the expression

$$(\omega^{x_1+\cdots+x_n} - 1)^{|F|-1}$$

since

$$(\omega^a - 1)^{|F|-1} = \begin{cases} 0 & \text{if} \quad a \equiv 0 \pmod{m} \\ 1 & \text{if} \quad a \not\equiv 0 \pmod{m} \end{cases}$$

thereby giving a Fourier sum of size $2^{|F|-1}$.

Then taking sums of these expressions shows that a $\mathbf{MOD}_p \circ \mathbf{MOD}_m$ circuit of size S can be computed by a Fourier sum of size at most $S2^{(|F|-1)(p-1)}$. But at the expense of increasing the size of the circuit we can even introduce small fanin AND gates as a middle layer.

Proposition 7 (Barrington et al.). *Let p be a prime not dividing m. For any* $\mathbf{MOD}_p \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ *circuit of size S there is a Fourier sum representing the output of the circuit of size at most* $S2^{d(|F|-1)}$*.*

Proof. We interpret the top two layers of the circuit as a polynomial over $\mathbf{Z}_p$ in S variables with at most S terms and of degree d. Express each MOD_m gate of the circuit as a Fourier sum of size $2^{|F|-1}$. Substituting these for the variables in the polynomial and expanding then yields the required Fourier sum representing the output of the circuit of size at most $S2^{d(|F|-1)}$.

Proposition 8 (Straubing and Thérien). *Suppose a Boolean function f can be represented by a Fourier sum of size S. Then f is computed by a* $\mathbf{MOD}_p \circ \mathbf{MOD}_m$ *circuit of size $m(p-1)S$.*

Proof. The field F is a vector space over $\mathbf{Z}_p$. We can thus pick a basis $v_1, \ldots, v_k$ of F where we can choose $v_1 = 1$. Let $\pi_1 : F \to \mathbf{Z}_p$ be the projection of an element of F onto the first coordinate in the basis $v_1, \ldots, v_k$. By linearity we have

$$\pi_1(\sum_{i=1}^{S} \omega^{a_{i,1}x_1 + \ldots a_{i,n}x_n}) \equiv \sum_{i=1}^{S} \pi_1(\omega^{a_{i,1}x_1 + \ldots a_{i,n}x_n}) \pmod{p} \ .$$

Thus to compute the sum we can compute each term $\pi_1(\omega^{a_{i,1}x_1+\dots a_{i,n}x_n})$ individually. For every $0 \le a < m$ we will have $(\pi_1(\omega^a)(p-1) \mod p)$ copies of a MOD_m gate that evaluate to 1 if $a_{i,1}x_1 + \dots + a_{i,n}x_n \not\equiv a \pmod{m}$. Furthermore we feed $\pi_1(\omega^a)$ copies of the constant 1. The sum of these will be $\pi_1(\omega^a)$ when the term has value $\pi_1(\omega^a)$ and will be 0 otherwise. Thus taking the sum for every a gives $m(p-1)$ MOD_m gates that compute the given term.

Combining Proposition 7 and Proposition 8 we obtain the following somewhat surprising result, showing that a middle layer of small fanin AND gates can be absorbed at the cost of a reasonable increase of the size of the circuit.

Theorem 9 (Straubing and Thérien). *Let p be a prime not dividing m. Then any function computed by a $\mathrm{MOD}_p \circ \mathrm{AND}_d \circ \mathrm{MOD}_m$ circuit of size S is also computed by a $\mathrm{MOD}_p \circ \mathrm{MOD}_m$ circuit of size $S2^{O(d)}$.*

3.2 Threshold Style Gates

It was suggested by Barrington and Straubing [2] to use Fourier representations over the complex numbers to study $\mathbf{THR} \circ \mathbf{MOD}_m$ circuits. The case of $m = 2$ is known as polynomial threshold functions [8] and these circuits are precisely representations by the sign of a Fourier sum. We will derive representations for $\mathcal{G} \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ circuits when $\mathcal{G}$ is a family of threshold, exact threshold or majority gates.

Let m be a positive integer and let $\omega = e^{\frac{2\pi i}{m}}$ be an mth root of unity. Similarly to the previous section we consider expressions in variables $x_1, \dots, x_n$ of the form

$$\sum_{i=1}^{S} c_i \omega^{a_{i,1}x_1+\dots a_{i,n}x_n} \quad ,$$

where c_i complex numbers and $a_{i,j}$ are elements of $\mathbf{Z}_m$. We will call that a Fourier sum over $\mathbf{C}$ of size S and we will call the numbers c_i the coefficients. We say such an expression $E(x)$ computes a Boolean f if $E(x) = f(x)$ for all $x \in \{0,1\}^n$. We say that $E(x)$ sign represents a Boolean function f if $E(x) \in \mathbf{R} \setminus \{0\}$ for all $x \in \{0,1\}^n$ and moreover $E(x) > 0$ if and only if $f(x) = 1$ for all $x \in \{0,1\}^n$. Finally we say that $E(x)$ equality represents a Boolean function f if $E(x) \in \mathbf{R}$ for all $x \in \{0,1\}^n$ and moreover $E(x) = 0$ if and only $f(x) = 1$ for all $x \in \{0,1\}^n$.

As the previous case of finite fields, when $x_1, \dots, x_n$ are variables from $\mathbf{Z}_m$ then these expressions can be viewed as Fourier transforms of functions $f : (\mathbf{Z}_m)^n \to \mathbf{C}$.

Lemma 10. *A MOD_m gate can be computed by a Fourier sum of size $m+1$ where the coefficients are either 1 or $\frac{1}{m}$.*

Proof. A MOD_m gate with inputs $x_1, \dots, x_n$ can be computed by the expression

$$1 - \frac{1}{m} \sum_{b=0}^{m-1} \omega^{b(x_1+\dots+x_n)}$$

since

$$\sum_{b=0}^{m-1} \omega^{ba} = \begin{cases} m & \text{if} \quad a \equiv 0 \pmod{m} \\ 0 & \text{if} \quad a \not\equiv 0 \pmod{m} \end{cases} .$$

thereby giving a Fourier sum of size $m+1$.

With this we can now derive Fourier sum representations of different classes of circuits. First we consider circuits with a threshold gate at the output.

Proposition 11. *For any* $\mathbf{THR} \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ *circuit of size* S *there is a Fourier sum of size at most* $S(m+1)^d + 1$ *sign representing the output of the circuit.*

Proof. We will assume that the threshold value of the output gate is 0. This can then afterward be corrected by increasing the size of the obtained Fourier sum by 1. We interpret the top two layers of the circuit as a polynomial over $\mathbf{R}$ in S variables with at most S terms and of degree d. Express each MOD_m gate of the circuit as a Fourier sum of size $m+1$. Substituting these for the variables in the polynomial yields the required Fourier sum sign representing the output of the circuit of size at most $S(m+1)^d$.

With the same proof but switching to equality representation we obtain the same with an exact threshold gate at the output.

Proposition 12. *For any* $\mathbf{ETHR} \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ *circuit of size* S *there is a Fourier sum of size at most* $S(m+1)^d + 1$ *equality representing the output of the circuit.*

With a majority gate at the output, we observe that the proof of Proposition 11 gives a Fourier sum where all coefficients are of the form $\frac{1}{m^i}$ for $i \in \{0, \ldots, d\}$, by Lemma 10. Then since all coefficients of the polynomial given by the top two layers are 1, multiplying by m^d yields a Fourier sign representation as stated below.

Proposition 13. *For any* $\mathbf{MAJ} \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ *circuit of size* S *there is a Fourier sum with integer coefficients of total absolute value at most* $S((m+1)^d m^d + 1)$ *sign representing the output of the circuit.*

Proposition 14. *Suppose a Boolean function* f *can be sign represented by a Fourier sum over* $\mathbf{C}$ *of size* S*. Then* f *is computed by a* $\mathbf{THR} \circ \mathbf{MOD}_m$ *circuit of size* mS*.*

Proof. By linearity we have

$$\text{Re}(\sum_{i=1}^{S} c_i \omega^{a_{i,1}x_1 + \ldots a_{i,n}x_n}) = \sum_{i=1}^{S} \text{Re}(c_i \omega^{a_{i,1}x_1 + \ldots a_{i,n}x_n}) .$$

Thus to compute the sum we can compute each term $\text{Re}(c_i \omega^{a_{i,1}x_1 + \ldots a_{i,n}x_n})$ individually. For every $0 \leq a < m$ we will have a MOD_m gate that evaluate to 1 if $a_{i,1}x_1 + \ldots a_{i,n}x_n \not\equiv a \pmod{m}$. This MOD_m gate is given the coefficient $-\text{Re}(c_i \omega^a)$ and we add $\text{Re}(c_i \omega^a)$ to the threshold value of the output gate, which effectively makes the MOD_m gate contribute the correct value to the sum.

With the same proof we obtain a similar result for equality representation.

Proposition 15. *Suppose a Boolean function f can be equality represented by a Fourier sum over* $\mathbf{C}$ *of size S. Then f is computed by a* $\mathbf{ETHR} \circ \mathbf{MOD}_m$ *circuit of size mS.*

To be able to compute sign representations with bounded integer coefficients we will need a slightly more involved approach, since we will only be able to compute the sum with limited precision.

We consider the cyclotomic field $\mathbf{Q}(\omega)$. Let $\omega_1, \ldots, \omega_{\phi(m)}$ be the conjugates of ω, where ϕ is Euler's totient function. Let $z = g(\omega)$ where $g \in \mathbf{Q}[X]$. The *norm* $N(z)$ is then given by $N(z) = \prod_{i=1}^{\phi(m)} g(\omega_i)$. It is well known that the norm has the property that $N(z) \in Q$ and $N(z) = 0$ if and only if $z = 0$. Furthermore, when $g \in \mathbf{Z}[X]$ we have that $N(z) \in \mathbf{Z}$.

Proposition 16. *Let $z \in \mathbf{Q}(\omega)$ be nonzero and assume $z = g(\omega)$, where $g(X) \in \mathbf{Z}[X]$ have integer coefficients of total absolute value at most M. Then we have*

$$|z| \geq \frac{1}{M^{\phi(m)-1}} \ .$$

Proof. Since $g(X) \in \mathbf{Z}[X]$ we have that $N(z) \in \mathbf{Z}$. Furthermore since the coefficients of g are of total absolute value at most M we have $|g(\omega_i)| \leq M$ for all i. Thus we have

$$1 \leq |N(z)| \leq \left| \prod_{i=1}^{\phi(m)} g(\omega_i) \right| = \prod_{i=1}^{\phi(m)} |g(\omega_i)| \leq |g(\omega)| M^{\phi(m)-1}$$

from which the result follows.

Corollary 17. *Let $z \in \mathbf{Q}(\omega)$ be such that $\mathrm{Re}(z) \neq 0$ and assume $z = g(\omega)$, where $g(X) \in \mathbf{Z}[X]$ have integer coefficients of total absolute value at most M. Then we have*

$$|\mathrm{Re}(z)| \geq \frac{1}{2(2M)^{\phi(m)-1}} \ .$$

Proof. Since $\mathrm{Re}(z) = \frac{1}{2}(z + \bar{z})$ Proposition 16 gives $|z + \bar{z}| \geq \frac{1}{(2M)^{\phi(m)-1}}$ from which the result follows.

Proposition 18. *Suppose a Boolean function f can be sign represented by a Fourier sum over* $\mathbf{C}$ *of size S with integer coefficients of absolute value at most M. Then f is computed by a* $\mathbf{MAJ} \circ \mathbf{MOD}_m$ *circuit of size $4mS(2M)^{\phi(m)}$.*

Proof. We will construct a $\mathbf{THR} \circ \mathbf{MOD}_m$ circuit and carefully track the size of the integer coefficients. Following the proof of Proposition 14 we derive

$$\mathrm{Re}(\sum_{i=1}^{S} c_i \omega^{a_{i,1}x_1 + \ldots a_{i,n}x_n}) = \sum_{i=1}^{S} \mathrm{Re}(c_i \omega^{a_{i,1}x_1 + \ldots a_{i,n}x_n}) \ .$$

Now from Corollary 17 the absolute value of the left-hand side is at least $\frac{1}{2(2M)^{\phi(m)-1}}$. We will approximate each term $\mathrm{Re}(c_i\omega^{a_{i,1}x_1+\dots a_{i,n}x_n})$ individually. Let x be arbitrary and define $a_i = a_{i,1}x_1 + \dots a_{i,n}x_n$. For every $0 \le a < m$ define

$$\hat{c}_{i,a} = \lfloor 4S(2M)^{\phi(m)-1}\,\mathrm{Re}(c_i\omega^a)\rfloor \ .$$

We then have that

$$\left| 4S(2M)^{\phi(m)-1}\,\mathrm{Re}(\sum_{i=1}^{S} c_i\omega^{a_i}) - \sum_{i=1}^{S}\hat{c}_{i,a}\right| \le S \ .$$

Since we also have that

$$\left| 4S(2M)^{\phi(m)-1}\,\mathrm{Re}(\sum_{i=1}^{S} c_i\omega^{a_i})\right| \ge 2S \ ,$$

the approximation has the correct sign. We can now conclude as in the proof of Proposition 14. For every $0 \le a < m$ we will have a MOD_m gate that evaluate to 1 if $a_{i,1}x_1 + \dots a_{i,n}x_n \not\equiv a \pmod m$. This MOD_m gate is given the coefficient $-\hat{c}_{i,a}$ and we add $\hat{c}_{i,a}$ to the threshold value of the output gate. The total absolute value of the coefficients is bounded by $m4S(2M)^{\phi(m)-1}M$ and the size of the resulting $\mathbf{MAJ} \circ \mathbf{MOD}_m$ circuit is then at most $4mS(2M)^{\phi(m)}$.

As the case of modular counting gates we obtain that a middle layer of AND gates can be absorbed with a reasonable increase in the size of the circuit by combining the results above.

Theorem 19. *Any* $\mathbf{THR} \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ *circuit of size* S *is computed by* $\mathbf{THR} \circ \mathbf{MOD}_m$ *circuit of size* $S2^{O(d)}$. *Any* $\mathbf{ETHR} \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ *circuit of size* S *is computed by* $\mathbf{ETHR} \circ \mathbf{MOD}_m$ *circuit of size* $S2^{O(d)}$. *Any* $\mathbf{MAJ} \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ *circuit of size* S *is computed by* $\mathbf{MAJ} \circ \mathbf{MOD}_m$ *circuit of size* $S^{O(1)}2^{O(d)}$.

4 Depth Reduction for Circuits

In this section we will combine the results about probabilistic polynomials with the Fourier sum representations to derive the stated depth reduction result for circuits with a single layer of MOD_m gates.

Theorem 20. *Let* $\epsilon > 0$. *Any depth* $h+1$ $\mathbf{AC^0}[p] \circ \mathbf{MOD}_m$ *circuit of size* S *is computed by a probabilistic* $\mathbf{MOD}_p \circ \mathbf{MOD}_m$ *circuit of size* $2^{O(\log(S)\log(\frac{S}{\epsilon})^h)}$ *with error at most* ϵ.

Proof. Let C be a depth $h+1$ $\mathbf{AC^0}[p] \circ \mathbf{MOD}_m$ circuit of size S. We first use Theorem 3 on the $\mathbf{AC^0}[p]$ circuit given by the top h layers of C. This gives a probabilistic $\mathbf{MOD}_p \circ \mathbf{AND}_d$ circuit of size S^d, where $d = O(\log(\frac{S}{\epsilon})^h)$, which in turn gives a probabilistic $\mathbf{MOD}_p \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ circuit of size S^d computing C with error at most ϵ. Then Theorem 9 gives a probabilistic $\mathbf{MOD}_p \circ \mathbf{MOD}_m$ circuit of size $S^d 2^{O(d)} = 2^{O(\log(S)d)}$.

Theorem 21. *Let $\epsilon > 0$. Any depth $h+2$ $\mathbf{THR} \circ \mathbf{AC^0} \circ \mathbf{MOD}_m$ circuit of size S is computed by a probabilistic $\mathbf{THR} \circ \mathbf{MOD}_m$ circuit of size $2^{O(\log(S)^{h+1} \log(\frac{S}{\epsilon})^h)}$ with one-sided positive error at most ϵ.*

Proof. Let C be a depth $h+2$ $\mathbf{THR} \circ \mathbf{AC^0} \circ \mathbf{MOD}_m$ circuit of size S, and let $C_1, \ldots, C_S$ be the $\mathbf{AC^0} \circ \mathbf{MOD}_m$ subcircuits that feed the output gate and let $w_1, \ldots, w_s$ be the corresponding weights. We first use Theorem 5 on the top h layers of $C_1, \ldots, C_S$ to give probabilistic integer polynomials $P_1, \ldots, P_S$ of degree $d = O((\log(\frac{S}{\epsilon}) \log(S))^h)$ with zero-sided error ϵ/S. When $w_i \geq 0$ we choose to have $P_i(x) \leq 1$ and when $w_i < 0$ we choose to have $P(x) \geq 0$. In this way we obtain that $\Pr[w_i P_i(x) \leq w_i C_i(x)] = 1$. We then feed all terms of P_i to output gate, with weight given by the product of w_i and the coefficient of the term, for all i. This gives a probabilistic $\mathbf{THR} \circ \mathbf{AND}_d$ circuit for the first $h+1$ layers of C with one-sided positive error ϵ of size S^{d+1}, and thus a probabilistic $\mathbf{THR} \circ \mathbf{AND}_d \circ \mathbf{MOD}_m$ circuit for C. Finally Theorem 19 gives a $\mathbf{THR} \circ \mathbf{MOD}_m$ circuit of size $S^{d+1} 2^{O(d)} = 2^{O(\log(S)d)}$.

With a similar proofs we also obtain.

Theorem 22. *Let $\epsilon > 0$. Then any depth $h+2$ $\mathbf{ETHR} \circ \mathbf{AC^0} \circ \mathbf{MOD}_m$ circuit of size S is computed by a probabilistic $\mathbf{ETHR} \circ \mathbf{MOD}_m$ circuit of size $2^{O(\log(S)^{h+1} \log(\frac{S}{\epsilon})^h)}$ with error at most ϵ. And any depth $h+1$ $\mathbf{AC^0} \circ \mathbf{MOD}_m$ circuit of size S is computed by a probabilistic $\mathbf{ETHR} \circ \mathbf{MOD}_m$ circuit of size $2^{O(\log(S)^{h+1} \log(\frac{S}{\epsilon})^h)}$ with one-sided error at most ϵ.*

Theorem 23. *Let $\epsilon > 0$. Any depth $h+2$ $\mathbf{MAJ} \circ \mathbf{AC^0} \circ \mathbf{MOD}_m$ circuit of size S is computed by a probabilistic $\mathbf{MAJ} \circ \mathbf{MOD}_m$ circuit of size $2^{O(\log(S)^{h+1} \log(\frac{S}{\epsilon})^h)}$ with one-sided positive error at most ϵ.*

Acknowledgments

I thank Laci Babai, Vladimir Trifonov and Gyuri Turán for valuable discussions.

References

1. Allender, E., Hertrampf, U.: Depth reduction for circuits of unbounded fan-in. Information and Computation 112(2), 217–238 (1994)
2. Barrington, D.A.M., Straubing, H.: Lower bounds for modular counting by circuits with modular gates. Computational Complexity 8(3), 258–272 (1999)
3. Barrington, D.A.M., Straubing, H., Thérien, D.: Non-uniform automata over groups. Information and Computation 89(2), 109–132 (1990)
4. Beigel, R., Maciel, A.: Upper and lower bounds for some depth-3 circuit classes. Computational Complexity 6(3), 235–255 (1997)
5. Beigel, R., Reingold, N., Spielman, D.: The perceptron strikes back. In: Proceedings of the 6th Annual Conference on Structure in Complexity Theory, pp. 286–293. IEEE Computer Society Press, Los Alamitos (1991)
6. Beigel, R., Tarui, J.: On ACC. Computational Complexity 4(4), 350–366 (1994)

7. Beigel, R., Tarui, J., Toda, S.: On probabilistic ACC circuits with an exact-threshold output gate. In: Ibaraki, T., Iwama, K., Yamashita, M., Inagaki, Y., Nishizeki, T. (eds.) ISAAC 1992. LNCS, vol. 650, pp. 420–429. Springer, Heidelberg (1992)
8. Bruck, J.: Harmonic analysis of polynomial threshold functions. SIAM Journal on Discrete Mathematics 3(2), 168–177 (1990)
9. Chattopadhyay, A., Goyal, N., Pudlák, P., Thérien, D.: Lower bounds for circuits with MOD_m gates. In: Proceedings og the 47st Annual IEEE Symposium on Foundations of Computer Science, pp. 709–718. IEEE Computer Society, Los Alamitos (2006)
10. Chattopadhyay, A., Hansen, K.A.: Lower bounds for circuits with few modular and symmetric gates. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 994–1005. Springer, Heidelberg (2005)
11. Goldmann, M.: A note on the power of majority gates and modular gates. Information Processing Letters 53(6), 321–327 (1995)
12. Grolmusz, V.: A weight-size trade-off for circuits with MOD_m gates. In: Proceedings of the 26th annual ACM Symposium on the Theory of Computing, pp. 68–74 (1994)
13. Grolmusz, V., Tardos, G.: Lower bounds for $(\text{MOD}_p - \text{MOD}_m)$ circuits. SIAM Journal on Computing 29(4), 1209–1222 (2000)
14. Hansen, K.A.: Lower bounds for circuits with few modular gates using exponential sums. Technical Report 79, Electronic Colloquium on Computational Complexity (2006)
15. Håstad, J., Goldmann, M.: On the power of small-depth threshold circuits. Computational Complexity 1, 113–129 (1991)
16. Jukna, S.: On graph complexity. Combinatorics, Probability & Computing 15(6), 855–876 (2006)
17. Krause, M., Pudlák, P.: On the computational power of depth-2 circuits with threshold and modulo gates. Theoretical Computer Science 174(1-2), 137–156 (1997)
18. Razborov, A.A.: Lower bounds for the size of circuits of bounded depth with basis $(\wedge, \oplus)$. Mathematical Notes of the Academy of Science of the USSR 41(4), 333–338 (1987)
19. Smolensky, R.: Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In: Proceedings of the 19th annual ACM Symposium on Theory of Computing, pp. 77–82 (1987)
20. Straubing, H., Thérien, D.: A note on MOD_p - MOD_m circuits. Theory of Computing Systems 39(5), 699–706 (2006)
21. Tarui, J.: Probabilistic polynomials, $\mathbf{AC}^0$ functions and the polynomial-time hierarchy. Theoretical Computer Science 113(1), 167–183 (1993)
22. Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. Theoretical Computer Science 47(1), 85–93 (1986)
23. Viola, E.: Pseudorandom bits for constant-depth circuits with few arbitrary symmetric gates. SIAM Journal on Computing 36(5), 1387–1403 (2007)
24. Yao, A.C.: On ACC and threshold circuits. In: Proceedings og the 31st Annual IEEE Symposium on Foundations of Computer Science, pp. 619–627. IEEE Computer Society Press, Los Alamitos (1990)

A Feebly Secure Trapdoor Function[⋆]

Edward A. Hirsch and Sergey I. Nikolenko

Steklov Institute of Mathematics at St.Petersburg, Russia

Abstract. In 1992, A. Hiltgen [1] provided the first constructions of provably (slightly) secure cryptographic primitives, namely *feebly one-way functions*. These functions are provably harder to invert than to compute, but the complexity (viewed as circuit complexity over circuits with arbitrary binary gates) is amplified by a constant factor only (with the factor approaching 2).

In traditional cryptography, one-way functions are the basic primitive of private-key and digital signature schemes, while public-key cryptosystems are constructed with trapdoor functions. We continue Hiltgen's work by providing an example of a *feebly trapdoor function* where the adversary is guaranteed to spend more time than every honest participant by a constant factor of $\frac{25}{22}$.

1 Introduction

Numerous public-key cryptographic protocols have been devised over the years, and yet there exists *not a single proof* of their security: neither an unconditional proof (that would necessarily imply P $\neq$ NP) nor a proof based on standard (not problem-specific) structural complexity assumptions. While universal primitives for one-way functions [2,3] and public-key cryptosystems are known [4] (see also [5]), they give no connection to the core assumptions of traditional structural complexity theory; more, the asymptotic nature of polynomial-time reductions leaves no hope for confidence that a particular scheme cannot be broken at a particular key length. In general, it appears like there is still a very long way to go before cryptography can claim any *provably* secure public-key construction.

If we are unable to prove a superpolynomial gap between the complexities of honest parties and adversaries, can we prove at least *some* gap? In 1992, Alain Hiltgen [1] managed to present a function that is *twice* harder to invert than to compute. His example is a linear function over $GF(2)$ with a matrix that has few non-zero entries while the inverse matrix has many non-zero entries; the complexity gap follows by a simple argument of Lamagna and Savage [6,7]: every bit of the output depends non-idly on many variables and all these bits correspond to different functions, hence a lower bound on the complexity of computing them all together. The model of computation here is the most general

⋆ This research was partially supported by the Russian Foundation for Basic Research (RFBR), grants 08-01-00640-a, 08-01-00649, 09-01-00784-a, the Dynasty Foundation, the President of RF grant for leading scientific schools support NSh 4392.2008.1, and the Fundamental research program of the Russian Academy of Sciences.

one: the number of gates in a Boolean circuit that uses arbitrary binary Boolean gates. Note that little more could be expected for this model at present, since known lower bounds here are linear in the number of inputs [8,9].

In this work, we construct another feebly secure cryptographic primitive: namely, a feebly trapdoor function. Of course, in order to obtain the result, we have to prove a lower bound on the circuit complexity of a certain function; we use the gate elimination technique from circuit complexity of the eighties. More formally, the complexity of inverting (decryption) without the use of trapdoor information in our construction is at least $\frac{25}{22}$ times greater than the complexities of honest encryption, decryption, and key generation. We also provide hardness amplification results that give exponential guarantees on the success probability for weaker attackers. In Section 2.1, we give basic definitions. Section 2.2 does preparational work, establishing some combinatorial properties of the matrices representing hard function candidates. Section 2.3 reviews the basic method of gate elimination, which we use to prove lower bounds on complexity, and applies it to the feeble security setting. Finally, Sections 3.1 and 3.2 present the construction of a feebly secure trapdoor function, Section 3.3 proves hardness amplification results, and Section 4 lists possible directions for further research.

2 Preliminaries

2.1 Basic Definitions

Let us denote by $\mathbb{B}_{n,m}$ the set of all 2^{m2^n} functions $f : \mathbb{B}^n \to \mathbb{B}^m$, where $\mathbb{B} = \{0, 1\}$ is the field of two elements. Our computational model is Boolean circuits with arbitrary binary gates. A circuit is a directed acyclic graph with in-degree zero (these nodes are called *circuit inputs*, or *variables*) and two (these nodes are called *gates*). Every gate is labelled by a binary Boolean function (any of the 16 functions in $\mathbb{B}_{2,1}$). Some gates are marked as *outputs* (to avoid extra negation gates, we may also define an output as the negation of the value obtained in a gate; to avoid identity gates, we also allow to define an output as the value of a variable or the negation of the negation of it). A circuit with n inputs and m outputs naturally computes a Boolean function in $\mathbb{B}_{n,m}$. We denote the size of a circuit c by $C(c)$. The *circuit complexity* (or simply *complexity*) of a function f, denoted (slightly abusing notation) by $C(f)$, is the smallest number of gates in a circuit computing f (such circuit is called an *optimal* circuit for f): $C(f) = \min_{c:\forall x\, c(x)=f(x)} C(c)$. We may safely assume that every gate of this circuit depends on both inputs, i.e., there are no constant functions and no unary functions Id and NOT, because such gates can be easily eliminated from the circuit without increasing the number of the gates.

Following Hiltgen, for every injective $f_n \in \mathbb{B}_{n,m}$ we can define its *measure of one-wayness* $M_F(f_n) = \frac{C(f_n^{-1})}{C(f_n)}$. Hiltgen's work was to find sequences of functions $f = \{f_n\}_{n=1}^{\infty}$ with large asymptotic constant $\liminf_{n\to\infty} M_F(f_n)$, which Hiltgen calls f's *order of one-wayness*. We will discuss his results in more detail in Section 2.3. We now extend this definition in order to capture feebly secure trapdoor functions. Since we are interested in constants here, we must pay attention to all the details.

Definition 1. *A* feebly trapdoor candidate *is a sequence of triples of circuits* $\mathcal{C} = \{(\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n)\}_{n=1}^{\infty}$ *where:*

- $\{\mathrm{Key}_n\}_{n=1}^{\infty}$ *is a family of sampling circuits* $\mathrm{Key}_n : \mathbb{B}^n \to \mathbb{B}^{\mathrm{pi}(n)} \times \mathbb{B}^{\mathrm{ti}(n)}$,
- $\{\mathrm{Eval}_n\}_{n=1}^{\infty}$ *is a family of evaluation circuits* $\mathrm{Eval}_n : \mathbb{B}^{\mathrm{pi}(n)} \times \mathbb{B}^{m(n)} \to \mathbb{B}^{c(n)}$, *and*
- $\{\mathrm{Inv}_n\}_{n=1}^{\infty}$ *is a family of inversion circuits* $\mathrm{Inv}_n : \mathbb{B}^{\mathrm{ti}(n)} \times \mathbb{B}^{c(n)} \to \mathbb{B}^{m(n)}$

such that for every security parameter n, every seed $s \in \mathbb{B}^n$, and every input $\mathrm{m} \in \mathbb{B}^{m(n)}$,

$$\mathrm{Inv}_n(\mathrm{Key}_{n,2}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), \mathrm{m})) = \mathrm{m},$$

where $\mathrm{Key}_{n,1}(s)$ *and* $\mathrm{Key}_{n,2}(s)$ *are the first* $\mathrm{pi}(n)$ *bits ("public information") and the last* $\mathrm{ti}(n)$ *bits ("trapdoor information") of* $\mathrm{Key}_n(s)$*, respectively.*

We call this function a "candidate" because the definition does not imply any security, it merely sets up the dimensions and provides correct inversion. In our constructions, $m(n) = c(n)$ and $\mathrm{pi}(n) = \mathrm{ti}(n)$. To find how secure a function is, we introduce the notion of a break. Informally, an adversary should invert the function without knowing the trapdoor information. We introduce break as inversion with probability greater than $\frac{3}{4}$. (We later investigate security against adversaries having smaller success probabilities, too.) We denote by $C_{3/4}(f)$ the minimal size of a circuit that correctly computes a function $f \in \mathbb{B}_{n,m}$ on more than $\frac{3}{4}$ of its inputs (of length n). Obviously, $C_{3/4}(f) \le C(f)$ for all f.

Definition 2. *A circuit N* breaks *a feebly trapdoor candidate* $\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ *on seed length n if for uniformly chosen seeds $s \in \mathbb{B}^n$ and inputs $m \in \mathbb{B}^{m(n)}$*

$$\Pr_{(s,m)\in U}\left[N(\mathrm{Key}_{n,1}(s), \mathrm{Eval}_n(\mathrm{Key}_{n,1}(s), m)) = m\right] > \frac{3}{4}.$$

Remark 1. In fact, in what follows we prove a stronger result: we prove that no circuit (of a certain size) can break our candidate *for any random seed s*, that is, for every seed s, every adversary fails with probability at least $1/4$.

For a trapdoor function to be secure, circuits that break the function should be larger than the circuits computing it.

Definition 3. *A feebly trapdoor candidate* $\mathcal{C} = \{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ *has* order of security k *if for every sequence of circuits $\{N_n\}_{n=1}^{\infty}$ that break f on every input length n,*

$$\liminf_{n\to\infty} \min\left\{\frac{C(N_n)}{C(\mathrm{Key}_n)}, \frac{C(N_n)}{C(\mathrm{Eval}_n)}, \frac{C(N_n)}{C(\mathrm{Inv}_n)}\right\} \ge k.$$

Remark 2. One could consider key generation as a separate process and omit its complexity from the definition of the order of security. However, we prove our results for the definition stated above as it makes them stronger.

Remark 3. Let us note explicitly that we are talking about *one-time* security. An adversary can amortize his circuit complexity on inverting a feebly trapdoor candidate for the second time for the same seed, for example, by computing the trapdoor information and successfully reusing it. Thus, in our setting one has to pick a new seed for every input.

In the next sections, we develop our construction of a feebly trapdoor function (that is, a feebly trapdoor candidate with nontrivial order of security).

2.2 Hard Matrices

All our constructions are based on a linear function $f : \mathbb{B}^n \to \mathbb{B}^n$ shown by A. Hiltgen [1] to be feebly one-way of order 3/2. We restrict ourselves to the case when $n \equiv 0 \pmod 4$. Note that Definition 3 carries through this restriction: for $n \not\equiv 0 \pmod 4$ one can simply consider circuit with input size equal to the lowest multiple of 4 greater than n.

In what follows, all computations are done over $\mathbb{F}_2$. We use standard matrix notation: $\boldsymbol{e}_k$ is the identity $k \times k$ matrix, $\mathbf{0}_k$ is the zero $k \times k$ matrix, $\boldsymbol{e}_{ij}$ is a matrix with a single non-zero element in position (i, j), $\boldsymbol{e}_{i*}$ is a matrix where the i^{th} row consists of 1's, and all other elements are zero, $\mathbf{1}_k$ is the $k \times k$ matrix filled with 1's, $\boldsymbol{u}_k$ is the upper triangular $k \times k$ matrix ($u_{ij} = 1 \Leftrightarrow i < j$), $\boldsymbol{l}_k$ is the lower triangular $k \times k$ matrix ($l_{ij} = 1 \Leftrightarrow i > j$), and $\boldsymbol{m}_\pi$ is the permutation matrix for π ($m_{ij} = 1 \Leftrightarrow j = \pi(i)$). By $\boldsymbol{e}$, $\mathbf{0}$, $\mathbf{1}$, $\boldsymbol{u}$, and $\boldsymbol{l}$ without subscripts we denote the correspondent matrices of dimension $\frac{n}{2} \times \frac{n}{2}$. We also set σ_n to be the cyclic permutation $(1\ 2\ \ldots\ n)$.

In this notation the matrix of the Hiltgen's function f is $A = \boldsymbol{e}_n + \boldsymbol{m}_{\sigma_n} + \boldsymbol{e}_{n,\frac{n}{2}+1}$. We are also interested in the $n \times 2n$ matrix $\mathfrak{A}$ consisting of A^{-2} and A^{-1} stacked together: $\mathfrak{A} = \left(A^{-2}\ A^{-1} \right)$.

The following lemma can be easily verified by direct calculation.

Lemma 1. *Let $n = 4k$ for some $k \in \mathbb{N}$. Then*

$$A^{-1} = \mathbf{1}_n + \begin{pmatrix} \boldsymbol{e}+\boldsymbol{u} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{l} \end{pmatrix}, \quad A^{-2} = \mathbf{1}_n\boldsymbol{u}_n + \boldsymbol{u}_n\mathbf{1}_n + \begin{pmatrix} \boldsymbol{e}+\boldsymbol{u}^2 & \mathbf{0} \\ \mathbf{0} & \boldsymbol{l}^2 \end{pmatrix}.$$

Lemma 2. *Let $n = 4k$ for some $k \in \mathbb{N}$. The following statements hold.*

1. *All columns of $\mathfrak{A}$ (and, hence, A^{-1}) are different.*
2. *Each row of A^{-1} (resp., $\mathfrak{A}$) contains at least $\frac{n}{2}$ (resp., $\frac{5n}{4}$) nonzero entries.*
3. *After eliminating all but two (resp., all but five) columns of A^{-1} (resp., $\mathfrak{A}$) there remains at least one row with two nonzero entries.*

Proof. Let us first interpret the results of Lemma 1. Each row of A contains two ones (on the diagonal and to the right) except for the last row that has three ones, in positions $(n, 1)$, $(n, \frac{n}{2}+1)$, and (n, n). Each row of A^{-1} has at least $\frac{n}{2}$ non-zero elements (ones), and the $(\frac{n}{2}+1)^{\text{th}}$ row does not contain a single zero.

The A^{-2} matrix also has lots of ones: $(\mathbf{1}_n\boldsymbol{u}_n + \boldsymbol{u}_n\mathbf{1}_n)$ is an $n \times n$ matrix filled with zeroes and ones chequered, since

$$\begin{aligned}(\mathbf{1}_n\boldsymbol{u}_n)_{ij} = 1 \quad &\Leftrightarrow \quad j \equiv 1 \pmod 2,\\ (\boldsymbol{u}_n\mathbf{1}_n)_{ij} = 1 \quad &\Leftrightarrow \quad i \equiv 0 \pmod 2.\end{aligned}$$

Moreover,

$$\begin{aligned}(\boldsymbol{e} + \boldsymbol{u}^2)_{ij} = 1 \quad &\Leftrightarrow \quad j > i \quad \text{and} \quad i + j \equiv 0 \pmod 2,\\ (\boldsymbol{l}^2)_{ij} = 1 \quad &\Leftrightarrow \quad i > j \quad \text{and} \quad i + j \equiv 0 \pmod 2,\end{aligned}$$

and thus A^{-2} has two triangular blocks filled with ones: for $1 \le i \le j \le \frac{n}{2}$ and for $\frac{n}{2} + 1 < j < i \le n$. Thus, each row of A^{-2} contains at least $\frac{n}{2}$ ones; moreover, its triangular blocks consisting of ones coincide with the triangular blocks of A^{-1} filled with zeroes, and the rest is covered with zeroes and ones chequered.

The first statement is obvious.

The i^{th} row of A^{-1} contains $\frac{n}{2} + i$ non-zero elements for $i \le \frac{n}{2}$ and $\frac{n}{2} + n - i$ non-zero elements for $i \ge \frac{n}{2}$. Thus, the second statement holds for A^{-1}. At the same time, the i^{th} row of A^{-2} contains at least $\frac{3n}{4} - \frac{i}{2}$ non-zero elements for $i \le \frac{n}{2}$ and at least $\frac{n}{2} + \frac{1}{2}(i - \frac{n}{2} - 1)$ non-zero elements for $i > \frac{n}{2}$. Therefore, the i^{th} row of A^{-2} contains at least $\frac{n}{2} + i + \frac{3n}{4} - \frac{i}{2} = \frac{5n}{4} + \frac{i}{2}$ nonzero entries for $i \le \frac{n}{2}$ and at least $\frac{n}{2} + n - i + \frac{n}{2} + \frac{1}{2}(i - \frac{n}{2} - 1) = \frac{7n}{4} - \frac{1}{2}(i-1) \ge \frac{5n}{4}$ nonzero entries for $i \ge \frac{n}{2}$.

Let us now prove the third claim. Since A^{-1} has a row that contains only nonzero entries, all but one columns of this matrix should be eliminated to leave just one nonzero entry. The same holds for the left part of the matrix A^{-2} (see its first row). The same holds for the right part of the matrix A^{-2} without the last column (see its last row). $\square$

2.3 Gate Elimination

In this section, we first briefly remind about Hiltgen's methods and then introduce gate elimination as the primary technique for proving our bounds. Hiltgen proved all his bounds with the following very simple argument due to Lamagna and Savage.

Proposition 1 ([6,7]; [1, Theorems 3 and 4])

1. *Suppose that $f : \mathbb{B}^n \to \mathbb{B}$ depends non-idly on each of its n variables, that is, for every i there exist values $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n \in \mathbb{B}$ such that $f(a_1, \ldots, a_{i-1}, 0, a_{i+1}, \ldots, a_n) \ne f(a_1, \ldots, a_{i-1}, 1, a_{i+1}, \ldots, a_n)$. Then $C(f) \ge n - 1$.*
2. *Let $f = (f^{(1)}, \ldots, f^{(m)}) : \mathbb{B}^n \to \mathbb{B}^m$, where $f^{(k)}$ is the k^{th} component of f. If the m component functions $f^{(i)}$ are pairwise different and each of them satisfies $C(f^{(i)}) \ge c \ge 1$ then $C(f) \ge c + m - 1$.*

Hiltgen counted the minimal complexity of computing one bit of the input (e.g. since each row of A^{-1} has at least $\frac{n}{2}$ nonzero entries, the minimal complexity of each component of $A^{-1}\boldsymbol{y}$ is $\frac{n}{2}$) and thus produced lower bounds on the complexity of inverting the function (e.g. the complexity of computing $A^{-1}\boldsymbol{y}$ is $\frac{n}{2} + n - 2 = \frac{3n}{2} - 2$).

Besides, in cryptography it is generally desirable to prove not only worst-case bounds, but also that an adversary is unable to invert the function on a substantial fraction of inputs. Indeed, for each of the matrices constructed here, any circuit using less than the minimal necessary number of gates inverts it on less than $\frac{3}{4}$ of the inputs. In Hiltgen's works, this fact followed from a very simple observation (which was not even explicitly stated).

Lemma 3. *Consider a function $f = \bigoplus_{i=1}^{n} x_i$. For any g that depends on only $m < n$ of these variables, $\Pr_{x_1,\ldots,x_n}\left[f(x_1,\ldots,x_n) = g(x_{i_1},\ldots,x_{i_m})\right] = \frac{1}{2}$.*

Proof. Since $m < n$, there exists an index $j \in 1..n$ such that g does not depend on x_j. This means that for every set of values of the other variables, whatever the value of g is, for one of the values of x_j f agrees with g, and on the other value f differs from g. This means that f differs from g on precisely $\frac{1}{2}$ of the inputs. □

The argument suffices for Hiltgen's feebly one-wayness result for the square matrix A^{-1}: first we apply the first part of Proposition 1 and see that every output has complexity at least $\frac{n}{2} - 1$, and then the second part of Proposition 1 yields the necessary bound of $\frac{3n}{2} - 1$. Moreover, if a circuit has less than the necessary number of gates, one of its outputs inevitably depends on less than the necessary number of input variables, which, by Lemma 3, gives the necessary $\frac{1}{2}$ error rate.

However, in this paper we also use non-square matrices, and it turns out that a similar simple argument does not provide sufficient bounds for our matrices. Therefore, we use a different way of proving lower bounds, namely *gate elimination* that has been previously used for every lower bound in "regular" circuit complexity [9]. The basic idea of this method is to use the following inductive argument. Consider function f and substitute some value c for some variable x thus obtaining a circuit for the function $f|_{x=c}$. The circuit can be simplified, because the gates that had this variable as inputs become either unary (recollect that the negation can be embedded into subsequent gates) or constant (in this case we can proceed to eliminating subsequent gates). The important case here is when the gate is non-linear, such as an AND or an OR gate. In this case it is always possible to choose a value for an input of such gate so that this gate becomes a constant. One then proceeds by induction as long as it is possible to find a suitable variable that eliminates many enough gates. Evidently, the number of eliminated gates is a lower bound on the complexity of f.

First, we prove a prerequisite to the master lemma.

Lemma 4. *Let $t \geq 0$. Assume that $\chi : \mathbb{B}^{v(n)} \to \mathbb{B}^n$ is a linear function with matrix X over $GF(2)$. Assume also that all columns of X are different, there* *are no zero rows in X, and after removing any t columns of X, the matrix still has at least one row containing at least two nonzero entries. Then $C(\chi) \geq t+1$ and, moreover, no circuit with less than $t+1$ gates can compute χ on more than $\frac{1}{2}$ of the inputs.*

Proof. We argue by induction on t. For $t = 0$ the statement is obvious: the value of a circuit with no gates[1] cannot agree on more than $\frac{1}{2}$ of the input assignments with a linear function essentially depending on two variables.

Let now $t \geq 1$ and consider an optimal circuit of size less than $t+1$ implementing χ on more than $\frac{1}{2}$ of the inputs. Let us fix a topological order on its nodes. We denote the actual function this circuit implements by h (it does not need to be linear, but does have to agree with χ on more than $\frac{1}{2}$ of the inputs).

Consider the topmost gate g in this order. Since g is topmost, its incoming edges come from the inputs of the circuit, denote them by x and y. To eliminate a gate, we simply substitute a value to x; substituting a value for one variable is equivalent to removing a column from the matrix, and it reduces t by at most 1.

To invoke the induction hypothesis, it remains to note that if h agrees with χ on more than $\frac{1}{2}$ of the inputs, then either $h|_{x=0}$ or $h|_{x=1}$ agrees with the corresponding restriction of χ on more than $\frac{1}{2}$ of the remaining inputs. Thus, if h did compute χ on more than $\frac{1}{2}$ of the inputs, substituting this value of x into h would yield a function of $n-1$ inputs that contradicted the induction hypothesis. Thus, substituting this value of x into χ yields a function of $n-1$ inputs satisfying the conditions of the theorem with parameter $t-1$, and this function can be computed by a circuit of size less than $t-1$, which contradicts the induction hypothesis. □

The following is a "master" lemma that we will apply to our matrices.

Lemma 5. *Let $t \geq u \geq 1$. Assume that $\chi : \mathbb{B}^{v(n)} \to \mathbb{B}^n$ is a linear function with matrix X over $GF(2)$. Assume also that all columns of X are different, every row of X has at least u nonzero entries, and after removing any t columns of X, the matrix still has at least one row containing at least two nonzero entries. Then $C(\chi) \geq u+t$ and, moreover, $C_{3/4}(\chi) \geq u+t$.*

Proof. This time, we argue by induction on u.

The induction base ($u = 1$) follows from Lemma 4.

Let now $u \geq 2$ and consider an optimal circuit of size less than $u+t$ implementing χ on more than $\frac{3}{4}$ of the inputs and fix a topological order on its nodes. We denote the actual function this circuit implements by h (it does not need to be linear, but does have to agree with χ on more than $\frac{3}{4}$ of the inputs).

Consider the topmost gate g in this order. Since g is topmost, its incoming edges come from the inputs of the circuit, denote them by x and y. By Lemma 3, neither of its input variables can be marked as an output, because for $u \geq 2$ each row has at least two variables.

The following possibilities exhaust all possible cases.

[1] Recall that it can still compute unary functions.

1. One of the input variables of g, say x, enters some other gate. In this case by setting x to any constant we can eliminate at least 2 gates. To invoke the induction hypothesis, it remains to note that if h agrees with χ on more than $\frac{3}{4}$ of the inputs, then either $h|_{x=0}$ or $h|_{x=1}$ agrees with the corresponding restriction of χ on more than $\frac{3}{4}$ of the remaining inputs. Thus, substituting this value of x into χ yields a function of $n-1$ inputs satisfying the conditions of the theorem with parameters $u-1$ and $t-1$, and this function can be computed by a circuit of size less than $u+t-2$, which contradicts the induction hypothesis.
2. Neither x nor y enters any other gate. In this case, h is a function of neither x nor y but only $g(x,y)$ (if any); we show that this is impossible. Note that χ depends on x and y separately; in particular, for one of these variables, say x, there exists an output gate[2] χ_i that depends only on x: $\chi_i = x \oplus \bigoplus_{z \in Z} z$, where $y \notin Z$. Since every gate of an optimal circuit essentially depends on both its arguments, there exist values a and b such that $g(0,a) = g(1,b)$. Thus, for every assignment of the remaining variables $h_i \neq \chi_i$ either for $x=0,\ y=a$ or for $x=1,\ y=b$, which means that χ and h disagree on at least $\frac{1}{4}$ of all assignments. □

In what follows we will also use block-diagonal matrices. Intuition hints that joint computation of two functions that have different inputs should be as hard as computing them separately (thus, the lower bound should be the sum of respective lower bounds). However, for certain functions it is not the case, as seen in [9, Section 10.2] We show it for our particular case.

Lemma 6. *Assume that a linear function ζ is determined by a block diagonal matrix*

$$\zeta\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(m)}\right) = \begin{pmatrix} X_1 & 0 & \ldots & 0 \\ 0 & X_2 & \ldots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \ldots & X_m \end{pmatrix} \begin{pmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(m)} \end{pmatrix},$$

and the matrices X_j satisfy the requirements of Lemma 5 with $u_j(n)$ and $t_j(n)$, respectively (the matrices may be non-square and of different dimensions). Then $C(\zeta) \geq \sum_{j=1}^{m}(u_j(n)+t_j(n))$ and, moreover, $C_{3/4}(\zeta) \geq \sum_{j=1}^{m}(u_j(n)+t_j(n))$.

Proof. We proceed similarly to Lemma 5. Note that when we substitute a variable from $x^{(1)}$, it does not change anything in X_2, and vice versa. Thus we substitute "good" variables (those that eliminate two gates) as long as we have them and then substitute "bad" variables (eliminating one gate per step) when we do not have good ones *separately for each matrix.* If one of the matrices runs out of rows that contain at least two nonzero entries (it may happen after eliminating $u_i(n)-1$ "good" and then $t_i(n)-u_i(n)+2$ other variables from it), we substitute the remaining variables corresponding to this matrix and forget about this part of the block-diagonal matrix.

[2] Recall that we distinguish outputs from output gates: an output can be the negation of a function computed in an output gate.

It can happen, however, that one of the inputs (variables) in the topmost gate is from $x^{(1)}$ and the other one is from $x^{(2)}$. Both cases from the proof of Lemma 5 go through smoothly in this situation: in the first case we substitute a value in the good variable, and the second case is impossible for the same reasons.

Thus, eliminating all columns from X_i leads to eliminating at least

$$2(u_i - 1) + (t_i - u_i + 2) = t_i + u_i$$

gates, and we obtain the overall bound of

$$C_{3/4}(\zeta) \geq \sum_{j=1}^{m} (u_j + t_j). \qquad \square$$

We now formulate the direct consequences of these lemmas and note upper bounds for our specific matrices.

Lemma 7. *Let $n, n' \equiv 0 \pmod 4$,*

$$\alpha(\boldsymbol{x}) = A^{-1}\boldsymbol{x}, \quad \alpha_2(\boldsymbol{x}) = \begin{pmatrix} A^{-1} & A^{-2} \end{pmatrix} \boldsymbol{x}, \quad \alpha_*(\boldsymbol{x}) = \begin{pmatrix} A^{-1} & A^{-2} & 0 \\ 0 & 0 & A_*^{-1} \end{pmatrix} \boldsymbol{x},$$

where A_*^{-1} denotes a matrix with the same structure as A^{-1}, but with dimension n' instead of n. Then $C_{3/4}(\alpha) \geq \frac{3n}{2} - 2$, $C_{3/4}(\alpha_2) \geq \frac{13n}{4} - 5$, $C_{3/4}(\alpha_*) \geq \frac{3n'}{2} + \frac{13n}{4} - 7$.

Proof. Follows from Lemmas 5 and 6, by substituting the respective bounds $u(n)$ and $t(n)$ from Lemma 2 (in particular, $t(n) = n - 2$ for the matrix A^{-1} and $t(n) = 2n - 5$ for $\mathfrak{A}$). $\square$

Lemma 8. *1. There exists a circuit of size $\frac{3n}{2} - 1$ that implements the linear function $\phi : \mathbb{B}^n \to \mathbb{B}^n$ with matrix A^{-1}.*

2. There exists a circuit of size $\frac{7n}{2}$ that implements the linear function $\phi : \mathbb{B}^{2n} \to \mathbb{B}^n$ with matrix $\begin{pmatrix} A^{-1} & A \end{pmatrix}$.

3. There exists a circuit of size $\frac{5n}{2} - 1$ that implements the linear function $\phi : \mathbb{B}^{2n} \to \mathbb{B}^n$ with matrix $\begin{pmatrix} A^{-1} & A^{-1} \end{pmatrix}$.

Proof

1. First construct the sum $\bigoplus_{i=1}^{n/2} x_i$ ($\frac{n}{2} - 1$ gates). Then, adding one by one each of the inputs x_i, $i = \frac{n}{2}..n$, compute all outputs y_i, $i = \frac{n}{2}..n$ and, by the way, the sum of all inputs $\bigoplus_{i=1}^{n} x_i$ (this takes another $\frac{n}{2}$ gates). Finally, the first $\frac{n}{2}$ outputs will be computed by "subtracting" the first $\frac{n}{2}$ inputs from the sum of all inputs one by one (another $\frac{n}{2}$ gates).
2. To implement the left part of this matrix, we need $\frac{3n}{2} - 1$ gates. Afterwards we add to each output the two bits from the right part of the matrix (three bits in case of the last row); we add $2n + 1$ gates in this way.
3. Note that in this case $\phi(a, b) = \begin{pmatrix} A^{-1} & A^{-1} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = A^{-1}(a \oplus b)$ for any $a, b \in \mathbb{B}^n$. Thus, we first add $a \oplus b$ (n gates) and then implement A^{-1} ($\frac{3n}{2} - 1$ gates). $\square$

3 The Feebly Secure Trapdoor Function

3.1 Two Constructions

We are almost ready to present the construction of our feebly trapdoor function (recall Definition 1). In this section, we consider two different constructions, none of which works alone; however, we will merge them into one in the subsequent section, and the resulting mixture will be feebly secure.

In our first construction, the inversion with trapdoor is faster than inversion without trapdoor, but, unfortunately, evaluating the function is even harder. In terms of Definition 1, we now present a feebly trapdoor candidate with identical lengths of the seed, public information, trapdoor, input, and output $c(n) = m(n) = \mathrm{pi}(n) = \mathrm{ti}(n) = n$. Given a random seed, the sampler produces a pair of public and trapdoor information $(\mathrm{pi}, \mathrm{ti})$, where ti is the random seed itself and $\mathrm{pi} = A(\mathrm{ti})$ (thus, the sampler can be implemented using $n + 1$ gates). In this construction, evaluation produces the output c for an input m as follows:

$$\mathrm{Eval}_n(\mathrm{pi}, m) = A^{-1}(\mathrm{pi}) \oplus A(m).$$

An upper bound on evaluation circuit complexity follows from Lemma 8; one can evaluate this function with a circuit of size $\frac{7n}{2}$. Inversion with trapdoor goes as follows:

$$\mathrm{Inv}_n(\mathrm{ti}, c) = A^{-1}(A^{-1}(\mathrm{pi}) \oplus c) = A^{-1}(\mathrm{ti} \oplus c).$$

Due to the nice linearity (note that bounds proven in previous sections do not apply here, because the inversion matrix has a lot of identical columns), this circuit can be implemented in $\frac{5n}{2} - 1$ gates: first one computes $\mathrm{ti} \oplus c$ using n gates, then one applies A^{-1} using another $\frac{3n}{2} - 1$ gates (see Lemma 8).

Finally, an adversary has to invert Bob's message the hard way:

$$m = A^{-1}(A^{-1}(\mathrm{pi}) \oplus c) = \mathfrak{A}\begin{pmatrix} \mathrm{pi} \\ c \end{pmatrix}.$$

By Lemma 7, the complexity of this function is at least $\frac{13n}{4} - 5$ gates, and any adversary with less than $\frac{13n}{4} - 5$ gates fails on at least $1/4$ of the inputs.

In this construction evaluation is harder than inversion without trapdoor. In order to fix this problem, we use also a different construction, also a candidate trapdoor function now with $c(n) = m(n) = n$ and $\mathrm{pi}(n) = \mathrm{ti}(n) = 0$. Our second construction is just the Hiltgen's feebly one-way function. Thus, the public and trapdoor information is not used at all, and the evaluation–inversion functions are as follows: $\mathrm{Eval}_n(m) = A(m)$, $\mathrm{Inv}_n(c) = A^{-1}(c)$, $\mathrm{Adv}_n(c) = A^{-1}(c)$.

This construction, of course, is not a trapdoor function at all because inversion is implemented with no regard for the trapdoor. For a message m of length $|m| = n$ the evaluation circuit has $n+1$ gates, while inversion, by Lemma 8, can be performed only by circuits of $\frac{3n}{2} - 1$ gates each. Thus, in this construction evaluation is easy, while inversion is hard, with or without trapdoor.

3.2 The Combined Construction

We now combine the two functions presented in Section 3.1. The resulting one will make it easier for both inversion with trapdoor and evaluation than for an adversary. We split the input into two parts; the first part m_1, of length n, will be subject to our first (less trivial) construction, while the second part m_2, of length αn, will be subject to the second construction. We will choose α later to maximize the relative hardness for an adversary.

Now each participant has a block-diagonal matrix:

$$\mathrm{Eval}_n(\mathrm{pi}, m) = \begin{pmatrix} A^{-1} & A & 0 \\ 0 & 0 & A_* \end{pmatrix} \begin{pmatrix} \mathrm{pi} \\ m_1 \\ m_2 \end{pmatrix} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix},$$
$$\mathrm{Inv}_n(\mathrm{ti}, c) = \begin{pmatrix} A^{-1} & A^{-1} & 0 \\ 0 & 0 & A_*^{-1} \end{pmatrix} \begin{pmatrix} \mathrm{ti} \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix},$$
$$\mathrm{Adv}_n(\mathrm{pi}, m) = \begin{pmatrix} A^{-2} & A^{-1} & 0 \\ 0 & 0 & A_*^{-1} \end{pmatrix} \begin{pmatrix} \mathrm{pi} \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} m_1 \\ m_2 \end{pmatrix},$$

where A_* denotes the matrix with the same structure as A, but with dimension αn instead of n. Thus, in terms of Definition 1, we get a feebly trapdoor candidate where inputs and outputs are longer than the seed and the public and trapdoor information: $\mathrm{pi}(n) = \mathrm{ti}(n) = n$, $c(n) = m(n) = (1+\alpha)n$.

Lemma 8 yields upper bounds for evaluation and inversion with trapdoor, and Lemma 7 yields a lower bound for the adversary: $C(\mathrm{Eval}_n) \le \frac{7n}{2} + \alpha n + 1$, $C(\mathrm{Inv}_n) \le \frac{5n}{2} + \frac{3\alpha n}{2} - 2$, $C_{3/4}(\mathrm{Adv}_n) \ge \frac{13n}{4} + \frac{3\alpha n}{2} - 7$. Thus, to get a feebly trapdoor function we simply need to choose α such that $\frac{13}{4} + \frac{3}{2} > \frac{7}{2} + \alpha$ and $\frac{13}{4} + \frac{3}{2} > \frac{5}{2} + \frac{3\alpha}{2}$. The second inequality is trivial, and the first one yields $\alpha > \frac{1}{2}$.

We would like to maximize the order of security of this trapdoor function (Definition 3); since sampling is always strictly faster than evaluation and inversion with trapdoor, we are maximizing

$$\min\left\{ \lim_{n\to\infty} \frac{C_{3/4}(\mathrm{Adv}_n)}{C(\mathrm{Inv}_n)}, \lim_{n\to\infty} \frac{C_{3/4}(\mathrm{Adv}_n)}{C(\mathrm{Eval}_n)} \right\} = \min\left\{ \frac{\frac{13}{4} + \frac{3\alpha}{2}}{\frac{5}{2} + \frac{3\alpha}{2}}, \frac{\frac{13}{4} + \frac{3\alpha}{2}}{\frac{7}{2} + \alpha} \right\}.$$

This expression reaches maximum when $\alpha = 2$, and the order of security in this case reaches $\frac{25}{22}$. We summarize this in the following theorem.

Theorem 1. *There exists a feebly trapdoor function with the seed length* $\mathrm{pi}(n) = \mathrm{ti}(n) = n$*, the length of inputs and outputs* $c(n) = m(n) = 3n$*, and the order of security* $\frac{25}{22}$*.*

3.3 Hardness Amplification

In the previous sections, we saw a construction of a linear feebly trapdoor function that guarantees that any circuit with less than precisely the necessary number of gates fails to invert this function on more that $\frac{3}{4}$ of its inputs. In this section, we use this construction to design a function with a superpolynomial bound on the probability of success (namely, $2^{-c\sqrt{n}+o(\sqrt{n})}$). Let us denote by h the function an adversary had to compute in the previous section, and by X its matrix. Consider the linear function H defined by the block diagonal matrix

$$H\left(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(m)}\right) = \begin{pmatrix} X & 0 & \dots & 0 \\ 0 & X & \dots & 0 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & X \end{pmatrix} \begin{pmatrix} \boldsymbol{x}^{(1)} \\ \boldsymbol{x}^{(2)} \\ \vdots \\ \boldsymbol{x}^{(m)} \end{pmatrix}.$$

By Lemma 6, H has complexity at least $mC(h)$. The dimensions of X are $(1+\alpha)n \times (2+\alpha)n$; we denote $n' = (1+\alpha)n$.

Lemma 9. *If a circuit computes H on more that $\frac{1}{p(m)}$ fraction of its inputs, and for each block in H:*

- *all columns of X are different;*
- *every row of X has at least $u(n)$ nonzero entries;*
- *after removing any $t(n)$ columns of X, this matrix still has at least one row containing at least two nonzero entries,*

then the complexity of this circuit is not less than $(u(n)+t(n))(m-\log_{4/3} p(m))$.

Proof. First, recall that H consists of m separate blocks with disjoint sets of variables X_i; let us denote $h_i = H|_{X_i}$. Since X_i are disjoint, mistakes in computing h_i are independent: if a circuit C computes h_i on β_i fraction of its inputs and h_j on β_j fraction of its inputs, it cannot compute H on more that $\beta_i\beta_j$ fraction of its inputs. Thus, there are at most $\log_{4/3} p(m)$ blocks where C violating our claim can afford to make mistakes on more than $\frac{1}{4}$ of the inputs. During this proof we will call them "terrible" blocks.

We proceed by the same gate elimination induction we had been using several times already. Consider the topmost gate and the two variables that enter it. In the proof of Lemma 6, we marked variables as "good" or "bad" depending on whether they fall into a block where all "good" variables have been eliminated. This time, we do the same thing, but mark all variables in terrible blocks as "bad" in advance. As in the previous proofs, whenever the topmost gate has at least one "bad" variable as input, we set this variable, thus eliminating only one gate from the circuit. Whenever the topmost gate has two "good" variables as inputs, we should always be able to eliminate two gates from the circuit. There are still the same basic possibilities as in Lemma 5, and we also have to always choose the part of the input assignments space where the circuit errs on less than $\frac{1}{4}$ of the remaining inputs (since the initial circuit errs on less than $\frac{1}{4}$ of these inputs, such a subcurcuit must exist).

The rest of the proof is the same as Lemma 5. We proceed by induction, eliminating two gates whenever two "good" variables enter a topmost gate, and eliminating one "bad" variable whenever it enters a topmost gate. Thus, the overall complexity is at least twice the number of "good" variables plus the number of remaining "bad" variables. We have to discard "terrible" blocks completely — after all, their complexity may actually be equal to zero. Thus, we get a bound of $(t+u)(m-\log_{4/3} p(m))$. $\square$

Note also that stacking the matrices up in a large block diagonal matrix does not change the parameters of a feebly trapdoor function. Thus, we have obtained the following theorem.

Theorem 2. *There exists a feebly trapdoor function candidate* $\mathcal{C}=\{\mathrm{Key}_n, \mathrm{Eval}_n, \mathrm{Inv}_n\}$ *with the seed length* $\mathrm{pi}(n) = \mathrm{ti}(n) = n$*, the length of inputs and outputs* $c(n) = m(n) = 3n$ *with complexities* $C(\mathrm{Inv}_n) \le \frac{11n}{2} + O(1)$*,* $C(\mathrm{Eval}_n) \le \frac{11n}{2} + O(1)$*,* $C(\mathrm{Key}_n) = n+1$*, and the order of security* $\frac{25}{22}$*. Moreover, no adversary with less than* $\frac{25}{4}n - \frac{25}{4}\delta n^{(a+1)/2}$ *gates is able to invert this feebly trapdoor function on more than* $(4/3)^{-\delta n^{a/2}+o(n)}$ *fraction of the inputs for any constant* $\delta > 0, 1 > a > 0$*.*

4 Conclusion and Further Work

In this work, we have presented the first known construction of a provably secure trapdoor function, although "security" should be understood in a very restricted sense of Definition 3.

Here are natural directions for further research. First, it would be interesting to devise a more natural construction. Both the second (keyless) construction and the merge of two matrices are counter-intuitive. Second, it would be great to substantially improve the order of security. While a certain improvement to the constant $\frac{25}{22}$ may be straightforward, further development will soon hit the general obstacle of our present inability to prove lower bounds greater than $4n - o(1)$ for $\mathbb{B}^n \to \mathbb{B}^n$ functions. Moreover, the constructions based on linear functions will never overcome a bound of $n-1$ gates per one bit of output; thus, nonlinear constructions are necessary. Finally, a natural extension of this work would be to devise other feebly secure cryptographic primitives.

Acknowledgements

The authors are grateful to Olga Melanich who noticed a flaw in the initial version of the proof at the very last moment and to the anonymous referees for useful comments.

References

1. Hiltgen, A.P.: Constructions of freebly-one-way families of permutations. In: Proc. of AsiaCrypt 1992, pp. 422–434 (1992)
2. Levin, L.A.: One-way functions and pseudorandom generators. Combinatorica 7(4), 357–363 (1987)

3. Goldreich, O.: Foundations of Cryptography. Cambridge University Press, Cambridge (2001)
4. Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfers and other primitives. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 96–113. Springer, Heidelberg (2005)
5. Grigoriev, D., Hirsch, E.A., Pervyshev, K.: A complete public-key cryptosystem. Groups, Complexity, Cryptology 1(1), 1–12 (2008)
6. Lamagna, E.A., Savage, J.E.: On the logical complexity of symmetric switching functions in monotone and complete bases. Technical report, Brown University, Rhode Island (July 1973)
7. Savage, J.E.: The Complexity of Computing. Wiley, New York (1976)
8. Blum, N.: A boolean function requiring $3n$ network size. Theoretical Computer Science 28, 337–345 (1984)
9. Wegener, I.: The Complexity of Boolean Functions. B. G. Teubner, and John Wiley & Sons (1987)

Partitioning Graphs into Connected Parts

Pim van 't Hof[1,⋆], Daniël Paulusma[1,⋆], and Gerhard J. Woeginger[2,⋆⋆]

[1] Department of Computer Science, University of Durham,
Science Laboratories, South Road, Durham DH1 3LE, England
{pim.vanthof,daniel.paulusma}@durham.ac.uk

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

Abstract. The 2-Disjoint Connected Subgraphs problem asks if a given graph has two vertex-disjoint connected subgraphs containing prespecified sets of vertices. We show that this problem is NP-complete even if one of the sets has cardinality 2. The Longest Path Contractibility problem asks for the largest integer ℓ for which an input graph can be contracted to the path P_ℓ on ℓ vertices. We show that the computational complexity of the Longest Path Contractibility problem restricted to P_ℓ-free graphs jumps from being polynomially solvable to being NP-hard at $\ell = 6$, while this jump occurs at $\ell = 5$ for the 2-Disjoint Connected Subgraphs problem. We also present an exact algorithm that solves the 2-Disjoint Connected Subgraphs problem faster than $\mathcal{O}^*(2^n)$ for any n-vertex P_ℓ-free graph. For $\ell = 6$, its running time is $\mathcal{O}^*(1.5790^n)$. We modify this algorithm to solve the Longest Path Contractibility problem for P_6-free graphs in $\mathcal{O}^*(1.5790^n)$ time.

1 Introduction

There are several natural and elementary algorithmic problems that check if the structure of some fixed graph H shows up as a pattern within the structure of some input graph G. One of the most well-known problems is the H-Minor Containment problem that asks whether a given graph G contains H as a minor. A celebrated result by Robertson and Seymour [12] states that the H-Minor Containment problem can be solved in polynomial time for every fixed pattern graph H. They obtain this result by designing an algorithm that solves the following problem in polynomial time for any fixed input parameter k.

Disjoint Connected Subgraphs
Instance: A graph $G = (V, E)$ and mutually disjoint nonempty sets $Z_1, \ldots, Z_t \subseteq V$ such that $\sum_{i=1}^{t} |Z_i| \leq k$.
Question: Do there exist mutually vertex-disjoint connected subgraphs $G_1, \ldots, G_t$ of G such that $Z_i \subseteq V_{G_i}$ for $1 \leq i \leq t$?

⋆ Supported by EPSRC (EP/D053633/1).
⋆⋆ Supported by NWO grant 639.033.403, and by BSIK grant 03018.

The first problem studied in this paper is the 2-Disjoint Connected Subgraphs problem, which is a restriction of the above problem to $t = 2$.

The *cyclicity* $\eta(G)$ of a connected graph G, introduced by Blum [2], is the largest integer ℓ for which G is contractible to the cycle C_ℓ on ℓ vertices. We introduce a similar concept: the *path contractibility number* $\vartheta(G)$ of a graph G is the largest integer ℓ for which G is P_ℓ-contractible. For convenience, we define $\vartheta(G) = 0$ if and only if G is disconnected. The second problem studied in this paper is the Longest Path Contractibility problem, which asks for the path contractibility number of a given graph G.

Like the 2-Disjoint Connected Subgraphs problem, the Longest Path Contractibility problem deals with partitioning a given graph into connected subgraphs. Since connectivity is a "global" property, both problems are examples of "non-local" problems, which are typically hard to solve exactly (see e.g. [5]). In an attempt to design fast exact algorithms for such problems, one can focus on restrictions of the problem to certain graph classes. One family of graph classes of particular interest is the family of graphs that do not contain long induced paths. Several authors have studied restrictions of well-known NP-hard problems, such as the k-Colorability problem (cf. [8,11,13]) and the Maximum Independent Set problem (cf. [7,10]), to the class of P_ℓ-free graphs for several values of ℓ.

Our Results. We show that the 2-Disjoint Connected Subgraphs problem is already NP-complete if one of the given sets of vertices has cardinality 2. We also show that the 2-Disjoint Connected Subgraphs problem restricted to the class of P_ℓ-free graphs jumps from being polynomially solvable to being NP-hard at $\ell = 5$, while for the Longest Path Contractibility problem this jump occurs at $\ell = 6$.

A trivial algorithm solves the Two Disjoint Connected Subgraphs problem in $\mathcal{O}^*(2^n)$ time. Let $\mathcal{G}^{k,r}$ denote the class of graphs all connected induced subgraphs of which have a connected r-dominating set of size at most k. We present an algorithm, called `SPLIT`, that solves the 2-Disjoint Connected Subgraphs problem for n-vertex graphs in the class $\mathcal{G}^{k,r}$ in $\mathcal{O}^*((f(r))^n)$ time for any fixed k and $r \geq 2$, where

$$f(r) = \min_{0<c\leq 0.5}\left\{\max\left\{\frac{1}{c^c(1-c)^{1-c}}, 2^{1-\frac{2c}{r-1}}\right\}\right\}.$$

The $\mathcal{O}^*$-notation, used throughout the paper, suppresses factors of polynomial order. In particular, `SPLIT` solves the 2-Disjoint Connected Subgraphs problem for any n-vertex P_6-free graph in $\mathcal{O}^*(1.5790^n)$ time. We modify `SPLIT` to obtain an $\mathcal{O}^*(1.5790^n)$ time algorithm for the Longest Path Contractibility problem restricted to P_6-free graphs on n vertices.

2 Preliminaries

All graphs in this paper are undirected, finite, and *simple*, i.e., without loops and multiple edges. We refer to [4] for terminology not defined below.

Let $G = (V, E)$ be a graph. For a subset $S \subseteq V$ we write $G[S]$ to denote the the subgraph of G *induced* by S. We write P_ℓ respectively C_ℓ to denote a path respectively a cycle on ℓ vertices. The *distance* $d_G(u, v)$ between two vertices u and v in a graph G is the *length* $|V_P| - 1$ of a shortest path P between them. For any vertex $v \in V$ and set $S \subseteq V$, we write $d_G(v, S)$ to denote the length of a shortest path from v to S, i.e., $d_G(v, S) := \min_{w \in S} d_G(v, w)$. The *neighborhood* of a vertex $u \in V$ is the set $N_G(u) := \{v \in V \mid uw \in E\}$. The set $N_G^r(S) := \{u \in V \mid d_G(u, S) \leq r\}$ is called the *r-neighborhood* of a set S. A set S *r-dominates* a set S' if $S' \backslash S \subseteq N_G^r(S)$. We also say that S r-dominates $G[S']$. A subgraph H of G is an *r-dominating subgraph* of G if V_H r-dominates G. In case $r = 1$, we use "dominating" instead of "1-dominating". A set $S \subseteq V$ is called a *(k, r)-center* of G if $|S| \leq k$ and $N_G^r(S) = V$. A set S is called *connected* if $G[S]$ is connected. The class of graphs all connected induced subgraphs of which have a connected (k, r)-center is denoted by $\mathcal{G}^{k,r}$. The graph G is called a *split graph* if V can be partitioned into a clique and an independent set.

Let $V' \subset V$ and $p, q \in V \backslash V'$. We say that p is *separated* from q by V' if every path in G from p to q contains a vertex of V'. A graph G is called *H-free* for some graph H if G does not contain an induced subgraph isomorphic to H. The *edge contraction* of edge $e = uv$ in G removes the two end-vertices u and v from G, and replaces them by a new vertex that is adjacent to precisely those vertices to which u or v were adjacent. We denote the resulting graph by $G \backslash e$. A graph G is *contractible to* a graph H (graph G is *H-contractible*) if H can be obtained from G by a sequence of edge contractions. An equivalent way of saying that G is H-contractible is that

- for every vertex h in V_H there is a corresponding nonempty subset $W(h) \subseteq V_G$ of vertices in G such that $G[W(h)]$ is connected, and $\mathcal{W} = \{W(h) \mid h \in V_H\}$ is a partition of V_G; we call a set $W(h)$ an *H-witness set* of G for h, and we call $\mathcal{W}$ an *H-witness structure* of G;
- for every $h_i, h_j \in V_H$, there is at least one edge between witness sets $W(h_i)$ and $W(h_j)$ in G if and only if h_i and h_j are adjacent in H.

If for every $h \in V_H$ we contract the vertices in $W(h)$ to a single vertex, then we end up with the graph H. Note that the witness sets $W(h)$ are not uniquely defined in general, since there may be different sequences of edge contractions that lead from G to H. A pair of vertices (u, v) of a graph G is *P_ℓ-suitable* for some integer $\ell \geq 3$ if and only if G has a P_ℓ-witness structure $\mathcal{W}$ with

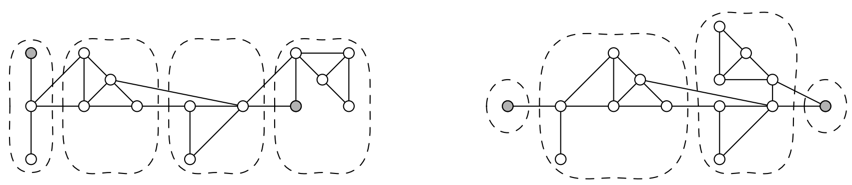


Fig. 1. Two P_4-witness structures of a graph; the grey vertices form a P_4-suitable pair

$W(p_1) = \{u\}$ and $W(p_\ell) = \{v\}$, where $P_\ell = p_1 \dots p_\ell$. See Figure 1 for two different P_4-witness structures and a P_4-suitable pair of a P_4-contractible graph.

A 2-*coloring* of a *hypergraph* $(Q, \mathcal{S})$, where $\mathcal{S}$ is a collection of subsets of Q, is a partition (Q_1, Q_2) of Q with $Q_1 \cap S \neq \emptyset$ and $Q_2 \cap S \neq \emptyset$ for all $S \in \mathcal{S}$.

3 The 2-Disjoint Connected Subgraphs Problem

3.1 An NP-Completeness Proof

Theorem 1. *The 2-*Disjoint Connected Subgraphs *problem restricted to instances with* $|Z_1| = 2$ *is* NP*-complete.*

Proof. We use a reduction from 3-SAT, which is well-known to be NP-complete (cf. [6]). Let $X = \{x_1, \dots, x_n\}$ be a set of variables and $C = \{c_1, \dots, c_m\}$ be a set of clauses forming an instance of 3-SAT. Let $\overline{X} := \{\overline{x} \mid x \in X\}$. We construct a graph G, depicted in Figure 2, as follows. Every literal in $X \cup \overline{X}$ and every clause in C is represented by a vertex in G. There is an edge between $x \in X \cup \overline{X}$ and $c \in C$ if and only if x appears in c. For $i = 1, \dots, n-1$, x_i and $\overline{x}_i$ are adjacent to both x_{i+1} and $\overline{x}_{i+1}$. We add two vertices f_1 and f_2 to G, where f_1 is adjacent to x_1 and $\overline{x}_1$, and f_2 is adjacent to x_n and $\overline{x}_n$.

We claim that the graph G, together with the sets $Z_1 := \{f_1, f_2\}$ and $Z_2 := C$, is a Yes-instance of the 2-Disjoint Connected Subgraphs problem if and only if C is satisfiable.

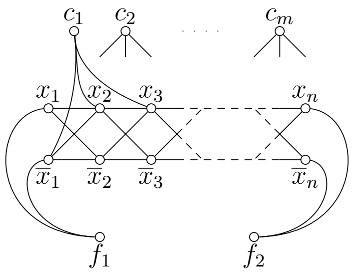


Fig. 2. The graph G, in case $c_1 = (\overline{x}_1 \vee x_2 \vee x_3)$

Suppose $t : X \to \{\text{true}, \text{false}\}$ is a satisfying truth assignment for C. Let X_T (respectively X_F) be the set of variables that are set to true (respectively false) by t, and let $\overline{X}_T := \{\overline{x} \mid x \in X_T\}$ and $\overline{X}_F := \{\overline{x} \mid x \in X_F\}$. We denote the set of true and false literals by T and F respectively, i.e., $T := X_T \cup \overline{X}_F$ and $F := X_F \cup \overline{X}_T$. Note that exactly one literal of each pair $x_i, \overline{x}_i$ belongs to T, i.e., is set to true by t, and the other one belongs to F. Hence, the vertices in $F \cup \{f_1, f_2\}$ induce a connected subgraph G_1 of G. Since t is a satisfying truth assignment, every clause vertex is adjacent to a vertex in T. Hence the vertices in $T \cup C$ induce a connected subgraph G_2 of G, which is vertex-disjoint from G_1.

To prove the reverse statement, suppose G_1 and G_2 are two vertex-disjoint connected subgraphs of G such that $\{f_1, f_2\} \subseteq V_{G_1}$ and $C \subseteq V_{G_2}$. Since f_1 and f_2 form an independent set in G and G_1 is connected, at least one of each pair $x_i, \overline{x}_i$ must belong to V_{G_1}. Since the vertices of C form an independent set in G, every clause vertex must be adjacent to at least one literal vertex in $(X \cup \overline{X}) \cap V_{G_2}$. Let t be a truth assignment that sets those literals to true, and their negations to false. For each pair $x_i, \overline{x}_i$ both literals of which belong to V_{G_1}, t sets exactly one literal to true, and the other one to false. Then t is a satisfying truth assignment for C. □

3.2 A Complexity Classification for P_ℓ-Free Graphs

Consider the following characterization of P_4-free graphs given in [9].

Theorem 2 ([9]). *A graph G is P_4-free if and only if each connected induced subgraph of G contains a dominating induced C_4 or a dominating vertex.*

We use this characterization of P_4-free graphs in the proof of the complexity classification of the 2-Disjoint Connected Subgraphs problem below. Note that we have strengthened the NP-complete cases to split graphs.

Theorem 3. *The* 2-Disjoint Connected Subgraphs *problem is polynomially solvable for P_ℓ-free graphs if $\ell \leq 4$ and* NP*-complete for P_ℓ-free split graphs if $\ell \geq 5$.*

Proof. Assume $\ell \leq 4$. Let $G = (V, E)$ be a P_ℓ-free, and consequently P_4-free, graph with nonempty disjoint sets $Z_1, Z_2 \subseteq V$. Suppose G, together with sets Z_1 and Z_2, is a Yes-instance of the 2-Disjoint Connected Subgraphs problem, and let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be vertex-disjoint connected subgraphs of G such that $Z_i \subseteq V_i$ for $i = 1, 2$. Note that both G_1 and G_2 are P_4-free. As a result of Theorem 2, there exist sets D_1, D_2 such that D_i dominates V_i and $|D_i| \in \{1, 4\}$ for $i = 1, 2$. So to check whether G, together with Z_1 and Z_2, is a Yes-instance of the 2-Disjoint Connected Subgraphs problem, we act as follows.

We guess a vertex $d_1 \in V \backslash Z_2$. If d_1 does not dominate Z_1, we guess another vertex d_1. If d_1 dominates Z_1, we check if Z_2 is contained in one component G_2 of $G[V \backslash (Z_1 \cup \{d_1\})]$. If so, then $G_1 := G[Z_1 \cup \{d_1\}]$ and G_2 form a solution of the 2-Disjoint Connected Subgraphs problem. Otherwise, we choose another vertex d_1. If we have checked every vertex in $V \backslash Z_2$ without finding a solution, then we guess a 4-tuple $D_1 \subseteq V \backslash Z_2$ and repeat the above procedure with D_1 instead of d_1. If we do not find a solution for any 4-tuple D_1, then (G, Z_1, Z_2) is a No-instance of the 2-Disjoint Connected Subgraphs problem. Since we can perform all checks in polynomial time, this finishes the proof of the polynomial cases.

We now show that the 2-Disjoint Connected Subgraphs problem is NP-complete for P_ℓ-free split graphs if $\ell \geq 5$. Clearly, the problem lies in NP. We prove NP-completeness by using a reduction from the NP-complete Hypergraph 2-Colorability problem that asks if a given hypergraph is 2-colorable (cf. [6]). Let $H = (Q, \mathcal{S})$ be a hypergraph with $Q = \{q_1, \ldots, q_n\}$

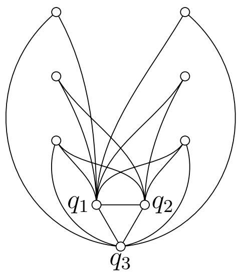


Fig. 3. The graph G

and $\mathcal{S} = \{S_1, \ldots, S_m\}$. We may assume $m \geq 2$ and $S_i \neq \emptyset$ for each S_i. Let G be the graph obtained from the incidence graph of H by adding the vertices $\mathcal{S}' = \{S'_1, \ldots, S'_m\}$, where $S'_i = S_i$ for every $1 \leq i \leq m$, and by adding the following edges: $q_i S'_j$ if and only if $q_i \in S'_j$, and $q_i q_j$ if and only if $i \neq j$. See Figure 3 for the graph G obtained in this way from the hypergraph $(Q, \mathcal{S})$ with $Q = \{q_1, q_2, q_3\}$ and $\mathcal{S} = \{\{q_1, q_3\}, \{q_1, q_2\}, \{q_1, q_2, q_3\}\}$. Clearly G is a split graph, and it is easy to check that G is P_5-free, and consequently P_ℓ-free for any $\ell \geq 5$. We claim that G, together with the sets $\mathcal{S}$ and $\mathcal{S}'$, is a YES-instance of the 2-DISJOINT CONNECTED SUBGRAPHS problem if and only if $(Q, \mathcal{S})$ has a 2-coloring.

Suppose G_1 and G_2 are vertex-disjoint connected subgraphs of G such that $\mathcal{S} \subseteq V_{G_1}$ and $\mathcal{S}' \subseteq V_{G_2}$. Without loss of generality, assume that $V_1 := V_{G_1}$ and $V_2 := V_{G_2}$ form a partition of V. Then there exists a partition (Q_1, Q_2) of Q such that $V_1 = \mathcal{S} \cup Q_1$ and $V_2 = \mathcal{S}' \cup Q_2$. Note that $\mathcal{S}$ is an independent set in G. Hence $Q_1 \neq \emptyset$ and every vertex in $\mathcal{S}$ is adjacent to at least one vertex in Q_1. Similarly, $Q_2 \neq \emptyset$ and every vertex in $\mathcal{S}'$ has at least one neighbor in Q_2. Since $S'_i = S_i$ for every $1 \leq i \leq m$, (Q_1, Q_2) is a 2-coloring of $(Q, \mathcal{S})$.

Now suppose $(Q, \mathcal{S})$ has a 2-coloring (Q_1, Q_2). Then it is clear that $G[\mathcal{S} \cup Q_1]$ and $G[\mathcal{S}' \cup Q_2]$ are connected, so we can choose $G_1 := G[\mathcal{S} \cup Q_1]$ and $G_2 := G[\mathcal{S}' \cup Q_2]$. This finishes the proof of the NP-complete cases.

3.3 An Exact Algorithm

Here, we present an algorithm that solves the 2-DISJOINT CONNECTED SUBGRAPHS problem for $\mathcal{G}^{k,r}$ for any k and $r \geq 2$ faster than the trivial $\mathcal{O}^*(2^n)$.

Lemma 1. *Let $G = (V, E)$ be a connected induced subgraph of a graph $G' \in \mathcal{G}^{k,r}$. For each subset $Z \subseteq V$, there exists a set $D^* \subseteq V$ with $|D^*| \leq (r-1)|Z| + k$ such that $G[D^* \cup Z]$ is connected.*

Proof. By definition of $\mathcal{G}^{k,r}$, G has a connected (k, r)-center D_0. Let $D_i := \{v \in V \mid d_G(v, D_0) = i\}$ for $i = 1, \ldots r$. Note that the sets $D_0, \ldots, D_r$ form a partition of V. Let z be any vertex of Z and suppose $z \in D_i$ for some $0 \leq i \leq r$; note that this i is uniquely defined. By definition, there exists a path P^z of length i from

z to a vertex in D_0, and it is clear that $D_0 \cup P^z \backslash \{z\}$ is a connected set of size $(i-1) + |D_0|$ that dominates z. Let $\mathcal{P} := \bigcup_{z \in Z} P^z \backslash \{z\}$. Clearly, $D^* := D_0 \cup \mathcal{P}$ is a connected set dominating Z. In the worst case, we have $Z \subseteq D_r$ and every pair of paths $P^z, P^{z'}$ is vertex-disjoint, in which case $|D^*| = (r-1)|Z| + |D_0| \leq (r-1)|Z| + k$. This finishes the proof of Lemma 1. □

The following corollary, the straightforward proof of which has been omitted due to page restrictions, is a result of Lemma 1.

Corollary 1. *For any fixed k, the 2-*Disjoint Connected Subgraphs *problem for $\mathcal{G}^{k,r}$ can be solved in polynomial time if $r = 1$, or if one of the given sets Z_1 or Z_2 of vertices has fixed size.*

From now on, we assume that $r \geq 2$ (and that the sets Z_1, Z_2 may have arbitrary size). We present the algorithm SPLIT that solves the 2-Disjoint Connected Subgraphs problem for any $G \in \mathcal{G}^{k,r}$, or concludes that a solution does not exist. We assume $1 \leq |Z_1| \leq |Z_2|$ and define $Z := V \backslash (Z_1 \cup Z_2)$. Algorithm SPLIT distinguishes between whether or not Z_1 has a "reasonably" small size, i.e., size at most an for some number $0 < a \leq \frac{1}{2(r-1)}$, the value of which will be determined later.

Case 1. $|Z_1| \leq an$. For all sets $Z' \subseteq Z$ in order of increasing cardinality up to at most $(r-1)|Z_1| + k$, check whether $G_1 := G[Z' \cup Z_1]$ is connected and $G[(Z \backslash Z') \cup Z_2]$ has a component G_2 containing all vertices of Z_2. If so, output G_1 and G_2. If not, choose another set Z' and repeat the procedure. If no solution is found for any set Z', then output No.

Case 2. $|Z_1| > an$. Perform the procedure described in Case 1 for all sets $Z' \subseteq Z$ in order of increasing cardinality up to at most $\lceil (1-2a)n \rceil$.

Theorem 4. *For any fixed k and $r \geq 2$, algorithm* SPLIT *solves the* 2-Disjoint Connected Subgraphs *problem for any n-vertex graph in $\mathcal{G}^{k,r}$ in $\mathcal{O}^*((f(r))^n)$ time, where*

$$f(r) = \min_{0 < c \leq 0.5} \left\{ \max \left\{ \frac{1}{c^c (1-c)^{1-c}}, 2^{1 - \frac{2c}{r-1}} \right\} \right\}.$$

Proof. Let $G = (V, E)$ be a graph in $\mathcal{G}^{k,r}$ with $|V| = n$, and let $Z_1, Z_2 \subseteq V$ be two nonempty disjoint sets of vertices of G with $1 \leq |Z_1| \leq |Z_2|$. If Case 1 occurs, the correctness of SPLIT follows from Lemma 1. If Case 2 occurs, correctness follows from the fact that all subsets of Z may be checked if necessary, as $|Z_1| > an$ implies $|Z_2| > an$, and therefore $|Z| \leq (1-2a)n$. We are left to prove that the running time mentioned in Theorem 4 is correct. We consider Case 1 and Case 2.

Case 1. $|Z_1| \leq an$. In the worst case, the algorithm has to check all sets $Z' \subseteq Z$ in order of increasing cardinality up to $(r-1)|Z_1| + k \leq (r-1)an + k$. Let $c := (r-1)a$, and note that $c \leq \frac{1}{2}$ since we assumed $a \leq \frac{1}{2(r-1)}$. Then we must check at most $\sum_{i=1}^{cn+k} \binom{n}{i}$ sets Z'. It is not hard to see that $\sum_{i=1}^{cn+k} \binom{n}{i} \leq (cn + (n-cn)^k)\binom{n}{cn}$. Using Stirling's approximation, $n! \approx n^n e^{-n} \sqrt{2\pi n}$, we find that the number of sets we have to check is $\mathcal{O}\left(\frac{cn+(n-cn)^k}{\sqrt{2\pi(1-c)cn}} \cdot \left(\frac{1}{c^c\cdot(1-c)^{1-c}}\right)^n\right)$. For each set all the required checks can be done in polynomial time. Since k is a fixed constant, independent of n, the running time for Case 1 is $\mathcal{O}^*\left(\left(\frac{1}{c^c\cdot(1-c)^{1-c}}\right)^n\right)$.

Case 2. $|Z_1| > an$. In the worst case, we must check all $\mathcal{O}\left(\left(2^{1-2a}\right)^n\right)$ sets of size up to $\lceil(1-2a)n\rceil$. Since for each set all the required checks can be done in polynomial time, the running time for Case 2 is $\mathcal{O}^*\left(\left(2^{1-2a}\right)^n\right) = \mathcal{O}^*\left(\left(2^{1-\frac{2c}{r-1}}\right)^n\right)$.
Since we do not know in advance whether Case 1 or Case 2 will occur, the appropriate value of c can be computed by taking

$$\min_{0<c\leq 0.5}\left\{\max\left\{\frac{1}{c^c\cdot(1-c)^{1-c}}, 2^{1-\frac{2c}{r-1}}\right\}\right\}.$$

This finishes the proof of Theorem 4. □

See Table 1 for the time complexities of SPLIT for some graph classes.

Table 1. The time complexities of SPLIT for some graph classes

Input graph is...	SPLIT runs in...
split	$\mathcal{O}^*(1.5790^n)$
P_5-free	$\mathcal{O}^*(1.5790^n)$
P_6-free	$\mathcal{O}^*(1.5790^n)$
P_ℓ-free ($\ell \geq 7$)	$\mathcal{O}^*((f(\ell-3))^n)$
P_7-free	$\mathcal{O}^*(1.7737^n)$
P_8-free	$\mathcal{O}^*(1.8135^n)$
P_{100}-free	$\mathcal{O}^*(1.9873^n)$

To prove that the time complexities in Table 1 are correct, we use the following result by Bacsó and Tuza [1].

Theorem 5 ([1]). *Let $\ell \geq 7$. A graph G is P_ℓ-free if and only if each connected induced subgraph of G has a dominating subgraph of diameter at most $\ell-4$.*

Theorem 6. *The time complexities of* SPLIT *shown in Table 1 are correct.*

Proof. Since a graph of diameter at most $\ell-4$ has an $(\ell-4)$-dominating vertex, every P_ℓ-free graph is in $\mathcal{G}^{1,\ell-3}$ for each $\ell \geq 7$ as a result of Theorem 5. Evaluating the function f in Theorem 4 at $r=4$, $r=5$ and $r=97$ yields the running times for P_7-free, P_8-free and P_{100}-free graphs in Table 1. Since $f(2) \approx 1.5790$, it remains to show that both the class of split graphs and the class of P_ℓ-free graphs for $\ell \in \{5,6\}$ belong to $\mathcal{G}^{k,2}$ for some constant k.

Since every connected induced subgraph of a split graph has a 2-dominating set of size 1 (namely any vertex of the "clique part" of the split graph), the family of split graphs belongs to $\mathcal{G}^{1,2}$. We now show that the class of P_6-free graphs is in $\mathcal{G}^{4,2}$. In [9], we prove that a graph G is P_6-free if and only if each connected induced subgraph of G contains a dominating induced C_6 or a dominating (not necessarily induced) complete bipartite graph. Every induced C_6 has a dominating connected set of size 4, and every complete bipartite graph has a dominating connected set of size 2, which means that the class of P_6-free graphs is in $\mathcal{G}^{4,2}$.

The observation that the class of P_5-free graphs is a subclass of the class of P_6-free graphs finishes the proof of Theorem 6. Note that the graph obtained from a complete graph on vertices $\{x_1, \ldots, x_p\}$ by adding an edge between each x_i and a new vertex y_i (which is only made adjacent to x_i) is P_5-free and belongs to $\mathcal{G}^{1,2}$. This example shows that we cannot reduce $r = 2$ to $r = 1$ for P_5-free graphs. $\square$

4 The Longest Path Contractibility Problem

4.1 A Complexity Classification for P_ℓ-Free Graphs

Before stating the main theorem of this section, we first present a number of useful results. Brouwer and Veldman [3] give an elegant reduction from the Hypergraph 2-Colorability problem to show that the P_4-Contractibility problem is NP-complete. Given a hypergraph $(Q, \mathcal{S})$ they construct a graph G such that $(Q, \mathcal{S})$ has a 2-coloring if and only if G is P_4-contractible. It is not hard to check that the graph G is P_6-free, which immediately implies the following result.

Theorem 7. *The P_4-Contractibility problem is NP-complete for the class of P_6-free graphs.*

The (straightforward) proofs of the following lemmas have been omitted due to page restrictions.

Lemma 2. *For $\ell \geq 3$, a graph G is P_ℓ-contractible if and only if G has a P_ℓ-suitable pair.*

Lemma 3. *Let x and y be two neighbors of a vertex u in a graph G with $xy \in E_G$, and let v be some other vertex in G. Then (u, v) is a P_ℓ-suitable pair of G if and only if (u, v) is a P_ℓ-suitable pair of $G \backslash xy$.*

Lemma 4. *For any edge xy of a P_ℓ-free graph G, the graph $G \backslash xy$ is P_ℓ-free.*

We now present a polynomial-time algorithm for deciding whether a P_5-free graph is P_4-contractible.

Theorem 8. *The P_4-Contractibility problem is solvable in polynomial time for the class of P_5-free graphs.*

Proof. Let $G = (V, E)$ be a connected P_5-free graph. Lemma 2 states that G is P_4-contractible if and only if G contains a P_4-suitable pair (u, v). Since G has $\mathcal{O}(|V|^2)$ pairs (u, v), it suffices to show that we can check in polynomial time whether a given pair (u, v) is P_4-suitable. It follows from the definition of a P_4-witness structure and the P_5-freeness of G that we only need to consider pairs of vertices at distance 3. If there does not exists such a pair, then G is not P_4-contractible. Suppose (u, v) is a pair of vertices of G with $d_G(u, v) = 3$.

Claim 1. We may without loss of generality assume that $N(u)$ and $N(v)$ are independent sets of cardinality at least 2.

We prove Claim 1 as follows. Lemma 3 and Lemma 4 together immediately imply that we may assume $N(u)$ and $N(v)$ to be independent sets. Now suppose that $N(u)$ has cardinality 1, say $N(u) = \{x\}$. It is clear that (u, v) is a P_4-suitable pair of G if and only if $N(v)$ is contained in one component of $G[V\backslash\{u, v, x\}]$, which can be checked in polynomial time. Hence we may assume that $|N(u)| \geq 2$, and by symmetry $|N(v)| \geq 2$.

Claim 2. Let x and x' be two vertices of G such that x is adjacent to a vertex $w \in N(u)$ but not to a vertex $w' \in N(u)$, and x' is adjacent to w' but not to w. Then $N(u) \subseteq N(x) \cup N(x')$.

We prove Claim 2 as follows. Clearly $u \notin \{x, x'\}$. As $N(u)$ is an independent set by Claim 1, u is neither adjacent to x nor to x'. Then $xx' \in E$, since otherwise the path $x'w'uwx$ is an induced P_5 as a result of Claim 1, contradicting the P_5-freeness of G. Now suppose there exists a vertex $w'' \in N(u)$ not in $N(x) \cup N(x')$. Since w' and w'' are not adjacent as a result of Claim 1, the path $w''uw'x'x$ is an induced P_5 in G. This contradiction proves Claim 2.

Claim 3. Suppose G has a P_4-witness structure $\mathcal{W}$ with $W(p_1) = \{u\}$ and $W(p_4) = \{v\}$. Then at least one of the following holds:

1. *there exists a vertex $x \in W(p_2)\backslash N(u)$ with $N(u) \subseteq N(x)$;*
2. *there exist vertices $x, x' \in W(p_2)\backslash N(u)$ with $N(u) \subseteq N(x) \cup N(x')$.*

We prove this claim as follows. Suppose $\mathcal{W}$ is a P_4-witness structure of G with $W(p_1) = \{u\}$ and $W(p_4) = \{v\}$, and suppose condition 1 does not hold. We show that condition 2 must hold. By Claim 1, $N(u)$ is an independent set of G containing at least two vertices. Since $N(u) \subseteq W(p_2)$ and $G[W(p_2)]$ is connected, we know that $W(p_2)\backslash N(u) \neq \emptyset$. Let $x \in W(p_2)\backslash N(u)$ be a vertex such that $|N(u) \cap N(x)|$ is maximal over all vertices in $W(p_2)\backslash N(u)$. Since condition 1 does not hold, there exists a vertex $w' \in N(u)$ that is not adjacent to x. Then w' is adjacent to a vertex $x' \in W(p_2)\backslash(N(u) \cup \{x\})$, as otherwise w' would be an isolated vertex in $G[W(p_2)]$. By choice of x, there exists a vertex $w \in N(u) \cap N(x)$ not adjacent to x'. By Claim 2, $N(u) \subseteq N(x) \cup N(x')$. This finishes the proof of Claim 3.

It remains to prove how we can check in polynomial time whether (u, v) is a P_4-suitable pair of G. If (u, v) is a P_4-suitable pair of G, then by definition G has a

P_4-witness structure $\mathcal{W}$ with $W(p_1) = \{u\}$ and $W(p_4) = \{v\}$. Any such witness structure satisfies at least one of the two conditions in Claim 3. We can check in polynomial time if these conditions hold after guessing one vertex (respectively two vertices) in $V \backslash (N(u) \cup N(v) \cup \{u, v\})$. If so, we check in polynomial time if $N(v)$ is contained in a component of the remaining graph (without vertex v). If all our guesses are negative, then (u, v) is not a P_4-suitable pair of G. $\square$

Theorem 7 and Theorem 8 together yield the following result.

Theorem 9. *The* Longest Path Contractibility *problem restricted to the class of P_ℓ-free graphs is polynomially solvable if $\ell \leq 5$ and* NP*-hard if $\ell \geq 6$.*

Proof. First assume $\ell = 5$. Let $G = (V, E)$ be a P_5-free graph. By definition, $\vartheta(G) = 0$ if and only if G is disconnected. Suppose G is connected. Since G does not contain an induced path on more than four vertices, G is clearly not contractible to such a path. Hence we have $\vartheta(G) \leq 4$. By Theorem 8, we can check in polynomial time whether G is P_4-contractible. If so, then $\vartheta(G) = 4$. Otherwise, we check if G has a P_3-suitable pair. This is a necessary and sufficient condition for P_3-contractibility according to Lemma 2. We can perform this check in polynomial time, since two vertices u, v form a P_3-suitable pair of G if and only if u and v are non-adjacent and $G[V \backslash \{u, v\}]$ is connected. If G is P_3-contractible, then $\vartheta(G) = 3$. If G is not P_3-contractible, then we conclude that $\vartheta(G) = 2$ if G has at least two vertices, and $\vartheta(G) = 1$ otherwise.

Now assume $\ell = 6$. Since a graph G is P_4-contractible if and only if $\vartheta(G) \geq 4$ and the P_4-Contractibility problem is NP-complete for P_6-free graphs by Theorem 7, the Longest Path Contractibility problem is NP-hard for P_6-free graphs.

The claim for all other values of ℓ immediately follows from the fact that the class of P_ℓ-free graphs is a subclass of the class of $P_{\ell'}$-free graphs whenever $\ell \leq \ell'$. $\square$

4.2 An Exact Algorithm

Algorithm SPLIT can be extended to an algorithm that solves the Longest Path Contractibility problem for P_6-free graphs in $\mathcal{O}^*(1.5790^n)$ time. Hence we have the following theorem, the proof of which has been omitted due to page restrictions.

Theorem 10. *The* Longest Path Contractibility *problem for P_6-free graphs on n vertices can be solved in $\mathcal{O}^*(1.5790^n)$ time.*

5 Conclusions

We showed that the 2-Disjoint Connected Subgraphs problem is already NP-complete if one of the given sets of vertices has cardinality 2. We also showed that the 2-Disjoint Connected Subgraphs problem for the class of P_ℓ-free graphs jumps from being polynomially solvable to being NP-hard at $\ell = 5$, while for the Longest Path Contractibility problem this jump occurs at $\ell = 6$.

Our algorithm SPLIT solves the 2-Disjoint Connected Subgraphs problem for P_ℓ-free graphs faster than $\mathcal{O}^*(2^n)$ for any ℓ. We do not know yet how to improve its running time for P_5-free and P_6-free graphs (which are in $\mathcal{G}^{1,2}$ and $\mathcal{G}^{4,2}$, respectively) but expect we can do better for P_ℓ-free graphs with $\ell \geq 7$ (by using a radius argument). The modification of SPLIT solves the Longest Path Contractibility problem for P_6-free graphs in $\mathcal{O}^*(1.5790^n)$ time. Furthermore, SPLIT might be modified into an exact algorithm that solves the Longest Path Contractibility problem for P_ℓ-free graphs with $\ell \geq 7$ as well. The most interesting question however is to find a fast exact algorithm for solving the 2-Disjoint Connected Subgraphs and the Longest Path Contractibility problem for general graphs.

Acknowledgements. The authors would like to thank Asaf Levin for fruitful discussions.

References

1. Bacsó, G., Tuza, Z.: Dominating Subgraphs of Small Diameter. Journal of Combinatorics, Information and System Sciences 22(1), 51–62 (1997)
2. Blum, D.: Circularity of Graphs. PhD thesis, Virginia Polytechnic Institute and State University (1982)
3. Brouwer, A.E., Veldman, H.J.: Contractibility and NP-completeness. Journal of Graph Theory 11, 71–79 (1987)
4. Diestel, R.: Graph Theory, 3rd edn. Springer, Heidelberg (2005)
5. Fomin, F.V., Grandoni, F., Kratsch, D.: Solving Connected Dominating Set Faster than 2^n. Algorithmica 52(2), 153–166 (2008)
6. Garey, M.R., Johnson, D.S.: Computers and Intractability. W.H. Freeman and Co., New York (1979)
7. Gerber, M.U., Lozin, V.V.: On the Stable Set Problem in Special P_5-Free Graphs. Discrete Applied Mathematics 125, 215–224 (2003)
8. Hoàng, C.T., Kamiński, M., Lozin, V.V., Sawada, J., Shu, X.: Deciding k-Colorability of P_5-Free Graphs in Polynomial Time. Algorithmica (to appear)
9. van 't Hof, P., Paulusma, D.: A New Characterization of P_6-Free Graphs. Discrete Applied Mathematics (to appear), doi:10.1016/j.dam.2008.08.025
10. Mosca, R.: Stable Sets in Certain P_6-Free Graphs. Discrete Applied Mathematics 92, 177–191 (1999)
11. Randerath, B., Schiermeyer, I.: 3-Colorability $\in$ P for P_6-Free Graphs. Discrete Applied Mathematics 136, 299–313 (2004)
12. Robertson, N., Seymour, P.D.: Graph minors. XIII. The Disjoint Paths Problem. Journal of Combinatorial Theory, Series B 63, 65–110 (1995)
13. Woeginger, G.J., Sgall, J.: The Complexity of Coloring Graphs without Long Induced Paths. Acta Cybernetica 15(1), 107–117 (2001)

Structural Complexity of AvgBPP

Dmitry Itsykson[⋆]

Steklov Institute of Mathematics at St. Petersburg,
27 Fontanka, St. Petersburg 191023, Russia
dmitrits@pdmi.ras.ru

Abstract. We study the class **AvgBPP** that consists of distributional problems which can be solved in average polynomial time (in terms of Levin's average-case complexity) by randomized algorithms with bounded error. We prove that there exists a distributional problem that is complete for **AvgBPP** under polynomial-time samplable distributions. Since we use deterministic reductions, the existence of a deterministic algorithm with average polynomial running time for our problem would imply **AvgP** = **AvgBPP**. Note that, while it is easy to construct a *promise problem* that is complete for **promise-BPP** [Mil01], it is unknown whether **BPP** contains complete *languages*. We also prove a time hierarchy theorem for **AvgBPP** (there are no known time hierarchy theorems for **BPP**). We compare average-case classes with their classical (worst-case) counterparts and show that the inclusions are proper.

1 Introduction

It is unknown whether **BPP** has a time hierarchy or complete problems under deterministic reductions. The main obstacle is the absence of effective enumeration of randomized bounded error Turing machines. Note that if **P** = **BPP**, then **BPP** does have a complete problem since **P** does. However, there is a relativized world where $\mathbf{BPP}^{\mathbf{A}}$ has no complete languages [HH86]. The existence of a **BPP**-complete problem implies a time hierarchy theorem for **BPP** (see for example [Bar02]). The best current result for time hierarchy is superpolynomial: $\mathbf{BPTime}[n^{\log n}] \subsetneq \mathbf{BPTime}[2^{n^{\epsilon}}]$ [KV87]. However, we are not able to prove that $\mathbf{BPTime}[n] \subsetneq \mathbf{BPTime}[n^{100 \log n}]$.

The first advancement in that direction was a time hierarchy theorem for randomized classes with several bits of nonuniform advice [Bar02, FS04], the latest results include a time hierarchy for classes with only one bit of advice : $\mathbf{BPP}/1$ [FS04], $\mathbf{ZPP}/1, \mathbf{MA}/1$, etc. [vMP07]. But the notion of advice used in those results is not standard as the machines can violate the promise if an advice string is incorrect. The second advancement was a time hierarchy for heuristic randomized algorithms (heuristic algorithms may err on a small fraction of inputs). A time hierarchy for the class $\mathbf{Heur}_{\frac{1}{n^c}}\mathbf{BPP}$ (with uniform distributions) was

⋆ Partially supported by Russian Science Support Foundation, grant RFBR 08-01-00640, and the president of Russia grant "Leading Scientific Schools" NSh-4392.2008.1.

proved in [FS04] and simplified in [Per07]. However in these results algorithms not only sometimes give incorrect answers, but also violate the promise on a small fraction of inputs. The ability to make errors on some inputs sometimes helps to prove completeness results, e.g. there exists a complete public-key cryptosystem if a decoding algorithm may err with some small probability [HKN+05] (see also [GHP06]).

In this paper we study the class **AvgBPP** [Imp95] that consists of distributional problems that can be solved in average polynomial time (in terms of Levin's average-case complexity [Lev86]) by randomized algorithms with bounded error. **AvgBPP** corresponds to an adequate model of feasible computations. If $(\mathbf{NP}, \mathbf{PSamp}) \subseteq \mathbf{AvgBPP}$, then there are no one-way functions [BT06].

It this paper we construct a language C and polynomial-time samplable distribution R such that the distributional problem (C, R) is complete for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ under deterministic Turing reductions. Our construction implies that if this problem belongs to $(\mathbf{AvgP}, \mathbf{PSamp})$ (or even to $(\mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}, \mathbf{PSamp})$), then $(\mathbf{AvgBPP}, \mathbf{PSamp})$ equals $(\mathbf{AvgP}, \mathbf{PSamp})$. The same result also holds for $(\mathbf{HeurBPP}, \mathbf{PSamp})$.

We use a modification of the standard complete problem. Language C consists of stings $(M, x, 1^t)$, where M is a randomized Turing machine, x is an input and t is a number of steps, such that M accepts x in t steps with probability at least $\frac{1}{2}$. The polynomial-time sampler (that samples distribution R) tests (by multiple executions) that M accepts (or rejects) x in t steps with probability at least 0.9. If the test fails, the sampler generates a useless string; therefore the R-mesure of "bad" inputs $(M, x, 1^t)$ (that violate bounded error promise) is very small. The key observation is that an average polynomial time algorithm may work exponential time on a very small fraction of inputs. To satisfy the domination condition in the reduction we also include a description of a sampler in instances of the language C.

The constructed distribution R is not uniform and is somewhat unnaturally samplable. We give a intuitive idea why it is very hard to construct a complete problem with uniform (or uniform-like) distribution: we prove that if there exists a complete problem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ with uniform (or uniform-like) distribution, then there exists a partial derandomization of **BPEXP**; namely for all languages $L \in \mathbf{BPEXP}$ the distributional problem (L, U) is solvable by a deterministic algorithm with average exponential running time, where U denotes uniform distribution.

We prove a time hierarchy theorem for the class $(\mathbf{AvgBPP}, \mathbf{PSamp})$. Namely, we prove that for every $c \geq 1$ there exists a language L and polynomial-time samplable distribution D such that $(L, D) \in \mathbf{AvgBPP}$ and $(L, D) \notin \mathbf{AvgBPTime}[n^c]$. Our technique is an extension of the one used in [Per07]; we also use delayed diagonalization as the base of our proof. We modify the multithreshold trick (invented in [Per07]) so that the algorithm holds the bounded error promise on all inputs and the algorithm may work exponential time on small fraction of inputs instead of violating the promise. The weakness of our result is that the distribution D is not uniform. It is an interesting open question to prove the same for uniform D. We compare the classes $\mathbf{AvgP}, \mathbf{AvgBPP}, \mathbf{HeurP}, \mathbf{HeurBPP}$ with their worst-case counterparts and show the following inclusions (for polynomial-time samplable distributions): $\mathbf{P} \subsetneq \mathbf{AvgP} \subseteq \mathbf{HeurP} \subsetneq \mathbf{EXP}$ and $\mathbf{BPP} \subsetneq \mathbf{AvgBPP} \subseteq \mathbf{HeurBPP} \subsetneq \mathbf{BPEXP}$.

Some proofs are omitted due to the space restrictions; please refer to the full version of the paper [Its08] for the details.

2 Preliminaries

We restrict ourselves to the binary alphabet $\{0,1\}$; we denote the set of all binary words as $\{0,1\}^*$. A language is any subset of $\{0,1\}^*$. We identify a language with its characteristic function: $x \in L \iff L(x) = 1$.

An *ensemble of distributions* D is a family of functions $\{D_n\}_{n=1}^{\infty}$, where D_n is a mapping $\{0,1\}^n \to [0,1]$ such that $\sum_{x \in \{0,1\}^n} D(x) = 1$. The set $\{x \in \{0,1\}^n | D_n(x) > 0\}$ is called the support of D_n and is denoted as $\operatorname{supp} D_n$; $\operatorname{supp} D = \cup_{n \in \mathbb{N}} \operatorname{supp} D_n$. *A distributional problem* is a pair (L, D) of a language L and an ensemble of distributions D.

Let $\mathfrak{P}$ be the class of *distributional problems* and $\mathfrak{D}$ be the class of distributions. $(\mathfrak{P}, \mathfrak{D}) = \{(L,D) | (L,D) \in \mathfrak{P}, D \in \mathfrak{D}\}$.

In this paper we consider only polynomial-time samplable distributions. An ensemble of distributions D is called *polynomial-time samplable* if there exists a polynomial-time randomized algorithm (sampler) $\mathcal{S}$ such that the outputs of $\mathcal{S}(1^n)$ are distributed according to D_n. The set of all polynomial-time samplable distributions is denoted by **PSamp**. By uniform distribution U we mean the ensemble of distributions U_n, where U_n is the uniform distribution on $\{0,1\}^n$. In what follows we always mean an ensemble of distributions whenever we use the word distribution.

The first notion of average-case tractability was given by Levin in [Lev86]. A function $t : \{0,1\}^* \times \mathbb{N} \to \mathbb{N}$ (with distribution D) is called *polynomial on the average*[1] if there exists $\epsilon > 0$ such that $\mathrm{E}_{x \leftarrow D_n} t^{\epsilon}(x) = O(n)$. The distributional problem is solvable in average polynomial time if there exists an algorithm that solves it with average polynomial running time.

An equivalent definition of average-case tractability was given by Impagliazzo [Imp95]. A distributional problem (L, D) is solvable in *polynomial on the average* time if there exists an algorithm $\mathcal{A}(x, \delta)$ (following [BT06] we call such algorithm *an errorless heuristic scheme*) that may explicitly "give up" (return $\perp$) so that the following conditions are satisfied: (Effectiveness) The running time of $\mathcal{A}(x, \delta)$ is bounded by $poly(\frac{n}{\delta})$; (Correctness) for all x in the support of D, $\mathcal{A}(x, \delta) \in \{L(x), \perp\}$; (Usefulness) $\Pr_{x \leftarrow D_n}\{\mathcal{A}(x,\delta) = \perp\} < \delta$.

[1] The naive approach is to define that $t(x)$ to be polynomial on the average if the expectation of $t(x)$ is polynomial. But this naive definition is not closed under some natural operations. For example, it is easy to construct $t(x)$ such that the expectation of $t(x)$ is polynomial but the expectation of $t^2(x)$ is exponential (see [BT06]).

A formal proof of the equivalence is given in [Imp95, BT06]. The output $\perp$ in Impagliazzo's definition corresponds to a manual interruption of the algorithm from Levin's definition. The set of all distributional problems which can be solved in average polynomial time is denoted by **AvgP**.

Both these definitions may be extended for bounded error randomized algorithms. We say that randomized algorithm $\mathcal{A}$ solves (L, D) with a bounded error if for all x in the support of D_n $\Pr\{\mathcal{A}(x) \neq L(x)\} < \frac{1}{4}$. In Levin's definition we define the running time of the algorithm $\mathcal{A}$ on the input x as $\min\{t | \Pr\{\mathcal{A}(x) \text{ stops in } t \text{ steps}\} \geq \frac{3}{4}\}$. The Impagliazzo-style definition is as follows:

Definition 1 ([BT06, Definition 2.13]). *The distributional problem (L, D) is solvable in randomized average polynomial time with a bounded error if there exists an algorithm (we call such algorithm as* randomized errorless heuristic scheme*) $A(x, \delta)$ such that the following conditions are satisfied: (Effectiveness) The running time of $\mathcal{A}(x, \delta)$ is bounded by poly$(\frac{n}{\delta})$; (Correctness) for all x in the support of D, $\Pr\{\mathcal{A}(x, \delta) \notin \{L(x), \perp\}\} < \frac{1}{4}$, where the probability is taken over the random bits of algorithm $\mathcal{A}$; (Usefulness) $\Pr_{x \leftarrow D_n}\{\Pr\{\mathcal{A}(x, \delta) = \perp\} \geq \frac{1}{4}\} < \delta$, where the inner probability is taken over the random bits of algorithm $\mathcal{A}$.*

The class **AvgBPP** *consists of all problems that are solvable in randomized average polynomial time with a bounded error.*

Lemma 1 ([BT06]). *The constant $\frac{1}{4}$ used in the correctness and usefulness conditions of Definition 1 is arbitrary and can be replaced by anything in the interval $(2^{-\Omega(n)}, \frac{1}{2})$.*

For every function $g(n)$, we define classes **AvgTime**$[g(n)]$[2] and **AvgBPTime**$[g(n)]$; the definitions are the same as the definitions of **AvgP** and **AvgBPP** but effectiveness conditions are substituted by the following: the running time of $\mathcal{A}(x, \delta)$ is bounded by $O(g(\frac{n}{\delta}))$. We also define class $\mathbf{AvgEXP} = \bigcup_{c>0} \mathbf{AvgTime}[2^{n^c}]$.

Similarly to errorless heuristic schemes it is possible to define general heuristic schemes. In the deterministic case we say that a distributional problem (L, D) is solvable by *a polynomial time heuristic scheme* $A(x, \delta)$ if (Effectiveness) The running of $\mathcal{A}(x, \delta)$ is bounded by *poly*$(\frac{n}{\delta})$; (Usefulness) $\Pr_{x \leftarrow D_n}\{\mathcal{A}(x, \delta) \neq L(x)\} < \delta$.

In the randomized case the usefulness condition is formulated as follows: (Usefulness) $\Pr_{x \leftarrow D_n}\{\Pr\{\mathcal{A}(x, \delta) \neq L(x)\} \geq \frac{1}{4}\} < \delta$, where the inner probability is taken over the random bits of the algorithm $\mathcal{A}$.

The set of all distributional problems solvable by polynomial-time (randomized) heuristic schemes is denoted by **HeurP** and **HeurBPP**.

[2] One may argue that the usage of the notion of average-case time is a controversial point since it uses Impagliazzio-style definition of averge-case tractability. We may also define class $\mathbf{Avg}_{Lev}\mathbf{Time}[n^c]$ as the set of all distributional problems (L, D) that can de solved in time $t(n)$, such that $\mathrm{E}_{x \leftarrow D_n} t^{1/c}(x) = O(n)$. It may be shown that $\mathbf{Avg}_{Lev}\mathbf{Time}[n^c] \subseteq \mathbf{AvgTime}[n^{c+2}] \subseteq \mathbf{Avg}_{Lev}\mathbf{Time}[n^{c+2}]$, therefore our definition is reasonable for our time hierarchy theorem.

We say that a problem (L, D) is solved by a $\delta(n)$*-heuristic algorithm* $\mathcal{A}$ if for every n, it holds that $\Pr_{x \leftarrow D_n}\{\mathcal{A}(x) \neq L(x)\} < \delta(n)$. We say that (L, D) is solved by *an errorless* $\delta(n)$*-heuristic algorithm* if $\mathcal{A}$ is $\delta(n)$-heuristic and $\mathcal{A}(x) \in \{L(x), \perp\}$ for all x and n.

The set of problems solvable by polynomial-time (errorless) δ-heuristic algorithms is denoted by $\mathbf{Heur}_{\delta(n)}\mathbf{P}$ (resp. $\mathbf{Avg}_{\delta(n)}\mathbf{P}$). The classes $\mathbf{Heur}_{\delta(n)}\mathbf{BPP}$ and $\mathbf{Avg}_{\delta(n)}\mathbf{BPP}$ are defined similarly. The classes $\mathbf{Heur}_{\delta(n)}\mathbf{Time}[g(n)]$, $\mathbf{Avg}_{\delta(n)}\mathbf{Time}[g(n)]$, $\mathbf{Heur}_{\delta(n)}\mathbf{BPTime}[g(n)]$, $\mathbf{Avg}_{\delta(n)}\mathbf{BPTime}[g(n)]$, $\mathbf{HeurTime}[g(n)]$, $\mathbf{HeurBPTime}[g(n)]$ are defined in a natural way.

It is easy to see that $\mathbf{AvgP} \subseteq \mathbf{HeurP}$. Indeed it is sufficient to modify the $\mathbf{AvgP}$ algorithm as follows: return 0 instead of $\perp$. A similar modification and Lemma 1 imply $\mathbf{AvgBPP} \subseteq \mathbf{HeurBPP}$. Whether these inclusions are proper or not is an important open question [Imp95].

Now we define *deterministic* Turing reductions between distributional problems. We distinguish errorless and heuristic reductions since average-case classes and heuristic classes use different computational models. Our definition is very similar to [BDCGL92] but here we use an Impagliazzo style definition (and computational model) and ensembles of distributions while [BDCGL92] used Levins definition and distributions on the set of all binary strings.

Definition 2 (cf. [BDCGL92]). *A distributional problem* (L, D) *is* errorlessly reducible *to a problem* (L', D')*, if there exists a* deterministic *algorithm with an oracle* $\mathcal{T}^{L'}(x; \delta)$ *with the following properties: (Effectiveness) running time of* $\mathcal{T}^{L'}(x; \delta)$ *is* $poly(\frac{|x|}{\delta})$*; (Correctness)* $\mathcal{T}^{L'}(x, \delta) \in \{L(x), \perp\}$ *for all* x *in the support of* D*; (Usefulness)* $\Pr_{x \leftarrow D_n}\{\mathcal{T}^{L'}(x, \delta) = \perp\} < \delta$ *for all* $n \in \mathbb{N}$*; (Domination) there exists a polynomial* $p(n)$ *and subset* $E_n \subseteq \{0, 1\}^n$ *of small measure* $D_n(E_n) \leq \delta$ *such that* $\sum_{x \in \{0,1\}^n \setminus E_n} Ask_{\mathcal{T},\delta}(x, y) D_n(x) \leq p(\frac{n}{\delta}) D'(y)$*, where* $Ask_{\mathcal{T},\delta}(x, y) = 1$ *if* $\mathcal{T}^{L'}(x, \delta)$ *asks the oracle for a string* y *and* $Ask_{\mathcal{T},\delta}(x, y) = 0$ *otherwise. Informally speaking,* E_n *is the small set on which* $\mathcal{T}$ *makes "incorrect" queries to the oracle.*

Eliminating the correctness condition and substituting the usefulness condition by $\Pr_{x \leftarrow D_n}\{\mathcal{T}^{L'}(x, \delta) \neq L(x)\} < \delta$ *we get a definition of* heuristic reduction.

The following lemma shows that the reductions defined above are reasonable.

Lemma 2. *(1) If* (L, D) *is errorlessly reducible to* (L', D') *and* $(L', D') \in \mathbf{AvgP}$*, then* $(L, D) \in \mathbf{AvgP}$*. (2) If* (L, D) *is heuristically reducible to* (L', D') *and* $(L', D') \in \mathbf{HeurP}$*, then* $(L, D) \in \mathbf{HeurP}$*.*

3 Complete Problem

In this section we construct a distributional problem that is complete for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ under errorless reductions and is complete for $(\mathbf{HeurBPP}, \mathbf{PSamp})$ under heuristic reductions.

The way tuples are encoded to be an input of an algorithm is important in average-case complexity. Namely, we may use only a logarithmic number of extra bits in encoding, because in this case the uniform probability of a string decreases only polynomially. Now we describe the way we encode tuples:

Remark 1. Let x and y be two strings of bits. One can encode the pair (x, y) as $0^{\lceil \log |x| \rceil} 1 |x|_2 xy$, where $|x|_2$ is the length of the string x written in binary. It is easy to see that $|(x, y)| = |x| + |y| + 2\lceil \log(|x|+1) \rceil + 1$. An m-tuple $z = (x_1, x_2, \dots, x_m)$ can be encoded as $(x_1, (x_2, (x_3, \dots (x_{m-1}, x_m) \dots)$. In this case $|z| = \sum_{i=1}^{m} |x_i| + 2\sum_{i=1}^{m-1} \lceil \log(|x_i|+1) \rceil + m + 1 \le \sum_{i=1}^{m} |x_i| + 2(m-1)\lceil \log(|z| - |x_m| + 1) \rceil + m - 1$.

Proposition 1 (Chernoff-Hoefding bound). *For $X_1, X_2, \dots, X_N$ identically and independently distributed such that $X_i \in [0, 1]$ and $E[X_i] = \mu$, it holds that* $\Pr\{|\frac{\sum_{i=1}^{N} X_i}{N} - \mu| \ge \epsilon\} \le 2e^{-2\epsilon^2 N}$.

We will use the Chernoff bound (Proposition 1) for several times. For this purpose we fix such a number N_0 that $2e^{-\frac{N_0}{1000}} < 0.001$. Each time we apply Chernoff bound, the number of random variables should be at least N_0.

We assume that all Turing machines output only an element from the set $\{0, 1, \perp\}$. Technically we may look on the first 2 bits of the first tape and interpret "00" as 0, "11" as 1, "01" and "10" as $\perp$. Let $M^{\le m}$ denote the Turing machine M which is forcedly interrupted after m steps if it has not reached the final state. The result of $M^{\le m}$ is the first 2 bits of the first tape. Let $\mathrm{freq}(M(x))$ be the most frequent answer from $\{0, 1\}$ returned by randomized Turing machine M on the input x and $\mathrm{prob}(M(x))$ be the probability of $\mathrm{freq}(M(x))$.

We construct a distributional problem (C, R), where C is a language and R is a polynomial-time samplable distribution. The language C will be defined explicitly, the distribution R will be defined by the sampler $\mathcal{R}$. We will show that the distributional problem (C, R) is in **AvgBPP** (and therefore in **HeurBPP**) and that (C, R) is complete for **AvgBPP** under errorless reductions (in a similar way it is possible to prove that the distributional problem (C, R) is complete for **HeurBPP** under heuristic reductions).

All strings of the language C have the form $(M, y, 1^m, b, S, 1^s)$, where M is the encoding of a randomized Turing machine, y is the input of this Turing machine, $m > |y| + N_0$ is the number of steps that M is allowed to do, $b \in \{0, 1\}$ is the answer that would be used instead of an answer $\perp$ of the machine $M^{\le m}$, S is the encoding of a sampler, s is the number of steps that S is allowed to do. The language C is defined by its characteristic function: $C(M, y, 1^m, b, S, 1^s) =$

$$\begin{cases} b, & \text{if } \Pr\{M^{\le m}(y) = \perp\} \ge \frac{1}{4}, \\ \mathrm{freq}(M^{\le m}(y)), & \text{otherwise.} \end{cases}$$

The next Lemma shows that if the mesuare of "bad" machines under the distribution H is small enough, then the problem (C, H) is solvable in **AvgBPP**.

Lemma 3. *Let a distribution H satisfy the following property: for every Turing machine M, if the* $\mathrm{prob}(M^{\le m}(y)) \le 0.85$, *then* $H(M, y, 1^m, b, S, 1^s) \le 2e^{-n^2}$, *where* $n = |(M, y, 1^m, b, S, 1^s)|$. *Then* $(C, H) \in$ **AvgBPP**.

Proof. Consider the following algorithm $\mathcal{A}(x,\delta)$: (1) Test that the input x is a string of the form $(M,y,1^m,b,S,1^s)$, where $m > |y| + N_0$, $b \in \{0,1\}$. If no, then reject. (2a) If $\delta > \frac{1}{2^m}$, then execute the machine $M^{\leq m}(y)$ for $200m^2$ times. If there exists $c \in \{0,1\}$ that appears at least 80% times, then return c, otherwise return $\perp$. (2b) If $\delta \leq \frac{1}{2^m}$, then go through all sequences of random bits of $M^{\leq m}(y)$. If the fraction of answers that are equal to $\perp$ is at least $\frac{1}{4}$, then return b. Otherwise return the most frequent answer from {0,1}. (3) Return $\perp$.

(Correctness) Let $\delta > \frac{1}{2^m}$ (otherwise the algorithm $\mathcal{A}$ works deterministically and always outputs the correct answer). If $\mathrm{prob}(M^{\leq m}(y)) \leq 0.75$, then the Chernoff bound implies $\Pr\{\mathcal{A}((M,y,1^m,b,S,1^s),\delta) = \perp\} \geq 0.99$. Otherwise, if $\mathrm{prob}(M^{\leq m}(y)) > 0.75$, then $\mathrm{freq}(M^{\leq m}(y)) = C(M,y,1^m,b,S,1^s)$. In this case the Chernoff bound implies $\Pr\{\mathcal{A}((M,y,1^m,b,S,1^s),\delta) = 1 - C(M,y,1^m,b,S,1^s)\} \leq 0.01$.

(Usefulness) Let $\delta > \frac{1}{2^m}$ (otherwise the algorithm $\mathcal{A}$ works deterministically and does not output $\perp$). If $\mathrm{prob}(M^{\leq m}(y)) > 0.85$, then the Chernoff bound implies $\Pr\{\mathcal{A}((M,y,1^m,b,S,1^s),\delta) = \perp\} < 0.01$. Otherwise, by the statement of the Lemma, $H(M,y,1^m,b,S,1^s) \leq 2e^{-n^2}$. The total probability of all such inputs Z may be estimated as follows: $H(Z) \leq e^{-n^2}2^{n+1} \leq 2^{-n} < \delta$ (for $n > N_0$). □

We define the distribution R by the sampler $\mathcal{R}$ (this distribution will be used in our complete problem).

Algorithm 1. *The sampler* $\mathcal{R}(1^n)$*:*
(1) Generate a string w of length n. If it is not of the form (M,y,r,b,S,σ), where $b \in \{0,1\}$, then return w.
(2) Execute the sampler $S^{\leq|\sigma|}(1^{|y|})$. Let x denote the result of $S^{\leq|\sigma|}$.
(3) Execute $M^{\leq|r|}(x)$ steps for $200n^2$ times. If each answer from $\{0,1\}$ appears less than 90% times, return 1^n. (Note that by Remark 1 the string 1^n does not encode any tuple.)
(4) Return $(M,x,1^{|r|},b,S,1^{|\sigma|})$.

Lemma 4. *Let the sampler S correspond to a distribution D. Let $z = (M,x,1^m,b,S,1^s)$, $n = |z|$. (1) If $\mathrm{prob}(M^{\leq m}(x)) \geq 0.95$, then $R(z) \geq (1 - 2e^{-n^2})D(x)2^{-10\log(n-s+1)-5} \cdot 2^{-|M|-|S|}$. (2) If $\mathrm{prob}(M^{\leq m}(x)) \leq 0.85$, then $R(z) \leq 2e^{-n^2}$.*

Proof. (1) With probability at least $2^{-10\log(n-|\sigma|+1)-5} \cdot 2^{-|M|-|S|}$ the sampler $\mathcal{R}$ generates the string (M,y,r,b,S,σ) on the first step, where $|y| = |x|, |r| = m, |\sigma| = s, b \in \{0,1\}$. (By the Remark 1, at most $10\log(n-|\sigma|+1)+5$ of bits are used to determine the lengths of the tuple's items). With probability $D(x)$ the sampler S outputs x on the second step of the sampler $\mathcal{R}$. The Chernoff bound implies that the test on the third step of the sampler $\mathcal{R}$ will be passed with probability at least $(1-2e^{-n^2})$.

(2) The Chernoff bound implies that the test on the third step of the sampler $\mathcal{R}$ will be passed with probability at most $2e^{-n^2}$. □

The claim (2) of Lemma 4 implies that the distribution R satisfies the condition of Lemma 3. Therefore we get:

Theorem 1. $(C, R) \in \mathbf{AvgBPP}$

Remark 2. Assume that a Turing machine M has two inputs: a string x and a rational number $\delta \in (0,1)$. Let M_δ be the Turing machine that simulates M with the value of the second parameter being equal to $\frac{1}{\lceil \frac{1}{\delta} \rceil}$. We may encode M_δ as the pair $(M, \lceil \frac{1}{\delta} \rceil)$, where $\lceil \frac{1}{\delta} \rceil$ is written in binary. By Remark 1 $|(M, \lceil \frac{1}{\delta} \rceil)| = |M| + \lceil \log \lceil \frac{1}{\delta} \rceil \rceil + 2\lceil \log |M| \rceil + 1$, and hence $2^{|M_\delta|} \leq 2^{|M|+3} M^2 (\frac{1}{\delta} + 1)$.

Theorem 2. *(C, R) is a complete problem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ under errorless reductions.*

Proof. Let us consider a distributional problem (L, D) from $\mathbf{AvgBPP}$. Let it be solvable by a machine M_δ the running time of which is bounded by polynomial $g(\frac{|x|}{\delta})$. (We assume that both constants in the Definition 1 are decreased to 0.01 by Lemma 1). Let distribution D be generated by a sampler S with running time bounded by the polynomial $q(n)$.

We describe a reduction in terms of Definition 2. The reduction $\mathcal{T}^C(x, \delta)$ makes 2 queries to the oracle: $z_0 = (M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, 0, S, 1^{q(|x|)})$ and $z_1 = (M_\delta, x, 1^{g(\frac{|x|}{\delta})+N_0}, 1, S, 1^{q(|x|)})$. If the answers of the oracle are different, then return $\perp$. Otherwise return the answer of the oracle.

Let us verify all conditions of a reduction: (Effectiveness) follows from the fact that strings z_0 and z_1 have lengths bounded by $poly(\frac{|x|}{\delta})$.

(Correctness) If $C(z_0) \neq C(z_1)$, then $\mathcal{T}^C(x, \delta) = \perp$. If $C(z_0) = C(z_1)$, then $\Pr\{M_\delta(x) = \perp\} < \frac{1}{4}$. By Definition 1 (with the decreased constants) $\Pr\{M_\delta(x) \in \{L(x), \perp\}\} \geq 0.99$, hence $\Pr\{M_\delta(x) = L(x)\} \geq 0.74$. By construction $C(z_0)$ is the most frequent answer of M_δ on input x for $g(\frac{x}{\delta})$ steps, therefore $C(z_0) = L(x)$ and $\mathcal{T}^C(x, \delta) = L(x)$.

(Usefulness) $\Pr_{x \leftarrow D_n}\{\mathcal{T}^C(x, \delta) = \perp\} = \Pr_{x \leftarrow D_n}\{\Pr\{M_\delta(x) = \perp\} \geq \frac{1}{4}\} < \delta$.

(Domination) Let $E_n = \{x \in \{0,1\}^n | \Pr\{M_\delta(x) = \perp\} \geq 0.01\}$. By definition of M_δ we have $D(E_n) < \delta$. If δ is fixed, then x is uniquely determined by z_0 and z_1. By the correctness condition of M_δ for every $x \in \{0,1\}^n$, $\Pr\{M_\delta(x) = 1 - L(x)\} < 0.01$ and for every $x \in \{0,1\}^n \backslash E_n$, $\Pr\{M_\delta(x) = \perp\} < 0.01$. Therefore for every $x \in \{0,1\}^n \setminus E_n$, $\Pr\{M_\delta(x) = L(x)\} > 0.98$. By the claim (1) of Lemma 4 for $x \in \{0,1\}^n \setminus E_n$ we have $R(z_i) \geq 0.99 D(x) 2^{-5\log(n' - q(|x|)+1) - 10} \cdot 2^{-|M_\delta| - |S|}$, where $n' = |z_0| = |z_1|$, $i \in \{0,1\}$. The last inequality proves the domination condition, since $|S|$ is a constant and $2^{|M_\delta|} \leq 2^{|M|+3}(|M|+1)^2(\frac{1}{\delta}+2)$ by Remark 2. □

Theorem 2 and Lemma 2 implies

Corollary 1. $(C, R) \in \mathbf{AvgP} \iff (\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$.

Theorem 3. $(C, R) \in \mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P} \iff (\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$.

Corollary 2. *If* $(\mathbf{AvgBPP}, \mathbf{PSamp}) \subseteq (\mathbf{Avg}_{\frac{1}{n^c}}\mathbf{P}, \mathbf{PSamp})$*, then* $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$*.*

Analogously it is possible to prove the same completeness result for **HeurBPP**. So classes $(\mathbf{AvgBPP}, \mathbf{PSamp})$ and $(\mathbf{HeurBPP}, \mathbf{PSamp})$ have the same complete problem (although under different reductions). In particular, it means that if $(\mathbf{AvgBPP}, \mathbf{PSamp}) \subseteq (\mathbf{HeurP}, \mathbf{PSamp})$, then $(C, R) \in (\mathbf{HeurP}, \mathbf{PSamp})$ and $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$. If $(\mathbf{AvgP}, \mathbf{PSamp}) = (\mathbf{AvgBPP}, \mathbf{PSamp})$, then since $\mathbf{AvgP} \subseteq \mathbf{HeurP}$, we get $(\mathbf{HeurP}, \mathbf{PSamp}) = (\mathbf{HeurBPP}, \mathbf{PSamp})$.

Now we give some intuition why the resulting complete problem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$ is not hard with respect to the uniform distribution, but is hard with respect to somewhat unnatural samplable distribution. We use ideas from [Gur91], where Gurevich shows that the existence of a complete problem in the distributional **NP** with uniform distribution under deterministic reductions implies $\mathbf{EXP} = \mathbf{NEXP}$. Gurevich used this argument as a motivation for the usage of randomized reductions (but a complete problem for **AvgBPP** under randomized reduction is trivial and useless).

Distribution D is called *flat* [Gur91] if there exists $\epsilon > 0$ such that for every $x \in \{0,1\}^*$, $D(x) \leq 2^{-|x|^{\epsilon}}$.

Theorem 4. *If there exists a problem* (L, D) *with flat distribution* D *that is complete for* $(\mathbf{AvgBPP}, \mathbf{PSamp})$ *under errorless reductions, then* $(\mathbf{BPEXP}, U) \subset \mathbf{AvgEXP}$*.*

4 Time Hierarchy Theorem

In this section we extend techniques from [Per07] to prove a time hierarchy for $(\mathbf{AvgBPP}, \mathbf{PSamp})$.

We consider a sequence n_i such that $n_1 = 1$, $n_{i+1} = 2^{2^{n_i}}$. We split all natural numbers into the segments from n_i to $n_i^* = n_{i+1} - 1$. For every randomized Turing machine M we denote by $\widehat{M}$ the machine which on the input x executes $M(x)$ for $200|x|^2$ times and outputs the most frequent answer. (Here we assume that all Turing machines return only one bit). Let M_i be an enumeration of randomized Turing machines with the time bound n^{c+1} where every Turing machine appears infinitely many times. We describe a language L that will be used in the proof of a time hierarchy. On the lengths from the i-th segment L depends on the Turing machine M_i. If $n_i \leq |x| < n_i^*$, then we identify x with the real number $0.x$ between 0 and 1. We define $\theta_x = \frac{1}{2} + (x - \frac{1}{2})\frac{1}{2n^a}$. Let $\pi_n = \Pr_{y \leftarrow U_{n+1}}\{\widehat{M}_i(y) = 1\}$, where the probability is taken over y and the random bits of M_i; $x \in L \iff \pi_n \geq \theta_x$. If $|x| = n_i^*$, then $x \in L \iff \Pr_{y \leftarrow U_{n_i}}\{\widehat{M}_i(y) = 1\} < \frac{1}{2}$, where the probability is taken over y and the random bits of $\widehat{M}_i$.

We introduce the probability distribution D, that will help us to solve the language L in **AvgBPP**. The hardest instance of L is x with $\theta_x \approx \pi_n$. We define distribution D in such a way that such x will have very small probability, therefore **AvgBPP** algorithm will have enough time to solve this instance. We define the distribution D by the following sampler $\mathcal{D}$.

Algorithm 2. *The sampler $\mathcal{D}(1^n)$:*
(1) If $n = n_i^*$, then return $x \leftarrow U_n$;
(2) Let $n_i \leq n < n_i^*$. Execute $\widehat{M_i}$ on a random input of length $n+1$ for $10^6 n^{2a+c+10}$ times and calculate the frequency $\widetilde{\tau}_n$ of answer 1.
(3) We call a string x bad *if $|\theta_x - \widetilde{\tau}_n| < \varepsilon = \frac{1}{100n^a}$ and we call it* good *otherwise.*
(4) Repeat n^{c+4} times: (a) Generate $x \leftarrow U_n$; (b) If x is good, return x.
(5) Return $x \leftarrow U_n$.

Lemma 5. *Let $n_i \leq n < n_i^*$. (1) For all $x \in \{0,1\}^n$ $D(x) \leq 2^{-n}\frac{1}{1-\alpha}$, where $\alpha = \frac{1}{n^a+2}$. (2) If $|\theta_x - \pi_n| \leq \frac{1}{2^n}$, then $D(x) \leq 2^{-n^{c+2}}$.*

Proof. (1) The probability that a uniformly generated random string is bad is less than $2\varepsilon = \frac{1}{50n^a} < \alpha = \frac{1}{n^a+2}$. The probability that a string x is generated in the first iteration of step 4 of the sampler $\mathcal{D}$ is 2^{-n}; the probability that x is generated on the second iteration is less then $\alpha 2^{-n}$, and so on. Therefore $D(x) \leq 2^{-n}(1+\alpha+\alpha^2+\dots) \leq 2^{-n}\frac{1}{1-\alpha}$. (2) The Chernoff bound implies that $\Pr\{|\pi_n - \widetilde{\tau}_n| \leq \frac{\varepsilon}{2}\} \geq 1 - 2e^{-2n^{c+10}} \geq 1 - 2^{-n^{c+3}}$.

$$\begin{aligned}
D(x) = \Pr\{x = \mathcal{D}(1^n)\} &= \Pr\{x = \mathcal{D}(1^n) |\ |\pi_n - \widetilde{\tau}_n| \leq \frac{\varepsilon}{2}\}\Pr\{|\pi_n - \widetilde{\tau}_n| \leq \frac{\varepsilon}{2}\} \\
&+ \Pr\{x = \mathcal{D}(1^n) |\ |\pi_n - \widetilde{\tau}_n| > \frac{\varepsilon}{2}\}\Pr\{|\pi_n - \widetilde{\tau}_n| > \frac{\varepsilon}{2}\} \\
&\leq \Pr\{x = \mathcal{D}(1^n) |\ |\pi_n - \widetilde{\tau}_n| \leq \frac{\varepsilon}{2}\} + \Pr\{|\pi_n - \widetilde{\tau}_n| > \frac{\varepsilon}{2}\} \\
&\leq \Pr\{x = \mathcal{D}(1^n) |\ x \text{ is bad}\} + 2^{-n^{c+3}} \leq 2^{-n^{c+4}} + 2^{-n^{c+3}} < 2^{-n^{c+2}}. \quad \square
\end{aligned}$$

Theorem 5. $(L, D) \in$ **AvgBPP**

Proof. We show that the distributional problem (L, D) is solvable by $\mathcal{L}(x, \delta)$ in **AvgBPP**: (1) If $|x| = n_i^*$, then execute $\widehat{M_i}$ on all inputs of length n_i and with all sequences of random bits and return the most infrequent answer. Now we have $n_i \leq |x| < n_i^*$.

(2) If $\delta \geq 2^{-n^{c+2}}$ (a) If $n_i \leq |x| < n_i^*$, then execute $\widehat{M_i}$ on $K = 4096\frac{n^{3a}}{\delta^2}$ random inputs of length $n+1$ and compute $\widetilde{\pi}_n$ that is the frequency of the answer 1. (b) If $\theta_x \geq \widetilde{\pi}_n + \frac{\delta}{32n^a}$, return 0. (c) If $\theta_x \leq \widetilde{\pi}_n - \frac{\delta}{32n^a}$, return 1. (d) Return $\perp$.

(3) If $\delta < 2^{-n^{c+2}}$, then compute π_n deterministically. That is, execute $\widehat{M_n}$ on all inputs of length $n+1$ with all sequences of random bits. If $\theta_x \leq \pi_n$, return 1, otherwise return 0.

The running time of $\mathcal{L}(x, \delta)$ is $poly(\frac{|x|}{|\delta|})$. If $\theta_x \geq \pi_n + \frac{\delta}{16n^a}$, then the Chernoff bound implies that with probability at least 0.99 (for large enough n) $\mathcal{L}(x, \delta)$ outputs $0 = L(x)$. If $\theta_x \leq \pi_n - \frac{\delta}{16n^a}$, then the Chernoff bound implies that with probability at least 0.99 (for large enough n) $\mathcal{L}(x,\delta)$ outputs $1 = L(x)$. If $\theta_x \geq \pi_n$, then by the Chernoff bound the probability of the answer 1 is at most 0.01 and if $\theta_x \leq \pi_n$, then the probability of the answer 0 is at most 0.01.

If $\delta > \frac{1}{2^{n-1}}$, then the probability of the answer $\perp$ may be estimated as: $\Pr_{x\leftarrow D_n}\{\Pr\{\mathcal{L}(x,\delta) =\perp\} > \frac{1}{4}\} \leq \Pr_{x\leftarrow D_n}\{\pi_n - \frac{\delta}{16n^a} < \theta_x < \pi_n + \frac{\delta}{16n^a}\} \leq \frac{1}{1-\alpha}\Pr_{x\leftarrow U_n}\{\pi_n - \frac{\delta}{16n^a} < \theta_x < \pi_n + \frac{\delta}{16n^a}\} \leq \frac{1}{1-\alpha}\frac{1+\frac{\delta}{8n^a}2^n\cdot 2n^a}{2^n} = \frac{n^a+2}{n^a+1}(\frac{1}{2^n}+\frac{\delta}{4}) < \delta$. If $\frac{1}{2^{n^{c+2}}} < \delta \leq \frac{1}{2^{n-1}}$, then the $\frac{\delta}{16n^a}$-neighbourhood of π_n contains at most one number θ_x and by the claim (2) of Lemma 5 $D(x) < 2^{-n^{c+2}} < \delta$. If $\delta \leq \frac{1}{2^{n^{c+2}}}$, then $\mathcal{L}(x,\delta) \neq \perp$. □

Theorem 6. $(L,D) \notin \mathbf{Heur}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c]$.

Proof. Proof by contradiction. Suppose that a problem (L,D) is solvable by a Turing machine M_k in $\mathbf{Heur}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c]$. The claim (1) of Lemma 5 implies that for every subset $S \subseteq \{0,1\}^n$, $D(S) \leq \frac{U(S)}{1-\alpha} = \frac{(n^a+2)U(S)}{n^a+1}$. Hence the machine M_k correctly solves L on a set of inputs with uniform measure at least $(\frac{1}{2}+\frac{1}{n^a})\frac{n^a+1}{n^a+2} = (\frac{1}{2}+\frac{1}{2n^a})$.

Let $n = n_k^* - 1$ and $L(x) = b \in \{0,1\}$ for every $x \in \{0,1\}^{n_k^*}$. Since M_k solves L on $\frac{1}{2}+\frac{1}{2n^a}$ fraction of inputs, and machine $\widehat{M_k}$ has probability of error at most $e^{-(n+1)^2}$ on such inputs, we may conclude that $\Pr_{x\leftarrow U_{n+1}}\{\widehat{M_k}(x) = b\} \geq (\frac{1}{2}+\frac{1}{2(n+1)^a})(1-\frac{1}{e^{(n+1)^2}}) > (\frac{1}{2}+\frac{1}{4n^a})$. However $\theta_x \in (\frac{1}{2}-\frac{1}{4n^a}, \frac{1}{2}+\frac{1}{4n^a})$, therefore for every $x \in \{0,1\}^n$, $L(x) = b$. If we continue this reasoning we get that for every $n_k \leq n \leq n_k^*$ and $x \in \{0,1\}^n$, $L(x) = b$. Hence b is the most frequent answer of $\widehat{M_k}$ on $x \in \{0,1\}^{n_k}$; it contradicts with the choice of b. □

Note, that $\mathbf{Heur}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c] \supseteq \mathbf{Avg}_{\frac{1}{2}-\frac{1}{n^a}}\mathbf{BPTime}[n^c] \supseteq \mathbf{AvgBPTime}[n^c]$. It completes the proof of the time hierarchy theorem for $(\mathbf{AvgBPP}, \mathbf{PSamp})$.

5 Worst-Case vs. Average-Case Classes

Definition 3. *Let $\mathfrak{C}$ be the class of* languages *and $\mathfrak{D}$ be the class of distributions.* $(\mathfrak{C},\mathfrak{D}) = \{(L,D) | D \in \mathfrak{D}, \exists L' \in \mathfrak{C} : \forall x \in \operatorname{supp} D\ L(x) = L'(x)\}$.

Theorem 7. *The following inclusions hold: (1)* $(\mathbf{P}, U) \subsetneq (\mathbf{AvgP}, U) \subseteq (\mathbf{HeurP}, U)$*; (2)* $(\mathbf{HeurP}, \mathbf{PSamp}) \subseteq (\mathbf{EXP}, \mathbf{PSamp})$*; (3) There exists language* $L_{EXP} \in \mathbf{EXP}$ *such that for any distribution* $D \in \mathbf{PSamp}$*, distributional problem* (L_{EXP}, D) *is not contained in* $(\mathbf{HeurP}, \mathbf{PSamp})$*.*

And we have a similar theorem for randomized classes:

Theorem 8. *The following inclusions hold: (1)* $(\mathbf{BPP}, U) \subsetneq (\mathbf{AvgBPP}, U) \subseteq (\mathbf{HeurBPP}, U)$*; (2)* $(\mathbf{HeurBPP}, \mathbf{PSamp}) \subseteq (\mathbf{BPEXP}, \mathbf{PSamp})$*; (3) There exists* $L \in \mathbf{BPEXP}$ *such that for every distribution* $D \in \mathbf{PSamp}$*, the distributional problem* (L, D) *is not contained in* $(\mathbf{HeurBPP}, \mathbf{PSamp})$*.*

Corollary 3. *Classes* $\mathbf{AvgP}, \mathbf{HeurP}, \mathbf{AvgBPP}$ *and* $\mathbf{HeurBPP}$ *are not closed under changing the distribution.*

Acknowledgments. The author thanks Dima Grigoriev for bringing this problem to his attention and Edward A. Hirsch for fruitful discussions. The author also thanks Alexei Pastor, Ilya Posov, Kirill Shmakov and Olga Sergeeva for useful comments.

References

[Bar02] Barak, B.: A probabilistic-time hierarchy theorem for slightly non-uniform algorithms. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 194–208. Springer, Heidelberg (2002)

[BDCGL92] Ben-David, S., Chor, B., Goldreich, O., Luby, M.: On the theory of average case complexity. J. Comput. Syst. Sci. 44(2), 193–219 (1992)

[BT06] Bogdanov, A., Trevisan, L.: Average-case complexity. Foundation and Trends in Theoretical Computer Science 2(1), 1–106 (2006)

[FS04] Fortnow, L., Santhanam, R.: Hierarchy theorems for probabilistic polynomial time. In: FOCS, pp. 316–324 (2004)

[GHP06] Grigoriev, D., Hirsch, E.A., Pervyshev, K.: A complete public-key cryptosystem. Technical Report 06-046, Electronic Colloquium on Computational Complexity (2006)

[Gur91] Gurevich, Y.: Average case complexity. In: ICALP, pp. 615–628 (1991)

[HH86] Hartmanis, J., Hemachandra, L.A.: Complexity classes without machines: On complete languages for up. In: Kott, L. (ed.) ICALP 1986. LNCS, vol. 226, pp. 123–135. Springer, Heidelberg (1986)

[HKN$^+$05] Harnik, D., Kilian, J., Naor, M., Reingold, O., Rosen, A.: On robust combiners for oblivious transfer and other primitives. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 96–113. Springer, Heidelberg (2005)

[Imp95] Impagliazzo, R.: A personal view of average-case complexity. In: SCT 1995: Proceedings of the 10th Annual Structure in Complexity Theory Conference (SCT 1995), Washington, DC, USA, p. 134. IEEE Computer Society, Los Alamitos (1995)

[Its08] Itsykson, D.M.: Structural complexity of AvgBPP. Technical Report 08-073, Electronic Colloquium on Computational Complexity (2008)

[KV87] Karpinski, M., Verbeek, R.: Randomness, provability, and the separation of Monte Carlo time and space, pp. 189–207. Springer, London (1987)

[Lev86] Levin, L.: Average case complete problems. SIAM Journal on Computing 15(1), 285–286 (1986)

[Mil01] Miltersen, P.B.: Handbook on Randomization, ch. 19. Derandomizing Complexity Classes, vol. II, ch. 19. Kluwer Academic Publishers, Dordrecht (2001)

[Per07] Pervyshev, K.: On heuristic time hierarchies. In: IEEE Conference on Computational Complexity, pp. 347–358 (2007)

[vMP07] van Melkebeek, D., Pervyshev, K.: A generic time hierarchy with one bit of advice. Computational Complexity 16(2), 139–179 (2007)

Lower Bounds for the Determinantal Complexity of Explicit Low Degree Polynomials

Maurice Jansen

Center for Theory in Natural Sciences, Aarhus University,
Department of Computer Science, Aarhus University, IT-Parken, Aabogade 34,
DK-8200 Aarhus N, Denmark, phone: +45 8942 5600
mjjansen@daimi.au.dk

Abstract. Asymptotically tight lower bounds are proven for the determinantal complexity of the elementary symmetric polynomial S_n^d of degree d in n variables, $2d$-fold iterated matrix multiplication of the form $\langle u|\, X^1X^2 \dots X^{2d}\, |v\rangle$, and the symmetric power sum polynomial $\sum_{i=1}^n x_i^d$, for any constant $d > 1$.

A restriction of determinantal computation is considered in which the underlying affine linear map must satisfy a rank lowerability property. In this model strongly nonlinear and exponential lower bounds are proven for several polynomial families. For example, for S_n^{2d} it is proved that the determinantal complexity using so-called r-lowerable maps is $\Omega(n^{d/(2d-r)})$, for constants d and r with $2 \leq d+1 \leq r < 2d$. In the most restrictive setting an $n^{\Omega(\epsilon n^{1/5-\epsilon})}$ lower bound is observed, for any $\epsilon \in (0, 1/5)$ and $d = \lfloor n^{1/5-\epsilon} \rfloor$.

Keywords: Computational complexity, arithmetical circuits, determinant, permanent, elementary symmetric polynomial.

1 Introduction

The main open problem in algebraic complexity theory is the resolution of Valiant's Hypothesis, which states that the complexity classes VP and VNP are distinct. The question is attractive to study as a first stepping stone towards the P versus NP conundrum, as in this area algebraic tools are more readily available. Over the field of complex numbers VP $\neq$ VNP is known to be implied by NP $\not\subseteq$ P/*poly* , provided the *Generalized Riemann Hypothesis* is true [1]. Currently, we do not know of a reverse implication.

The complexity class VNP is characterized by the permanent polynomial $per_n = \sum_\sigma \prod_{i\in[n]} x_{i\sigma(i)}$, over fields of characteristic other than two [2], cf. [3]. Here the summation is over all permutations σ of $[n]$. For comparison, we have the determinant polynomial defined by $det_n = \sum_\sigma \mathrm{sgn}(\sigma) \prod_{i\in[n]} x_{i\sigma(i)}$, where $\mathrm{sgn}(\sigma)$ denotes the *signature* of the permutation σ. Resolution of Valiant's Hypothesis is tantamount to proving *determinantal complexity* lower bounds. The determinantal complexity of a polynomial $f(x)$, denoted by $\mathrm{dc}(f)$, is the minimum m such that $f = det_m(L(x))$, where $L(x)$ is a matrix whose entries are affine linear forms in x. Proving $\mathrm{dc}(per_n) = n^{\omega(\log n)}$ is known to be sufficient to separate VP from VNP. The plausibility of this condition is fueled by the fact that computing the permanent of a $0,1$-matrix over the integers is #P-complete [4]. By Toda's Theorem [5] this implies the permanent is PH-hard. On the other hand, integer determinants can be computed with NC_2-circuits.

Currently the best known lower bound on determinantal complexity of the permanent is $\mathrm{dc}(per_n) \geq n^2/2$, due to Mignon and Ressayre [6] (cf. [7]). Their lower bound technique is a dimension argument employing second order partial derivatives. Using partial derivatives of order at most two limits the technique to proving lower bounds that are *linear* in the number of input variables. In this paper the question is investigated whether any stronger results can be obtained by considering higher than second order partial derivatives. Such higher order considerations have proven to be instrumental in previous landmark work in algebraic complexity theory [8,9,10,11,12].

The fundamental observation, first made by Valiant, is that determinantal complexity minorizes arithmetical formula size up to constant factors, i.e. $\mathrm{dc}(f) \leq 2L(f) + 2$ [2], cf. [13]. As a matter of fact, $\mathrm{dc}(f) = O(B(f))$, where $B(f)$ denotes algebraic branching program size of f (See [8]). In the latter model, currently no nonlinear lower bounds are known beyond the trivial bound $B(f) = \Omega(\deg(f))$ and the geometric degree bounds of Baur-Strassen [14]. The later bounds are of level $\Omega(n \log deg(f))$, and are established for general arithmetical circuits. For explicit f of "reasonable" degree, proving strongly nonlinear lower bounds for $B(f)$, and even more so for $\mathrm{dc}(f)$, is a major open problem.

In this paper the first aim is to investigate under what additional restrictions to the determinantal model we can achieve above goal for $\mathrm{dc}(f)$, and in doing so to build tools that may assist in resolving the general case. It is shown the Mignon-Ressayre technique can be generalized to higher order partial derivatives, and this relates the technique to work of Nisan on algebraic branching programs over non-commuting variables [8] and Raz on multilinear formulas [11,12]. This will be applied to a version of determinantal complexity, in which the affine map L is restricted to be so-called *r-lowerable*. This condition stipulates that there exists a point a, such that $rank\ L(a) \leq m - r$.

For low degree polynomials this will yield strongly nonlinear and exponential lower bounds. For example, for the elementary symmetric polynomial of degree d in n variables, defined by $S_n^d(X) = \sum_{I \subset [n], |I| = d} \prod_{i \in I} x_i$, it will be shown that the determinantal complexity of S_n^{2d} while using r-lowerable maps is $\Omega(n^{d/(2d-r)})$, for any constants d and r with $2 < d+1 \leq r < 2d$. In the most restrictive setting, for $d = \lfloor n^{1/5-\epsilon} \rfloor$, an exponential lower bound of level $n^{\Omega(\epsilon n^{1/5-\epsilon})}$ is observed for S_n^{2d}, for any constant $\epsilon \in (0, 1/5)$.

The second aim of this paper is to consider unrestricted determinantal complexity of several important polynomial families. For example, it will be shown that $\mathrm{dc}(S_{2n}^{2d-1}) \geq n$ and $\mathrm{dc}(S_{2n+2}^{2d}) \geq n/2$, for $d > 1$, over fields of characteristic zero. For constant d, this determines the determinantal complexity of S_n^d up to a constant factor, since $L(S_n^d) = O(nd^3 \log d)$ [10].

2 Preliminaries

For integer n, $[n]$ denotes $\{1, 2, \ldots, n\}$. Let F be a field. Let $X = \{x_1, \ldots, x_N\}$ and $Y = \{y_1, \ldots, y_M\}$ be sets of variables. Let polynomials $f \in F[X]$ and $g \in F[Y]$ be given. For a vector $r = (r_1, r_2, \ldots, r_M)^T \in F[X]^M$ denote by $g(r)$ the polynomial obtained by substitution of y_i by r_i in g, for all $1 \leq i \leq M$. For a matrix G whose entries are elements in $F[Y]$, let $G(r)$ denote the matrix that has ij-th entry $G(r)_{ij} = G_{ij}(r)$. We generalize Mignon and Ressayre's framework [6] for proving determinantal complexity lower bounds by considering arbitrary order partial derivatives. Their results are obtained by setting $k = 1$ in the following suite of results (Lemma 1, Proposition 1, and Lemma 2).

2.1 Partial Derivatives Matrix

Let $k \geq 1$ be an integer. Define the *partial derivatives matrix* of a polynomial f of order $2k$, denoted by $T^{2k}f$, to be an $N^k \times N^k$ matrix of *formal partial derivatives*, with rows and columns indexed by k-tuples $v, w \in X^k$, respectively, where

$$(T^{2k}f)_{v,w} = \frac{\partial^{2k} f}{\partial v \partial w},$$

where ∂v is shorthand for $\partial v_1 \partial v_2 \ldots \partial v_k$. In this paper $(i_1, i_2, \ldots, i_k)$ is used as a shorthand for the index of a row or column given by the k-tuple $(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$. In expressions, taking derivatives is given precedence over substitution, e.g. $T^{2k}g\ (L(x))$ means $(T^{2k}g)(L(x))$.

Lemma 1. *Let $k \geq 1$ be an integer. Let C be an $M \times N$ matrix with entries from F. Let $c = (c_1, c_2, \ldots, c_M)^T \in F^M$. Let $x = (x_1, x_2, \ldots, x_N)^T$ be a vector of variables. Suppose for M-vector $L(x) = Cx + c$ of affine linear forms that $f = g(L(x))$. Then*

$$T^{2k}f = (C^T)^{\otimes k} \cdot \left(T^{2k}g\ (L(x))\right) \cdot C^{\otimes k},$$

and hence for any $a \in F^n$,

$$\mathit{rank}\ T^{2k}f\ (a) \leq \mathit{rank}\ T^{2k}g\ (L(a)).$$

Proof. Lemma 1 is proved by induction on k. The basis $k = 1$ is given by Lemma 3.1 in [6], and is obtained using the chain rule as follows: $\frac{\partial f}{\partial x_i} = \sum_{k=1}^M \frac{\partial g}{\partial y_k}(L(x)) \cdot \frac{\partial (L(x))_k}{\partial x_i} = \sum_{k=1}^M \frac{\partial g}{\partial y_k}(L(x)) \cdot C_{ki}$. Hence $(T^2f)_{ij} = \frac{\partial^2 f}{\partial x_i \partial x_j} = \sum_{k=1}^M \sum_{l=1}^M \frac{\partial^2 g}{\partial y_k \partial y_l}(L(x)) \cdot \frac{\partial (L(x))_l}{\partial x_j} \cdot C_{ki} = \sum_{k=1}^M \sum_{l=1}^M \frac{\partial^2 g}{\partial y_k \partial y_l}(L(x)) \cdot C_{lj} C_{ki} = \sum_{k=1}^M \sum_{l=1}^M \left(T^2 g\ (L(x))\right)_{kl} \cdot C_{lj} C_{ki} = \left(C^T \cdot T^2 g\ (L(x)) \cdot C\right)_{ij}$.

Now assume the statement of the lemma is true for a $k \geq 1$. Let $(v, i)(w, j) \in [N]^{k+1}$. We have that

$$(T^{2k+2}f)_{(v,i)(w,j)} = \frac{\partial^2 (T^{2k}f)_{vw}}{\partial x_i \partial x_j}$$

$$= \frac{\partial^2 \left[(C^T)^{\otimes k} \cdot \left(T^{2k} g\ (L(x))\right) \cdot C^{\otimes k}\right]_{vw}}{\partial x_i \partial x_j}$$

$$= \sum_{s,t \in [N]^k} [(C^T)^{\otimes k}]_{vs} \cdot \frac{\partial^2 \left[T^{2k} g\ (L(x))\right]_{st}}{\partial x_i \partial x_j} \cdot [C^{\otimes k}]_{tw}. \qquad (1)$$

Let $h_{st} = [T^{2k} g\ (L(x))]_{st}$. We have $h_{st} = [T^{2k} g]_{st}(L(x))$. By the base case

$$\frac{\partial^2 h_{st}}{\partial x_i \partial x_j} = \left[C^T \cdot \left[T^2[T^{2k} g]_{st}\right]\ (L(x)) \cdot C\right]_{ij}$$
$$= \sum_{k,l \in [N]} C^T_{ik} \cdot \left[\left[T^2[T^{2k} g]_{st}\right]\ (L(x))\right]_{kl} \cdot C_{lj}$$
$$= \sum_{k,l \in [N]} C^T_{ik} \cdot \left[T^{2k+2} g\ (L(x))\right]_{(s,k)(t,l)} \cdot C_{lj}.$$

Substituting this into Expression 1, we get

$$(T^{2k+2} f)_{(v,i)(w,j)} =$$
$$\sum_{s,t \in [N]^k} \sum_{k,l \in [N]} [(C^T)^{\otimes k}]_{vs} C^T_{ik} \cdot \left[T^{2k+2} g\ (L(x))\right]_{(s,k)(t,l)} \cdot C_{lj} [C^{\otimes k}]_{tw}$$
$$\sum_{s,t \in [N]^k} \sum_{k,l \in [N]} [(C^T)^{\otimes (k+1)}]_{(v,i)(s,k)} \cdot \left[T^{2k+2} g\ (L(x))\right]_{(s,k)(t,l)} \cdot C^{\otimes (k+1)}]_{(t,l)(w,j)}.$$

We conclude that $T^{2k+2} f = (C^T)^{\otimes (k+1)} \cdot T^{2k+2} g\ (L(x)) \cdot C^{\otimes (k+1)}$. □

2.2 Rank Deficiency of the Determinant

Proposition 1. *Let $k \geq 1$ be an integer. Let A and C be invertible $m \times m$ matrices. Then for any $m \times m$ matrix B,*

$$rank\ \ T^{2k} det_m(B) = rank\ \ T^{2k} det_m\ (ABC).$$

Proof. Let X be an $n \times n$ matrix with variables with ij-th entry equal to x_{ij}. The linear map $L : X \mapsto AXC$ is invertible. So if we let $f(X) = det_m(L(X))$, we have by Lemma 1 that $rank\ \ T^{2k} f\ (B) = rank\ \ T^{2k} det_m\ (L(B))$. However, $f(X) = det_m(AXC) = \mu det_m(X)$, where $\mu = det(A) det(C) \neq 0$. Hence $T^{2k} f = \mu \cdot T^{2k} det_m$. We conclude $rank\ \ T^{2k} det_m\ (B) = rank\ \ T^{2k} f\ (B) = rank\ \ T^{2k} det_m\ (L(B))$. □

Lemma 2. *Let $2k \geq r \geq 1$ be integers. Suppose B is an $m \times m$ matrix of rank at most $m - r$. Then $rank\ \ T^{2k} det_m\ (B) \leq ((2k)!/(2k-r)!)^2 m^{2k-r}$.*

Proof. By Proposition 1, we can assume wlog. that B equals $\text{diag}(0, 0, \ldots, 0, 1, 1, \ldots, 1)$, where there are r zeroes on the diagonal. Let $H = T^{2k} det_m$. For $i, j, s, t \in [m]^k$, on row (i,j) and column (s,t) of H we have

$$\frac{\partial^{2k} det_m}{\partial x_{i_1 j_1} \partial x_{i_2 j_2} \ldots \partial x_{i_k j_k} \cdot \partial x_{s_1 t_1} \partial x_{s_2 t_2} \ldots \partial x_{s_k t_k}}$$

For this entry to be nonzero when evaluated at B we must have that both $[r] \subseteq \{i_1, i_2, \ldots, i_k, s_1, s_2, \ldots, s_k\}$ and $[r] \subseteq \{j_1, j_2, \ldots, j_k, t_1, t_2, \ldots, t_k\}$. There are $((2k)!/(2k-r)!)^2$ ways of fixing particular indices to be $1, 2, \ldots, r$. Once fixed we are left with $2k-r$ variables that index rows and $2k-r$ variables that index columns. These take values in the range $[m]$. Hence each choice of fixing values gives rise to a submatrix of rank at most m^{2k-r}. Observe this implies we can write H as a sum of $((2k)!/(2k-r)!)^2$ matrices each with rank bounded by m^{2k-r}. Hence $rank\ H \leq ((2k)!/(2k-r)!)^2 m^{2k-r}$. $\square$

Note in [6] it is proved that $rank\ \ T^2 det_m\ (B) \leq 2m$, for singular B. The following lemma and its proof are similar to Proposition 3.3 in [11].

Lemma 3. *Let f be homogeneous of degree $2d$ in variables $X = \{x_1, x_2, \ldots, x_n\}$, and let g be homogeneous of degree $2e$ in variables $Y = \{y_1, y_2, \ldots, y_m\}$. Then $rank\ T^{2(d+e)} fg \geq rank\ T^{2d} f \cdot rank\ T^{2e} g$.*

Proof. Let $H = T^{2(d+e)} fg$. The rows and columns of H are indexed by $(d+e)$-sequences v and w of in variables from $X \cup Y$. $H_{v,w}$ equals the coefficient of the monomial that is the product of the entries of (v, w) in fg. We will only consider a minor H' of H, where v (and similarly w) consists of d X-variables and e Y-variables. Let $s = n^d$ and $t = n^e$. Enumerate all d-sequences in X: $a_1, a_2, \ldots, a_s$. Enumerate all e-sequences in Y: $b_1, b_2, \ldots, b_t$. Consider the minor with rows and columns $(a_1, b_1), (a_1, b_2), \ldots, (a_1, b_t)$, $(a_2, b_1), (a_2, b_2), \ldots, (a_2, b_t), \ldots, (a_s, b_1), (a_s, b_2), \ldots, (a_s, b_t)$. Observe that entry $H'_{(a_i,b_j),(a_p,b_q)} = (T^{2d} f)_{(a_i,a_p)} (T^{2e} g)_{(b_j,b_q)}$. Hence $H' = (T^{2d} f) \otimes (T^{2e} g)$, and the lemma follows. $\square$

3 The Mignon-Ressayre Bound

Mignon and Ressayre's lower bound proof for the permanent uses the special case where $k = 1$ in the framework of Section 2. The proof strategy can be outlined as follows. Suppose $f(x) = det_m(L(x))$, for some affine linear map L. The objective is to find a point $a \in F^n$ that simultaneously *minimizes* $rank\ T^{2k} det_m(L(a))$ and *maximizes* $rank\ T^{2k} f(a)$. For points a such that $f(a) = 0$, $L(a)$ must be singular, in which case Lemma 2 yields $rank\ \ T^2 det_m\ (L(a)) \leq 2m$. By Lemma 1, for any a, $rank\ T^2 f(a) \leq rank\ T^2 det_m(L(a))$. For $f = per_n$ it is possible to find a with $per(a) = 0$ and $rank\ T^2 per_n(a)$ at the maximum value n^2, implying $\mathrm{dc}(per_n) \geq n^2/2$.

4 *r*-Determinantal Complexity

Results in this section hold for any choice of the underlying field F.

Definition 1. *Call a map $L : F^n \to M_m(F)$* r-lowerable *if there exists $a \in F^n$ such that $rank\ L(a) \leq m - r$. We define the* r-determinantal complexity *of a polynomial f, denoted by $dc_r(f)$, to be the minimum m for which $f(x) = det_m(L(x))$, where L is an r-lowerable affine linear map.*

For any $r \geq 0$, $\mathrm{dc}_r(f) \leq \mathrm{dc}_{r+1}(f)$. Note that $\mathrm{dc}(f) = \mathrm{dc}_0(f)$, and if $f^{-1}(0) \neq \emptyset$, then $\mathrm{dc}(f) = \mathrm{dc}_1(f)$.

It should be remarked that the affine maps obtained by the constructions that show universality of the determinant of [2,13] will not be 2-lowerable, since these constructions create an upper triangular minor in $L(x)$ of size $m-1$. As a matter of fact, universality fails in the r-determinantal model for *inhomogeneous* polynomials, for $r > 2$. To give an example, let us consider computing a polynomial f of degree four using a 3-lowerable map, where f is of the form:

$$f = x_1x_2(x_3x_4) + x_5x_6(x_3x_4 - 1) + g(x_7, x_8, \ldots, x_n).$$

Assume $f(x) = det_m(L(x))$, where for some a, $rank\ L(a) \leq m-3$. We have that all $(m-2) \times (m-2)$ minors of $L(a)$ are singular. Hence all 2nd order partial derivatives of det_m vanish at $L(a)$. Since the second order partial derivatives of $f(x)$ are in the linear span of those of det_m (See e.g. the proof of Lemma 1), we have that all second order partial derivatives of $f(x)$ vanish at a. For the given f, $\partial^2 f/\partial x_1 \partial x_2 = x_3x_4$ and $\partial^2 f/\partial x_5 \partial x_6 = x_3x_4 - 1$, which clearly cannot simultaneously vanish.

Whether universality fails for *homogeneous* polynomials in the r-determinantal model is an open problem, for $1 < r < deg(f)$. We do however have the following general lower bound theorem:

Theorem 1. *For any polynomial f of degree $2d$ and integer r such that $1 \leq r < 2d$, $dc_r(f) \geq \left(\frac{(2d-r)!}{(2d)!}\right)^{2/(2d-r)} \left(rank\ T^{2d}f\right)^{1/(2d-r)}$.*

Proof. Suppose we can write $f(X) = det_m(L(X))$, where L is r-lowerable. Since f is of degree $2d$, for any $a \in F^n$, $rank\ T^{2d}f(a) = rank\ T^{2d}f$. Fix arbitrary a such that $L(a)$ has rank at most $m-r$. By Lemma 2, $rank\ T^{2d}Det_m(L(a)) \leq ((2d)!/(2d-r)!)^2 m^{2d-r}$. However, by Lemma 1, $rank\ T^{2d}f(a) \leq rank\ T^{2d}Det_m(L(a)) \leq ((2d)!/(2d-r)!)^2 m^{2d-r}$. □

4.1 Applications

Theorem 2. *$dc_r(S_n^{2d}) \geq \left(\frac{(2d-r)!}{(2d)!}\right)^{2/(2d-r)} \binom{n}{d}^{1/(2d-r)}$, for $1 \leq r < 2d < n$.*

Proof. Consider any $\binom{n}{d} \times \binom{n}{d}$ minor H of $T^{2d}S_n^{2d}$, where rows and columns are indexed by all d-subsets $I = \{i_1, i_2, \ldots, i_d\}$ and $J = \{j_1, j_2, \ldots, j_d\}$ of $[n]$, respectively. Then

$$H_{I,J} = \frac{\partial^{2d} S_n^{2d}}{\partial x_{i_1} \partial x_{i_2} \ldots \partial x_{i_d} \partial x_{j_1} \partial x_{j_2} \ldots \partial x_{j_d}} = \begin{cases} 1 \text{ if } I \cap J = \emptyset \\ 0 \text{ otherwise.} \end{cases}$$

In other words H is the *communication matrix* of set disjointness, which is known to have rank $\binom{n}{d}$ (See page 175, [15]). The result now follows by applying Theorem 1. □

Corollary 1. *$dc_r(S_n^{2d}) = \Omega(n^{d/(2d-r)})$, for $1 < d < r < 2d = O(1)$.*

In the most restrictive setting one has that $\mathrm{dc}_{2d-1}(S_n^{2d}) = \Omega(n^d)$, for constant $d > 1$. As a matter of fact, Theorem 2 yields an exponential lower bound in this case for non-constant d:

Corollary 2. *$dc_{2d-1}(S_n^{2d}) = n^{\Omega(\epsilon n^{1/5-\epsilon})}$, for any fixed $0 < \epsilon < 1/5$, where $d = \lfloor n^{1/5-\epsilon} \rfloor$.*

The above is the most restrictive case in which computability of S_n^{2d} is not yet ruled out by Theorem 1. Careful inspection of Theorem 1 does shows that S_n^{2d} cannot be computed using a $2d$-lowerable map if $\binom{n}{d} > (2d!)^2$, e.g. for all large enough n, if d is constant.

Using Lemma 3, we obtain lower bounds for products of symmetric polynomials on disjoint variables sets.

Theorem 3. *For constants $d > 1$, $p \geq 1$, and r with $dp < r < 2dp$, let $X^1, X^2, \ldots, X^p$ be disjoint sets of variables of size n each. Then $dc_r(\prod_{i\in[p]} S_n^{2d}(X^i)) = \Omega(n^{2dp/(2dp-r})$.*

Proof. Let $f = \prod_{i\in[p]} S_n^{2d}(X^i)$. By Lemma 3, *rank* $T^{2dp}f \geq \prod_{i\in[p]} \mathit{rank}\ T^{2d}S_n^{2d}(X^i) \geq \binom{n}{d}^p$. The latter inequality follows from the proof of Theorem 2. Applying Theorem 1 yields the result. □

Next, we consider iterated matrix product. Define $IMM_{n,d}$ by summing entries of an iterated matrix multiplication: $IMM_{n,d} = \sum_{i,j\in[n]}(\prod_{r\in[d]} X^r)_{ij}$, where $X^1, X^2, \ldots, X^d$ are $n \times n$ matrices with disjoint sets of variables. Since $IMM_{n,d}$ has algebraic branching programs of size dn, we have that $\mathrm{dc}(IMM_{n,d}) = O(nd)$, which is *sublinear* in the number of variables n^2d.

Lemma 4. *For $d \geq 1$, rank $T^{2d}IMM_{n,2d} \geq n^d$.*

Proof. We prove by induction on d the following claim:

Claim. Let $(g_1, g_2, \ldots, g_n) = (1, 1, \ldots, 1)X^1X^2 \ldots X^{2d-1}X^{2d}$. Then for each i, *rank* $T^{2d}g_i \geq n^d$.

The case $d = 1$ follows directly by inspection. Now suppose, $(g_1, g_2, \ldots, g_n) = (1, 1, \ldots, 1)X^1X^2 \ldots X^{2d-1}X^{2d}$. Let $G_i = T^{2d}g_i$. Let $(h_1, h_2, \ldots, h_n) = (g_1, g_2, \ldots, g_n)X^{2d+1}X^{2d+2}$. Consider the minor H of $T^{2d+2}\sum_i h_i$ where rows contain a variable from X^{2d+1}, but not from X^{2d+2}, and vice-versa for columns. Provided we order at the top level rows according to variables from X^{2d+1} and columns according to variables from X^{2d+2}, this minor will be of the following form:

$$H = \begin{pmatrix}
G_1 & \dots & G_1 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\
 & 0 & & G_1 & \dots & G_1 & \dots & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \dots & 0 & 0 & \dots & 0 & \dots & G_1 & \dots & G_1 \\
G_2 & \dots & G_2 & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\
 & 0 & & G_2 & \dots & G_2 & \dots & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \dots & 0 & 0 & \dots & 0 & \dots & G_2 & \dots & G_2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
G_n & \dots & G_n & 0 & \dots & 0 & \dots & 0 & \dots & 0 \\
 & 0 & & G_n & \dots & G_n & \dots & 0 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & \dots & 0 & 0 & \dots & 0 & \dots & G_n & \dots & G_n
\end{pmatrix}$$

For any i, $T^{2d+2}h_i$ is obtained by setting to zero in the above columns which at the top level are indexed by variables X_{kj}^{2d+2} with $j \neq i$. Hence for each i, *rank* $T^{2d+2}h_i \geq n \cdot \max_{k\in[n]}$ *rank* $G_k \geq n^{d+1}$. This proves the claim.

Observe that in the above the individual $T^{2d+2}h_i$ do not interfere, in the sense that *rank* $H \geq$ *rank* $T^{2d}h_i$, for each i. Hence it can be observed that *rank* $T^{2d}\sum_i g_i \geq n^d$. □

Combining Theorem 1 and Lemma 4 yields the following results:

Theorem 4. $dc_r(IMM_{n,2d}) \geq \left(\frac{(2d-r)!}{(2d)!}\right)^{2/(2d-r)} n^{d/(2d-r)}$*, for* $1 \leq r < 2d$*.*

Corollary 3. $dc_r(IMM_{n,2d}) = \Omega(n^{d/2d-r})$*, for* $1 \leq d < r < 2d = O(1)$*.*

Corollary 4. $dc_{2d-1}(IMM_{n,2d}) = n^{\Omega(\epsilon n^{1/4-\epsilon})}$*, for any fixed* $0 < \epsilon < 1/4$*, where* $d = \lfloor n^{1/4-\epsilon} \rfloor$*.*

5 Tight Bounds on Determinantal Complexity

Results in this section hold for fields F of characteristic zero.

5.1 Elementary Symmetric Polynomials

Note using Theorem 2 with $d = 1$ and $r = 1$ yields that $\mathrm{dc}(S_n^2) \geq n/4$. For $n \geq 1$, we define $2n$-vector $p_n = (1, -1, 1, -1, \dots, 1, -1)$. Easily verified by induction on n, we have that the univariate polynomial $((t-1)(t+1))^n$ equals $\sum_{r=0}^n t^{2r}(-1)^{n-r}\binom{n}{r}$. Since $S_{2n}^d(p_n)$ is the coefficient of t^{2n-d} we obtain the following:

Proposition 2. *For any* $1 \leq d \leq n$, $S_{2n}^{2d-1}(p_n) = 0$.

Proposition 3. *For any* $0 \leq d \leq n$, $S_{2n}^{2d}(p_n) = (-1)^d \binom{n}{d}$.

Lemma 5. *rank* $T^2 S_{2n}^{2d-1}(p_n) = 2n$, *for* $2 \leq d \leq n$.

Proof. Let $H = T^2 S_{2n}^{2d-1}$. Then $H_{ij} = S_{2n-2}^{2d-3}(X/x_i, x_j)$, if $i \neq j$, and zero otherwise. By symmetry and Proposition 2, if i and j have distinct parity, $H_{ij}(p_n) = S_{2n-2}^{2d-3}(p_{n-1}) = 0$. Let H^o and H^e be the $n \times n$ minors of H formed by all odd, respectively even, rows and columns. H acts independently on the odd and even indices of a vector, so *rank* $H =$ *rank* $H^o +$ *rank* H^e. H^o has zeroes on its diagonal, and by symmetry off-diagonal entries will be the same value $S_{2n-2}^{2d-3}(-1, -1; p_{n-2})$. Provided this is a nonzero value, H^o will be non-singular. A straightforward calculation shows this value is $(-1)^{d-2}\binom{n-2}{d-2}$. Similarly *rank* $H^e = n$. □

Following the proof strategy as outlined in Section 3, and using the bound *rank* $T^2 det_m(B) \leq 2m$, for singular B of [6], we obtain the following corollary:

Corollary 5. $dc(S_{2n+1}^{2d-1}) \geq dc(S_{2n}^{2d-1}) \geq n$, *for* $2 \leq d \leq n$.

For even degree, a simple calculation shows that $S_{2n+2}^{2d}(a, b, p_n) = 0$ whenever $2ab = (n+1)/d$.

Lemma 6. *For* $2 \leq d \leq n$, *there exists constant* μ, *such that for* $q_{n+2} = (\mu(n+1), \frac{1}{2d\mu}, p_n)$, *rank* $T^2 S_{2n+2}^{2d}(q_{n+2}) \geq n$.

Proof. Let $H = T^2 S_{2n+2}^{2d}$. Consider the $n \times n$ minor H' of H given by rows i and columns j, where $i, j \geq 3$ and i, j are both odd. The diagonal of H' has all entries zero. By symmetry, $H'_{i,j}(q_{n+2})$ has the same value $S_{2n}^{2d-2}(\mu(n+1), \frac{1}{2d\mu}, -1, -1, p_{n-2})$, for all for $i \neq j$. A simple calculation shows that one can easily pick μ large enough so this value does not vanish, in which case H' has rank n. □

Corollary 6. $dc(S_{2n+3}^{2d}) \geq dc(S_{2n+2}^{2d}) \geq n/2$, *for* $2 \leq d \leq n$.

Shpilka and Wigderson give depth 6 arithmetical formulas of size $O(nd^3 \log d)$ for S_n^d [10]. By the universality of the determinant [2] (cf. [13]) we have $dc(S_n^d) = O(nd^3 \log d)$. Hence the following theorem holds:

Theorem 5. *For any constant* $d > 1$, $dc(S_n^d) = \Theta(n)$.

In other words, for constant d, $\mathrm{dc}(S_n^d) = \Theta(L(S_n^d))$. For non-constant d, it is an open problem whether any improved upper bound can be given for $\mathrm{dc}(S_n^d)$ beyond applying the universality of the determinant to best-known formulas. We note this can easily be done for the *permanental complexity* of S_n^d. Also permanental complexity minorizes formula size, i.e. for any polynomial f, $\mathrm{pc}(f) = O(L(f))$. However, for any d, $\mathrm{pc}(S_n^d) \leq n$, over fields of characteristic zero. Namely, the permanent of a matrix A whose first d rows equal

$(x_1, x_2, \ldots, x_n)$, and with other entries equal to one, is $d!(n-d)!S_n^d$. Note one could try to obtain a polynomial h_n^d that has the same support as S_n^d by taking $h_n^d = det_n(A \circ C)$, where $\circ$ denotes the Hadamard product, and C is some matrix of constants $C = (_{ij})_{i,j\in[n]}$. In this case for $I \subset [n]$ of size d, the monomial $\prod_{i\in I} x_i$ appears with coefficient $\pm det(C_{[d],I})det(C_{\{d+1,d+2,\ldots,n\},[n]/I})$, where $C_{I,J}$ denotes the submatrix with rows I and columns J. It is easily possible to choose C so that all these coefficients are nonzero. However, requiring all coefficients to equal 1 will not be feasible in general. We immediately know this from our previous investigation, since the mapping defined this way is d-lowerable. We know by Theorem 1, one cannot compute S_n^{2d} using $2d$-lowerable maps, if $\binom{n}{d} > (2d!)^2$.

5.2 Iterated Matrix Product

Theorem 6. *For any $d \geq 1$, $dc(IMM_{n,2d}) \geq n/2$.*

Proof. Consider the minor of $T^2 IMM_{n,2d}$ corresponding to rows with variables $X_{11}^{2d-1}, \ldots, X_{1n}^{2d-1}$ and columns with variables of X^{2d}. With the appropriate ordering of variables from X^{2d}, this minor looks like:

$$\begin{pmatrix} g_{11} \ldots g_{1n} & 0 & \ldots & 0 & \ldots & 0 & \ldots & 0 \\ 0 & & g_{21} & \ldots g_{2n} & \ldots & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 \ldots 0 & 0 & \ldots & 0 & \ldots & g_{n1} & \ldots & g_{nn} \end{pmatrix},$$

where $g_{ij} = \partial^2 IMM_{n,2d}/\partial X_{1i}^{2d-1}\partial X_{ij}^{2d}$. Note g_{ij} is the same for all $i, j \in [n]$. It suffices to find point a, such that $IMM_{n,2d}(a) = 0$ and $g_{ij}(a) \neq 0$. Since g_{ij} does not contain any variables from X^{2d-1} and X^{2d}, this can easily be achieved by setting to zero all variables in X^{2d-1} and X^{2d}, and setting all other variables to 1. $\square$

By the remarks before Lemma 4, we have that $\mathrm{dc}(IMM_{n,2d}) = \Theta(n)$, for constant d. More generally, by means of reduction, we have for row vector $\langle u|$ and column vector $|v\rangle$ of variables the following corollary:

Corollary 7. *For constant d, $dc(\langle u| X^1 X^2 \ldots X^{2d} |v\rangle) = \Theta(n)$.*

5.3 Symmetric Power Sum Polynomials

The symmetric power sum polynomial of degree d in n variables is defined by $P_n^d = \sum_{i\in[n]} x_i^d$. P_n^d has arithmetical formula size $L(P_n^d) = O(nd)$. The partial derivatives matrix $T^2 P_n^d$ is given by the diagonal matrix $d(d-2)\mathrm{diag}(x_1^{d-2}, x_2^{d-2}, \ldots, x_n^{d-2})$. By the strategy described in Section 3, it suffices to find a zero a of P_n^d with all entries $a_i \neq 0$ to obtain that for all $d \geq 2$, $\mathrm{dc}(P_n^d) \geq n/2$. For an arbitrary field F of characteristic zero one achieves this by going to an extension field G of F where the dth root of $(1-n)$ exists. Over G, one takes $a = (1, 1, \ldots, 1, (1-n)^{1/d})$ to show $\mathrm{dc}_G(P_n^d) \geq n/2$. The lower bound follows since $\mathrm{dc}_F(P_n^d) \geq \mathrm{dc}_G(P_n^d)$.

Corollary 8. *For any constant $d \geq 2$, $dc(P_n^d) = \Theta(n)$.*

Note the statement of the corollary would be false if we drop the restriction (which holds section-wide) on the characteristic of the underlying field, e.g. $\mathrm{dc}(P_n^2) = 2$ over $GF(2)$.

6 Conclusions

In this paper efforts have been made to understand the current barrier to proving strongly nonlinear determinantal complexity lower bounds. One of the main objectives has been to demonstrate this barrier can be broken through under the mathematically natural restriction of r-lowerability. It will be interesting to see whether one can do this under any weaker assumptions. The inquiry leaves us with some intriguing questions regarding determinantal representation:

Problem 1. Can S_n^{2d} be computed using an r-lowerable map, with $d < r < 2d$?

Problem 2. Can *every* homogeneous polynomial of degree $2d$ be computed using an r-lowerable map for some r such that $d < r < 2d$?

Problem 3. For non-constant d depending on n, can we give better upper bounds for $\mathrm{dc}(S_n^d)$ than would be obtained by applying Valiant's universality construction of the determinant to best-known formulas for S_n^d ?

Problem 4. Can we explicitly construct a family of polynomials of non-constant degree d in n variables, with $d = o(n)$, whose determinantal complexity is shown to be $\Theta(n)$?

References

1. Bürgisser, P.: Cook's versus valiant's hypothesis. Theor. Comp. Sci. 1, 71–88 (2000)
2. Valiant, L.: Completeness classes in algebra. Technical Report CSR-40-79, Dept. of Computer Science, University of Edinburgh (April 1979)
3. Bürgisser, P.: Completeness and Reduction in Algebraic Complexity Theory. Springer, Heidelberg (2000)
4. Valiant, L.: The complexity of computing the permanent. Theor. Comp. Sci. 8, 189–201 (1979)
5. Toda, S.: PP is as hard as the polynomial-time hierarchy. SIAM J. Comput. 20, 865–877 (1991)
6. Mignon, T., Ressayre, N.: A quadratic bound for the determinant and permanent problem. International Mathematics Research Notices, 4241–4253 (2004)
7. Cai, J.-Y., Chen, X., Li, D.: A quadratic lower bound for the permanent and determinant problem over any characteristic $\neq 2$. In: STOC 2008: Proceedings of the 40th annual ACM symposium on Theory of computing, pp. 491–498 (2008)
8. Nisan, N.: Lower bounds for non-commutative computation: extended abstract. In: Proc. 23rd Annual ACM Symposium on the Theory of Computing, pp. 410–418 (1991)

9. Nisan, N., Wigderson, A.: Lower bounds on arithmetic circuits via partial derivatives. Computational Complexity 6, 217–234 (1996)
10. Shpilka, A., Wigderson, A.: Depth-3 arithmetic formulae over fields of characteristic zero. Journal of Computational Complexity 10(1), 1–27 (2001)
11. Raz, R.: Multilinear formulas for permanent and determinant are of super-polynomial size. Proc. 36th Annual ACM Symposium on the Theory of Computing, 633–641 (2004)
12. Raz, R.: Separation of multilinear circuit and formula size. In: Proc. 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 344–351 (2004)
13. von zur Gathen, J.: Feasible arithmetic computations: Valiant's hypothesis. Journal of Symbolic Computation 4, 137–172 (1987)
14. Baur, W., Strassen, V.: The complexity of partial derivatives. Theor. Comp. Sci. 22, 317–330 (1982)
15. Ukna, S.J.: Extremal Combinatorics. Springer, Heidelberg (2001)

Simulation of Arithmetical Circuits by Branching Programs with Preservation of Constant Width and Syntactic Multilinearity

Maurice Jansen[1,⋆] and Raghavendra Rao B.V.[2]

[1] Centre for Theory in Natural Science, Aarhus University, IT-Parken, Aabogade 34, DK-8200 Aarhus N, Denmark, phone: +45 8942 5600
mjjansen@daimi.au.dk

[2] The Institute of Mathematical Sciences, Chennai, 600 113, India
bvrr@imsc.res.in

Abstract. We study structural properties of restricted width arithmetical circuits. It is shown that syntactically multilinear arithmetical circuits of *constant* width can be efficiently simulated by syntactically multilinear algebraic branching programs of *constant* width, i.e. that sm-VSC^0 $\subseteq$ sm-VBWBP. Also, we obtain a direct characterization of *poly*-size arithmetical formulas in terms of *multiplicatively disjoint* constant width circuits, i.e. MD-VSC^0 = VNC^1.

For *log*-width *weakly-skew* circuits a width efficient multilinearity preserving simulation by algebraic branching programs is given, i.e. weaklyskew-sm-VSC^1 $\subseteq$ sm-VBP[width=$\log^2 n$].

Finally, coefficient functions are considered, and closure properties are observed for sm-VSC^i, and in general for a variety of other syntactic multilinear restrictions of algebraic complexity classes.

1 Introduction

In this paper the computational power of space bounded computation is studied in the arithmetical setting by considering arithmetical circuits of restricted width. For such circuits several elementary questions are still left unanswered. We are interested in the following question posed in [6]: can arithmetical circuits of constant width and polynomial degree be simulated by polynomial size arithmetical formulas ? If indeed so, this would yield an equivalence, since a simulation the other way around follows from a construction by Ben Or and Cleve [1].

One strategy to approach the above question is to investigate under what additional assumptions one can indeed do the simulation. Mahajan and Rao show that every *syntactically multilinear* constant width circuit has an equivalent polynomial size arithmetical formula [6]. However, it was left open whether this arithmetical formula can be guaranteed to be syntactically multilinear. The starting point of this note is the observation that with a careful modification this can in fact be achieved. In other words, letting sm-VSC^0 and sm-VNC^1

⋆ Corresponding author.

stand for the syntactically multilinear restrictions of the arithmetical complexity classes corresponding to *poly*-size constant width circuits and *poly*-size circuits of $O(\log n)$-depth, respectively, we have the following theorem:

Theorem 1. *sm-VSC^0 $\subseteq$ sm-VNC^1.*

Details regarding this modification will appear in the full version of this paper. Theorem 1 raises the following question: is it that syntactically multilinear constant width arithmetical circuits are equivalent to constant width syntactically multilinear branching programs? The main result of this paper is an answer to this question in the affirmative. Letting sm-VBWBP stand for the class corresponding to *poly*-size algebraic branching program of constant width, we strengthen Theorem 1 as follows:

Theorem 2. *sm-VSC^0 $\subseteq$ sm-VBWBP.*

Syntactically multilinear circuits can be assumed without loss of generality to be *multiplicatively disjoint* (See [8]). Hence it is natural to consider enhancements to the construction underlying Theorem 2 under the assumption of multiplicative disjointness, and also the even more stringent condition of *weak skewness* (See [8]). The best dependence on width we obtain under the latter restriction, allowing us to conclude $O(\log n)$-width, *poly*-size weakly-skew arithmetical circuits can be simulated by $O(\log^2 n)$-width, *poly*-size algebraic branching program with preservation of syntactic multilinearity. This will follow from a careful modification of the construction given in [5].

In the general world, we observe that if a given constant width circuit is multiplicatively disjoint, then it can be depth reduced to yield a formula. To the best of our knowledge, this is the largest chunk of the class of constant width arithmetical circuits which are known to have equivalent formulas.

Following [7], we study the complexity of coefficient functions of polynomials. Closure properties will be observed that hold quite universally among the syntactically multilinear circuit classes, in particular for the restricted width classes mentioned above. Although coefficient functions are known to be VNP hard even in the case of depth three arithmetic formulas, generally syntactically multilinear circuit classes are closed for coefficient functions. Also, we show that if any coefficient function of a polynomial is in a syntactic multilinear class, then so is the polynomial itself. Hence, in the terminology of [7], generally a syntactically multilinear arithmetic circuit class is *stable* for coefficient functions.

The rest of this paper is divided as follows. In Section 2 we introduce definitions and notations. Section 3 contains the construction for weakly-skew circuits. Section 4 contains a proof of Main Theorem 2. In Section 5 we study coefficient functions. Finally, we end in Section 6 by posing several open problems.

2 Preliminaries

For integer n, $[n]$ denotes the set $\{1, 2, \ldots, n\}$. Let R be a commutative ring with multiplicative identity 1. An *arithmetical circuit* C over R is a directed acyclic graph, with nodes (called *gates*) with labels taken from $\{+, \times\} \cup X \cup R$, where $X = \{x_1, x_2, \ldots, x_n\}$. A node with in-degree zero must take its label from X or R, depending on whether it represents and *input* or *constant gate*. C has at least one node of out-degree zero called an *output gate*. Every gate in g computes a polynomial in $R[X]$ defined in the usual way. The polynomial computed by the circuit is the polynomial computed by the output gate, and if C has more than one output gate, then this is the union of all such polynomials. For a gate f, $Var(f)$ denotes the subset of X of variables that appear in the subcircuit rooted at f. A circuit is *layered*, if the node set can be partitioned into a sequence of sets, called *layers*, with edges only between consecutive layers.

Fan-in (*fan-out*) of C is the maximum in-degree (out-degree) of any gate in C. The size of C is defined as the number of gates and edges (called *wires*) in C. If C has a constant fan-in then we take size to be simply the total number of gates. *Depth* of a circuit is the length of longest directed path in the underlying graph. *Width* of a layered circuit is the maximum number of nodes at any layer. A *formula* is a circuit, where fan-out of every gate is bounded by one. The *formal degree* of a gate (degree for short) is defined inductively as follows: input gates have degree one, and for an addition or multiplication gate it is the sum or product of the degree of its inputs, respectively.

A circuit C is said to be *skew*, if for every multiplication gate $f = g \times h$, one of g or h is an input or constant gate. C is said to be *weakly skew*, if for every $f = g \times h$, either the edge (g, f) or (h, f) is a bridge in the circuit, i.e removing the edge increase the number of weakly connected components. Further, in a *multiplicatively disjoint* (MD for short) circuit, for every gate $f = g \times h$, the sub-circuits rooted at g and h are disjoint (as graphs). Note that MD-circuits are generalizations of weakly skew circuits, which in turn contains skew circuits.

The following are the arithmetical circuit classes that will be used in the next sections. These contain families of polynomials $(f_m)_{m \geq 1}$, where $f_m \in R[x_1, x_2, \ldots, x_{p(m)}]$, for some polynomial p. The measures size, width, depth, fan-in and fan-out are all defined in terms of m.

VAC^0	: *poly*-size, constant-depth, unbounded fan-in circuits.
VNC^1	: *poly*-size, *log*-depth, constant fan-in circuits.
VF	: *poly* size formulas.
VsSC^i	: *poly*-size, $O(\log^i m)$-width, *poly*-degree circuits.
VSAC^1	: *poly*-size, *log*-depth, constant $\times$-fan-in circuits.
VP	: *poly*-size, *poly*-degree circuits.

The class VNP is defined as families of polynomials $(g_m)_{m \geq 1}$, where $g_m = \sum_{b \in \{0,1\}^{q(m)}} f_m(x, b)$, for polynomial $q(m)$ and $(f_m)_{m \geq 1} \in \mathrm{VP}$.

An *algebraic branching program* (BP for short) is a directed acyclic graph, where edges are labeled from $X \cup R$. There are two designated nodes, s and t, where s has in-degree zero and t has out-degree zero. We assume that a BP is layered. Size of a BP is the number of nodes and edges in it and width is the maximum number of nodes at any layer. Length of a BP is the number of layers in it. Depth of a BP B equals 1+length(B). The polynomial computed by a BP P, is

$$\begin{tikzcd}
\mathrm{VAC}^0 \arrow[r] & \boxed{\begin{array}{l}\mathrm{VBWBP} = \mathrm{VF} = \\ \text{MD-VsSC}^0 = \mathrm{VNC}^1\end{array}} \arrow[r] \arrow[drr] & \mathrm{VsSC}^0 \arrow[r] & \mathrm{VsSC}^1 \arrow[r] & \mathrm{VSAC}^1 \\
 & & & \mathrm{VLWBP} \arrow[u] \arrow[r] & \mathrm{VBP} \arrow[u]
\end{tikzcd}$$

Fig. 1. Relationship among arithmetic classes around arithmetic formulas

$$\begin{tikzcd}
\text{sm-VAC}^0 \arrow[r] & \boxed{\begin{array}{l}\text{sm-VBWBP} = \\ \text{sm-VSC}^0\end{array}} \arrow[r] \arrow[d] & \text{sm-VSC}^1 \arrow[r] & \text{sm-VSAC}^1 \\
 & \text{sm-VNC}^1 \arrow[r] & \text{sm-VLWBP} \arrow[u] \arrow[r] & \text{sm-VBP} \arrow[u]
\end{tikzcd}$$

Fig. 2. Syntactic Multilinear Versions

the sum of weights of all s-t paths in P, where the weight of a path is the product of all edge labels in the path. We will also consider multi output BPs, where the above is generalized in the obvious way to several nodes $t_1, t_2, \ldots, t_m$ existing at the last level. Note that BPs can be simulated by skew circuits and vice versa with a constant blow up in the width. We let VBP, VLWBP and VBWBP stand for classes corresponding to *poly*-size BPs of *poly*, *log* and *constant* width, respectively. Known relationships among the classes defined so far are: $\mathrm{VAC}^0 \subseteq \mathrm{VNC}^1 = \mathrm{VF} = \mathrm{VBWBP}$ [2,1], and $\mathrm{VBP} \subseteq \mathrm{VSAC}^1 = \mathrm{VP} \subseteq \mathrm{VNP}$ [13].

A circuit C is *syntactically multilinear*, if for every multiplication gate $f = g \times h$, $\mathit{Var}(g) \cap \mathit{Var}(h) = \emptyset$. A BP is said to be syntactically multilinear, if no s-t path contains the same variable twice. For a complexity class $\mathcal{C}$, its syntactically multilinear version is denoted by sm-$\mathcal{C}$. For sSC^i we drop one 's' as *poly* degree is implied, i.e. we write sm-VSC^i and also MD-SC^i. Known relationships are: sm-VBWBP $\subseteq$ sm-VsSC^0 $\subseteq$ sm-VNC^1 $=$ sm-VF ([6]) and sm-VBP $\subseteq$ sm-VSAC^1 $=$ sm-VP ([10]).

The above results together with the main theorem of our paper lead to the scenario as depicted in Figures 1 and 2. The main contrasting point is that $\mathrm{VBWBP} = \mathrm{VNC}^1 \subseteq \mathrm{VsSC}^0$, whereas sm-$\mathrm{VSC}^0$ = sm-VBWBP $\subseteq$ sm-VNC^1.

3 Weakly Skew Circuits

The main objective of this section is to prove the following result:

Theorem 3. weaklyskew-sm-VSC^0 = sm-VBWBP.

Lemma 1. *Let C be an arithmetical circuit of width w and size s. Then there is an equivalent arithmetic circuit C' of width $O(w)$ and size poly(s) such that fan-in and fan-out of every gate is bounded by two, and every layer has at most one $\times$ gate. Moreover, C' preserves any of the properties of syntactic multilinearity, weakly-skewness and multiplicatively disjointness.*

Proof. Let k be the bound on maximum fan-in and fan-out of C. First we can reduce the fan-in to two by staggering the circuit and keeping copies of the gates as and when needed. This blows up the width to $2w$ and size to wks. Now in a similar manner we can ensure that the fan-out of a gate is bounded by two and the size blow up will now be w^2k^2s and width will be $4w$. To ensure the second condition we need to push the gates (using staggering and dummy + gates) up by at most $4w$ levels, thus making the total width $8w$ and size $2w^2k^2s$. Since $k \leq w + n$ and $w \leq s$ we have size bounded by $poly(s, n)$. □

For a BP B of depth d with a single source s, we say B *is endowed with a mainline*, if there exist nodes $v_1, v_2, \ldots, v_{d-1}$ reachable only along the path $s, v_1, v_2, \ldots, v_{d-1}$, and if the labels on this path are all set to the field constant 1. We require this feature to ensure our construction builds programs with a single source. For BPs B_1 and B_2, *piping* the mainline of B_1 into the mainline of B_2 is the operation of removing the edge from the source of B_2 to the first node v of the mainline of B_2, and adding an edge from the last node w of the mainline of B_1 to v.

In the following, as a typical next step we draw an edge from an output of B_1 to the source of B_2. This can be from last node w, if we just want to compute B_2 or some other output in case we want to do multiplication. Note the width of this composition is bounded by $\max(\text{width}(B_1), \text{width}(B_2))$.

Lemma 2. *Let C be a weakly skew circuit of width $w > 1$ and size $s > 1$ as given by Lemma 1. Let $f_1, \ldots, f_w$ be the output gates of C. Then there exists an equivalent arithmetical BP $[C]$ of width $w^2 + 1$, length $O(2^w s)$ and size $O((w^2 + 1)2^w s)$. $[C]$ has a single start node b and and terminal nodes $[f_1], \ldots, [f_w], v$ and will be endowed with a mainline ending in v. Moreover, if C is syntactically multilinear then so is $[C]$.*

Proof. We proceed by induction on both s and w. If $s = 2$, the lemma holds trivially. If $w = 2$, C is a skew-circuit hence can be seen as a BP of width 3 (We also need to add a mainline hence width is 3).

Let $s > 2$ and $w > 2$ be given, and assume that C has at least 2 layers. By induction hypothesis, the lemma holds for all circuits of size s' and w', where either $s' < s$ and $w' \leq w$ or $s' \leq s$ and $w' < w$.

Wlog. assume that f_1 is a $\times$ gate and $f_2, \ldots, f_w$ are + gates. Let C' be the circuit obtained by removing the output gates of C. Let $g_1, \ldots, g_w$ be the output gates of C'. Wlog. assume $f_1 = g_1 \times g_2$, and also that the edge (g_1, f_1) is a bridge in the circuit. We define subcircuits D, E and of C' as follows: D is obtained from C' by deleting the sub-circuit rooted at g_1, E is the sub-circuit rooted at g_1. Let $s' = size(C'), w' = \text{width}(C')$, $s_J = \text{size}(J)$ and $w_J = \text{width}(J)$ for $J \in \{D, E\}$. Note that $s = s' + w$, and $s_J < s$ for $J \in \{D, E\}$.

By induction hypothesis, we have branching programs $[D]$ and $[E]$, both endowed with a mainline. Let $[g_1], v'$ denote the output of E and $[g_2], \ldots, [g_w], v''$ denote the output nodes of $[D]$, where v' and v'' are the last nodes on the mainlines. Let $[F]$ be the subprogram of $[D]$, which consists of all paths from the source of $[D]$ to $[g_2]$ and v''. Construct the program $[C]$ with output nodes $[f_1], \ldots, [f_w], v$ as follows:

Case 1. $w_E \leq w - 1$.
We compose $[D]$ before $[E]$ as follows:

1. For $i, j \geq 2$, $[g_j]$ has an edge to $[f_i]$ iff g_j is an input to f_i.
2. For input gates f_i, draw an edge from v'' to $[f_i]$ with the appropriate label.
3. Add an edge from $[g_2]$ to $[f_1]$.
4. Identify the start node of $[E]$ with $[f_1]$ and relabel the output node of $[E]$ as $[f_1]$. Pipe the mainline of $[D]$ into the mainline of $[E]$.
5. Stagger the nodes $[f_2], \ldots, [f_w]$ until the last level of the new program.

Size and width analysis: By induction hypothesis width$([E]) \leq (w_E)^2 + 1 \leq (w-1)^2 + 1$, and width$([D]) \leq w^2 + 1$, and length$([E]) \leq 2^{w-1} size(E)$ and length$([D]) \leq 2^w size(D)$. Now width$([C]) = \max\{$width$([D])$, width$([E]) + w - 1\} \leq w^2 + 1$ and length$([C]) =$ length$([D]) +$ length$([E)] \leq 2^{w_D} s_D + 2^{w_E} s_E \leq 2^w s_D + 2^{w-1} s_E \leq 2^w s$ as $s = s_D + s_E + w$.

Case 2. $w_E = w$, and hence $w_F \leq w - 1$ and $w_D \leq w - 1$.
We compose $[E]$ before $[F]$ before $[D]$ as follows:

1. Identify $[g_1]$ with the source of $[F]$, and pipe the mainline of $[E]$ into the mainline of $[F]$.
2. Add an edge from v' (last node of mainline of $[F]$) to the source of $[D]$,
3. Pipe the mainline of $[F]$ into the mainline of $[D]$.
4. Alongside $[D]$ stagger the output of $[F]$ (which equals $[f_1]$).
5. For $i, j \geq 2$, $[g_j]$ has an edge to $[f_i]$ iff g_j is an input to f_i.
6. Finally, for input gates f_i, draw an edge $(v'', [f_i])$ with the appropriate label.

Size and width analysis: By induction hypothesis, width$([E]) \leq w^2 + 1$, width$([D]) \leq (w-1)^2 + 1$. Hence also width$([F]) \leq (w-1)^2 + 1$. Observe, width$([C]) \leq max($width$([E])$, width$([F])$, width$([D]) + 1) \leq w^2 + 1$.

Now, length$([C]) =$ length$([E]) +$ length$([F]) +$ length$([D]) + 1 \leq 2^w s_E + 2^{w-1} s_F + 2^{w-1} s_D + 1 \leq 2^w (s_D + s_E) + 1 \leq 2^w s$. Since size of a layered BP is length × width we have the required size bound. If C was syntactic multilinear to start with, then it is easy to see that so is $[C]$. □

Corollary 1. weaklyskew-VSC^0 = VNC^1 = VBWBP.

Corollary 2. weaklyskew-sm-VSC^1 $\subseteq$ sm-VBP[width $= \log^2 n$].

Conversion to Weakly Skew. We note that it is possible to process a multiplicatively disjoint circuit into a weakly-skew circuit with preservation of syntactic multilinearity.

Lemma 3. *For any leveled syntactically multilinear multiplicatively disjoint circuit C of width $w \geq 1$ and size $s \geq 1$ such that each layer has at most one multiplication gate, there exists a leveled syntactically multilinear weakly-skew circuit $[C]$ of size at most s^w such that for any gate g of C, there is a gate $[g]$ in $[C]$ that computes the same polynomial.*

Proof. We prove the above lemma by induction on both s and w. We have two base cases: $s = 1$ and $w = 1$. In both cases the lemma trivially holds.

Let $s > 2$ and $w > 1$ be given. By induction hypothesis, the lemma holds for any circuit of size s' and width w' for which either $s' < s$ or $w' < w$.

Case I. The output layer has addition and input gates only.
Let C' be the circuit C excluding the output layer. Recursively process C' and add the output layer back to form $[C]$ from $[C']$. We have that size$([C]) \leq$ size$([C'])^w + w \leq (s - w)^w + w \leq s^w$.

Case II. The output layer contains a multiplication gate f.
Let C_1 and C_2 be the subcircuits rooted at the inputs of the multiplication gate f. Since C_1 and C_2 are disjoint, one of them is guaranteed to have width at most $w - 1$. Wlog. assume C_1 has width at most $w - 1$. Let $g_1, g_2, \ldots, g_m$ be all gates not in C_1 that take input from C_1. Let D be the subcircuits formed by the gates in C excluding C_1, where any input g taken by a g_i from C_1 is removed. Let $s_d =$ size(D) and $s_1 =$ size(C_1). Then $s = s_d + s_1$. Recursively process D and C_1 (separately) to obtain weakly skew circuits $[D]$ and $[C_1]$ of sizes s_d^w and s_1^{w-1}, respectively. Now we put back removed inputs to each of $[g_1], [g_2], \ldots, [g_m]$ from the appropriate gate in $[C_1]$.

The circuit we obtain from $[C_1]$ and $[D]$ this way is almost weakly skew. The only issue is that the adding back of original inputs from say $[g] \in [C_1]$ at input $[g_i]$ can violate the weak skewness condition for $[g_i]$ and also for gates in $[C_1]$. We resolve this by simply duplicating the subcircuit rooted at $[g]$. Observe that size$([C]) \leq s_d^w + m \cdot s_1^{w-1} \leq s_d^w + s_d \cdot s_1^{w-1} \leq (s_d + s_1)^w = s^w$. □

The above gives an alternative proof of sm-VSC$^0 \subseteq$ sm-VBP. Namely, by above lemma we get that sm-VSC$^0 \subseteq$ weaklyskew-sm-VP. Next use the construction from [5] that shows weaklyskew-sm-VP $\subseteq$ sm-VBP. The other way to arrive at this is by Theorem 1. However, there the size of the resulting BP is $O(2^{w^2} s^{25w})$. In this regard, Lemma 3 still yields a slightly better size bound than the construction underlying Theorem 2, since there the resulting size is $O(w^2 s^w)$.

4 Multiplicatively Disjoint Circuits

In this section we prove Theorem 2. In fact, we prove that multiplicatively disjoint circuits of constant width and polynomial size can be simulated by BPs of constant width and polynomial size preserving the syntactic multilinearity property. In general multiplicatively disjoint circuits are equivalent to polynomial degree circuits (see [8]). The following theorem can be deduced from [6]:

Theorem 4. MD-VSC^0 = VNC^1

Proof. Let C be a multiplicatively disjoint arithmetic circuit of width w and size s. Let $X = \{x_1, x_2, \ldots, x_n\}$ be the set of variables in the circuit. Construct a new circuit by replacing j^{th} occurrence of x_i by a new variable $y_{i,j}$, for all *i.j*. Note that C' is a circuit with at most ns many variables and of size s and width w. Also, as C is multiplicatively disjoint, C' is syntactic multilinear in the variables $Y = \{y_{1,1}, y_{1,2} \ldots, y_{n,s}\}$. Now applying the construction of [6], we get an arithmetic formula of depth $O(w^2 \log s)$ and size $O(2^{w^2} s^{25w})$, but as w a constant, we get the required formula by replacing the $y_{i,j}$s with x_is. Now the equivalence follows from [1]. □

Remark 1. Note that the above theorem does not already give Theorem 2, as the only known procedure to convert an arithmetic formula into an equivalent bounded width branching program of [1] does not preserve syntactic multilinearity (For an example see [6]).

We strengthen the above result by giving a direct construction of BPs from multiplicatively disjoint circuits.

Lemma 4. *C be a multiplicatively disjoint arithmetical circuit of width w and size s as given by Lemma 1. Let $f_1, \ldots, f_w$ be the output gates of C. Then there exists an equivalent arithmetical branching program $[C]$ of width $O(w^2)$, length $O(s^w)$, and size $O(w^2 s^w)$. $[C]$ has a single start node b and and terminal nodes $[f_1], \ldots, [f_w], v$, and will be endowed with a mainline ending in v. Moreover, if C is syntactic multilinear then so is $[C]$.*

Proof. The proof is similar to that of Lemma 2. We proceed by induction on both s and w. If $s = 2$, the lemma holds trivially. If $w = 1$, C is a skew-circuit, and hence can be seen as a BP of width 3 (by adding a mainline).

Let $s > 2$ and $w > 2$ be given, and assume that C has at least 2 layers. Suppose, by induction hypothesis that the lemma holds for all circuits of size s' and w', where either $s' < s$ and $w' \leq w$ or $s' \leq s$ and $w' < w$.

Let C' be the sub-circuit obtained by deleting $f_1, \ldots, f_w$. Let $G = \{g_1, \ldots, g_w\}$ be the output gates of C'. Wlog. let $f_1 = g_1 \times g_2$ be the only multiplication gate at the output layer of C. Let D denote the sub-circuit rooted at g_1. Since C is multiplicatively disjoint, we have either width$(D) \leq w - 1$ or width$(E) \leq w - 1$. Wlog. assume that width$(D) \leq w - 1$.

Let $s' =$ size(C'), $s_D =$ size(D), $w' =$ width(C'), and $w_D =$ width(D). By induction hypothesis, we obtain BPs $[C']$ and $[D]$. $[C']$ has $w + 1$ output nodes, namely $[g_1], \ldots, [g_w], v$. $[D]$ has two output nodes $[g'_1]$ and v'.

Now construct the BP $[C]$ with output nodes $[f_1], \ldots, [f_w], v$ by composing $[C']$ before $[D]$ as follows: For all $i \geq 2$, connect $[g_j]s$ to $[f_i]s$ according the edges in the circuit C, i.e edge $([g_j], [f_i])$ is in $[C]$ iff g_j is an input for f_i. In case f_i is an input gate, draw an appropriately labeled edge from v. Put an edge from $[g_2]$ to $[f_1]$. Now identify the start node of $[D]$ with $[f_1]$ and re-label the terminal node of $[D]$ as $[f_1]$. Do the necessary staggering to carry on the values $f_2, \ldots, f_w$ to the last layer. We also pipe the mainline of $[C']$ into the mainline of $[D]$.

Analysis: By induction hypothesis, we have length$([C']) \leq s'^{w'} \leq (s - w)^w$ as $s' = s - w$ and $w' \leq w$. Furthermore, width$([C']) \leq w'^2 + 1 \leq w^2 + 1$, length$([D]) \leq s_D^{w_D} \leq (s - w)^{w-1}$, and width$([D]) \leq (w - 1)^2 + 1$ as $s_D \leq s - w$ and $w_D \leq w - 1$.

Now by the construction, width$([C]) = \max\{\text{width}([C'], \text{width}([D]) + w - 1\} \leq \max\{w^2 + 1, (w-1)^2 + w - 1\} \leq w^2 + 1$. Hence, length$([C]) =$ length$([C']) +$ length$([D]) \leq (s-w)^w + (s-w)^{w-1} \leq s^w$, for $w > 2$ and $w < s$. Hence size$([C]) = (w^2+1)s^w$. It is easy to see that this construction preserves the syntactic multilinearity property. □

4.1 Proof of Theorem 2

Given a syntactically multilinear circuit C of width w and size s, we first replace all the wires carrying only ring constants in C by new variables (as done in [6]), to get a circuit D of width $w_d \leq w^2$ and size $s_d \leq ws$. Note that the circuit D is multiplicatively disjoint. By Lemma 4 we get a syntactic multilinear BP $[D]$ of width $w_d^2 + 1$ and size s_d^w. Now replacing the introduced variables by the original constants they represent, we get the required syntactic multilinear BP $[C]$. □

Remark 2. By closer inspection of how Lemma 4 deals with input gates one can actually conclude width$([D]) \leq w^2 + 1$ and size$([D]) \leq O(s^w)$ in the above.

5 Coefficient Functions

Let f be a polynomial over variables $X = \{x_1, x_2, \ldots, x_n\}$. For a monomial m in variables from X, the partial coefficient function *coef*(f, m) is defined to be the unique polynomial g for which there exists polynomial h with none of its monomials divisible by m such that $f = mg + h$.

Malod studies the complexity of computing coefficient functions computed by class of arithmetic circuits [7]. From an old observation by Hammon, it can be seen that the permanent polynomial equals *coef*$(f, y_1 y_2 \ldots y_n)$, where f is given by the following depth three formula $f = \prod_{i \in [n]} \sum_{j \in [n]} x_{ij} y_j$. In [7] it is shown that Hamiltonian polynomial can be represented as a coefficient of a polynomial g computed by polynomial size arithmetic circuits. A closer inspection shows that this g is actually in VBP. Thus the arithmetic circuit classes which are at least as powerful as VAC^0 can generate VNP complete polynomials as coefficient functions, and hence the coefficient functions in general are hard.

In the case of polynomials computed by syntactic multilinear circuits we will prove that the situation is markedly different compared to the general case. For a multilinear polynomial f over variables $x_1, x_2, \ldots, x_n$, we define coefficient function *mcoef*$(f, \cdot)$ by *mcoef*$(f, a_1, a_2, \ldots, a_n) =$ *coef*$(f, x_1^{a_1} x_2^{a_2} \ldots x_n^{a_n})$, for any $a \in \{0,1\}^n$. Corresponding to *mcoef*$(f, \cdot)$ is a unique multilinear polynomial $g(x, e)$ in variables from X and E, such that for all $a \in \{0,1\}^n$, $g(x, a) =$ *mcoef*(f, a). Per abuse of notation we will denote this g by *mcoef*(f, a).

5.1 Closure Property

A syntactically multilinear complexity class sm-V$\mathcal{C}$ is said to be *closed under taking coefficients*, if for any $f \in$ sm-V$\mathcal{C}$, *mcoef*$(f, e) \in$ sm-V$\mathcal{C}$. We have the following identities: for any polynomials f and g,

$$mcoef(f + g, e) = mcoef(f, e) + mcoef(g, e). \tag{1}$$

For polynomials f and g with $\mathit{Var}(f) \cap \mathit{Var}(g) = \emptyset$,

$$\begin{aligned} \mathit{mcoef}(fg, e) = & \, \mathit{mcoef}(f, e)[\, e_i := 0 \text{ for } x_i \notin \mathit{Var}(f)] \cdot \\ & \mathit{mcoef}(g, e)[e_i := 0 \text{ for } x_i \notin \mathit{Var}(g)] \cdot \\ & \prod_{x_i \notin \mathit{Var}(f) \cup \mathit{Var}(g)} (1 - e_i) \end{aligned} \tag{2}$$

For individual variables and constants μ we have $\mathit{mcoef}(x_i) = (x_i(1 - e_i) + e_i) \prod_{j \in [n], j \neq i}(1 - e_j)$, and $\mathit{mcoef}(\mu) = \mu \prod_{j \in [n]}(1 - e_j)$.

Theorem 5. *Each of the following syntactically multilinear classes is closed under taking coefficients:* sm-VP, sm-VBP, sm-VNC^1, sm-VSC^i, sm-VBWBP*, and* sm-VAC^i*, for all* $i \geq 0$.

Proof sketch. For formula classes sm-VNC^1 and AC^i it is immediately clear how to convert a formula Φ computing f into a formula computing $\mathit{mcoef}(f, e)$ using Equations (1) and (2). For circuits one has to ensure the substitution at a multiplication gate $g = g_1 \times g_2$ using Equation (2) are consistent with other uses of g_1 and g_2. This can be guaranteed by first leveling the circuit with alternating layers of multiplication and addition gates. □

Consequently, we have no analogue of Hammon's observation for the permanent with $f \in$ sm-VNC$_1$ by [9].

Corollary 3. *Permanent (and also Determinant) cannot be expressed as a coefficient of some monomial of a polynomial computed by a syntactically multilinear arithmetic formula of polynomial size.*

5.2 Stability

Following [7], we say a complexity class sm-V$\mathcal{C}$ is *stable for coefficient functions*, if sm-V$\mathcal{C}$ is closed under taking coefficients and whenever $\mathit{mcoef}(f, e) \in$ sm-V$\mathcal{C}$, then $f \in$ sm-V$\mathcal{C}$. For a multilinear polynomial $f(x, e)$, let $\Sigma(E)\ f$ denote $\sum_{b \in \{0,1\}^m} f(x, b)$. We say a complexity class sm-V$\mathcal{C}$ is *closed under taking exponential sums*, if whenever $f(x, e) \in$ sm-V$\mathcal{C}$, then $\Sigma(E) f \in$ sm-V$\mathcal{C}$. Again the situation is contrary to the non-multilinear case, e.g. one can obtain the permanent as $\Sigma(E)\ f$, for $f \in$ VNC^1 [12], cf. [3].

Theorem 6. *The following are closed under exponential sums, and hence stable for coefficient functions:* sm-VP, sm-VBP, sm-VNC^1, sm-VSC^i, sm-VBWBP*, and* sm-VAC^i*, for all* $i \geq 0$.

The theorem will follow from the following straightforward proposition by patching a given circuit at gates with constant multiplications of appropriate powers of two. A proof will appear in the full version of the paper.

Proposition 1. *Let f and g be multilinear polynomials over X and E. We have that*

$$\Sigma(E)\ (f+g) = 2^a \Sigma(\mathit{Var}(f) \cap E)\ f + 2^b \Sigma(\mathit{Var}(g) \cap E)\ g,$$

where $a = |E - \mathit{Var}(f)|$ and $b = |E - \mathit{Var}(g)|$. Furthermore, if f and g are defined on disjoint variables sets,

$$\Sigma(E)\ fg = 2^c \Sigma(\mathit{Var}(f) \cap E)\ f \cdot \Sigma(\mathit{Var}(g) \cap E)\ g,$$

where $c = m - |\mathit{Var}(f) \cup \mathit{Var}(g)|$.

6 Open Problems

Regarding upper bounds, we pose the following four questions:

- Is sm-VSC^1 $\subseteq$ sm-VBP ?
- Is weaklyskew-sm-VSC^1 $\subseteq$ LWBP?
- Is V*s*SC^0 $\subseteq$ VBP ?
- Can we preserve width in Theorem 1 of [8]? If so, VsSC^0 = VNC^1.

Regarding lower bounds, note that constant width syntactically multilinear arithmetical circuits require size $n^{\Omega(\log n)}$ to compute the permanent, due to Theorem 2 and Raz's lower bound for multilinear formulas [9]. The best known multilinear formula for the permanent is given by Ryser, which is of size $O(n2^n)$ [11]. Raz and Yehudayoff strengthen the lower bound to $2^{n^{\Omega(1/d)}}$, for syntactically multilinear formulas with product depth d [10]. The latter can be simulated within sm-VBWBP, and this class appears to be the appropriate next place in line to strengthen the $n^{\Omega(\log n)}$ bound. One view of Theorem 2 is that it presents an obstacle to this, i.e. despite its conceptual simplicity, the class sm-VBWBP contains more than perhaps expected. Can we at least prove the desired strengthened lower bounds for width three syntactically multilinear BPs ?

Acknowledgments. The second author thanks Meena Mahajan for enlightening discussions and useful feedback.

References

1. Ben-Or, M., Cleve, R.: Computing algebraic formulas using a constant number of registers. SIAM J. Comput. 21(1), 54–58 (1992)
2. Brent, R.P.: The parallel evaluation of arithmetic expressions in logarithmic time. In: Proc. Symp., Carnegie-Mellon Univ., Pittsburgh, pp. 83–102 (1973)
3. Bürgisser, P.: Completeness and Reduction in Algebraic Complexity Theory. Algorithms and Computation in Mathematics. Springer, Heidelberg (2000)
4. Caussinus, H., McKenzie, P., Thérien, D., Vollmer, H.: Nondeterministic NC^1 computation. Journal of Computer and System Sciences 57, 200–212 (1998)
5. Jansen, M.J.: Lower bounds for syntactically multilinear algebraic branching programs. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 407–418. Springer, Heidelberg (2008)

6. Mahajan, M., Raghavendra Rao, B.V.: Arithmetic circuits, syntactic multilinearity, and the limitations of skew formulae. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 455–466. Springer, Heidelberg (2008)
7. Malod, G.: The complexity of polynomials and their coefficient functions. In: IEEE Conference on Computational Complexity, pp. 193–204 (2007)
8. Malod, G., Portier, N.: Characterizing valiant's algebraic complexity classes. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 704–716. Springer, Heidelberg (2006)
9. Raz, R.: Multi-linear formulas for permanent and determinant are of super-polynomial size. In: STOC, pp. 633–641 (2004)
10. Raz, R., Yehudayoff, A.: Lower bounds and separations for constant depth multilinear circuits. In: CCC 2008, pp. 128–139 (2008)
11. Ryser, H.J.: Combinatorial mathematics. In: Carus Mathematical Monograph, vol. 14, p. 23. Mathematical Association of America, Washington (1963)
12. Valiant, L.G.: Reducibility by algebraic projections. L'Enseignement mathématique 28, 253–268 (1982)
13. Valiant, L.G., Skyum, S., Berkowitz, S., Rackoff, C.: Fast parallel computation of polynomials using few processors. SIAM J. Comput. 12(4), 641–644 (1983)

One-Nonterminal Conjunctive Grammars over a Unary Alphabet

Artur Jeż[1,⋆] and Alexander Okhotin[2,3,⋆⋆]

[1] Institute of Computer Science, University of Wrocław, Poland
aje@ii.uni.wroc.pl
[2] Academy of Finland
[3] Department of Mathematics, University of Turku, Finland
alexander.okhotin@utu.fi

Abstract. Conjunctive grammars over an alphabet $\Sigma = \{a\}$ are studied, with the focus on the special case with a unique nonterminal symbol. Such a grammar is equivalent to an equation $X = \varphi(X)$ over sets of natural numbers, using union, intersection and addition. It is shown that every grammar with multiple nonterminals can be encoded into a grammar with a single nonterminal, with a slight modification of the language. Based on this construction, the compressed membership problem for one-nonterminal conjunctive grammars over $\{a\}$ is proved to be EXPTIME-complete, while the equivalence, finiteness and emptiness problems for these grammars are shown to be undecidable.

1 Introduction

Conjunctive grammars are an extension of the context-free grammars with an explicit intersection operation, introduced by Okhotin [9]. These grammars are characterized by language equations of the form

$$\begin{cases} X_1 = \varphi_1(X_1, \ldots, X_n) \\ \quad\vdots \\ X_n = \varphi_n(X_1, \ldots, X_n) \end{cases} \tag{*}$$

in which the right-hand sides φ_i may contain union, intersection and concatenation of languages, as well as singleton constants; in context-free grammars, intersection is not allowed. Despite their higher expressive power compared to context-free grammars, conjunctive grammars still possess efficient parsing algorithms and can be parsed in $DTIME(n^3) \cap DSPACE(n)$ [10], which makes them potentially useful for practical use.

Consider the special case of a *unary alphabet* $\Sigma = \{a\}$. All unary context-free languages are known to be regular. The question of whether conjunctive grammars can generate any non-regular unary languages has been an open problem

⋆ Supported by MNiSW grants N206 024 31/3826 2006–2009 and N N206 259035 2008–2010.

⋆⋆ Supported by the Academy of Finland under grant 118540.

for some years [10], until recently solved by Jeż [2], who constructed a conjunctive grammar for the language $\{a^{4^n} | n \geqslant 0\}$. This result was followed by a general theorem due to the authors [3], which asserts representability of a large class of unary languages by conjunctive grammars.

As strings over a one-letter alphabet can be regarded as natural numbers, language equations (*) corresponding to unary conjunctive grammars accordingly become *equations over sets of numbers*, in which concatenation of languages is represented by element-wise addition of sets defined as $S + T = \{m + n \mid m \in S,\ n \in T\}$. Such equations can be regarded as a generalization of *integer expressions*, introduced by Stockmeyer and Meyer [13] and further studied by McKenzie and Wagner [8]. In particular, the membership problem for integer expressions with the operations of union and addition is NP-complete [13], and allowing intersection makes it PSPACE-complete [8]. For equations over sets of numbers of the form (*) and using the same operations, the membership problem is NP-complete without intersection [1] and EXPTIME-complete with intersection [4]. The latter result was established by constructing a conjunctive grammar that generates unary notation of all numbers in a certain EXPTIME-complete set. The results on unary conjunctive grammars have further led to realising the computational completeness of more general equations over sets of natural numbers [5].

This paper investigates a special case of unary conjunctive grammars containing a unique nonterminal symbol. This subfamily has recently been studied by Okhotin and Rondogiannis [11], who constructed a one-nonterminal conjunctive grammar generating a non-periodic language, as well as presented two classes of conjunctive languages that are not representable by any grammars with a single nonterminal. The goal of this paper is to generalize the example of a nonregular language given by Okhotin and Rondogiannis [11] to a general representability theorem, and to explore its implications on decidability and complexity.

The main result of the paper, established in Section 3, is that for every unary conjunctive grammar the languages generated by all of its nonterminal symbols can be encoded together in a single unary language generated by a one-nonterminal conjunctive grammar. This construction is used in Section 4 to show that the EXPTIME-completeness result for unary conjunctive grammars [4] holds already for one-nonterminal grammars. In Section 5, the new theorem is used to demonstrate that equivalence, finiteness and co-finiteness problems are already undecidable for one-nonterminal unary conjunctive grammars. At the same time, equality to a constant language is shown to be decidable for a large class of constants, in contrast to the multiple-nonterminal case where it is undecidable for every fixed conjunctive constant [3].

2 Conjunctive Grammars and Systems of Equations

Conjunctive grammars generalize context-free grammars by allowing an explicit conjunction operation in the rules.

Definition 1 (Okhotin [9]). *A conjunctive grammar is a quadruple $G = (\Sigma, N, P, S)$, in which Σ and N are disjoint finite non-empty sets of terminal and nonterminal symbols respectively; P is a finite set of grammar rules, each of the form*

$$A \to \alpha_1 \& \ldots \& \alpha_n \quad (\textit{with } A \in N,\ n \geqslant 1 \textit{ and } \alpha_1, \ldots, \alpha_n \in (\Sigma \cup N)^*),$$

while $S \in N$ is a nonterminal designated as the start symbol.

The semantics of conjunctive grammars may be defined either by term rewriting [9], or, equivalently, by a system of language equations. According to the definition by language equations, conjunction is interpreted as intersection of languages as follows.

Definition 2 ([10]). *Let $G = (\Sigma, N, P, S)$ be a conjunctive grammar. The associated system of language equations is the following system in variables N:*

$$A = \bigcup_{A \to \alpha_1 \& \ldots \& \alpha_n \in P} \ \bigcap_{i=1}^{n} \alpha_i \quad (\textit{for all } A \in N)$$

Let $(\ldots, L_A, \ldots)$ be its least solution and denote $L_G(A) := L_A$ for each $A \in N$. Define $L(G) := L_G(S)$.

If the right-hand side of the equation is taken as an operator on vectors of languages, then the least fixed point of this operator is the least solution of the system with respect to componentwise inclusion.

An equivalent definition of conjunctive grammars is given via *term rewriting*, which generalizes the string rewriting used by Chomsky to define context-free grammars.

Definition 3 ([9]). *Given a conjunctive grammar G, consider terms over concatenation and conjunction with symbols from $\Sigma \cup N$ as atomic terms. The relation $\Longrightarrow$ of immediate derivability on the set of terms is defined as follows:*

- *Using a rule $A \to \alpha_1 \& \ldots \& \alpha_n$, a subterm $A \in N$ of any term $\varphi(A)$ can be rewritten as $\varphi(A) \Longrightarrow \varphi(\alpha_1 \& \ldots \& \alpha_n)$.*
- *A conjunction of several identical strings can be rewritten by one such string: $\varphi(w \& \ldots \& w) \Longrightarrow \varphi(w)$, for every $w \in \Sigma^*$.*

The language generated by a term φ is $L_G(\varphi) = \{w \mid w \in \Sigma^*,\ \varphi \Longrightarrow^* w\}$. The language generated by the grammar is $L(G) = L_G(S) = \{w | w \in \Sigma^*,\ S \Longrightarrow^* w\}$.

Consider the following grammar generating the language $\{a^{4^n} \mid n \geqslant 0\}$, which was the first example of a unary conjunctive grammar representing a non-regular language.

Example 1 (Jeż [2]). The conjunctive grammar

$$\begin{aligned}
A_1 &\to A_1A_3 \& A_2A_2 \mid a \\
A_2 &\to A_1A_1 \& A_2A_6 \mid aa \\
A_3 &\to A_1A_2 \& A_6A_6 \mid aaa \\
A_6 &\to A_1A_2 \& A_3A_3
\end{aligned}$$

with the start symbol A_1 generates the language $L(G) = \{a^{4^n} \mid n \geqslant 0\}$. In particular, $L_G(A_i) = \{a^{i \cdot 4^n} \mid n \geqslant 0\}$ for $i = 1, 2, 3, 6$.

The construction in Example 1 essentially uses all four nonterminal symbols, and there seems to be no apparent way to replicate it using a single nonterminal. However, this was achieved in the following example:

Example 2 (Okhotin, Rondogiannis [11]). The conjunctive grammar

$$S \to a^{22}SS\&a^{11}SS \mid a^9SS\&aSS \mid a^7SS\&a^{12}SS \mid a^{13}SS\&a^{14}SS \mid a^{56} \mid a^{113} \mid a^{181}$$

generates the language $\{a^{4^n-8} \mid n \geqslant 3\} \cup \{a^{2\cdot 4^n-15} \mid n \geqslant 3\} \cup \{a^{3\cdot 4^n-11} \mid n \geqslant 3\}$ $\cup \{a^{6\cdot 4^n-9} \mid n \geqslant 3\}$.

This grammar is actually derived from Example 1, and the language it generates encodes the languages of all four nonterminals in Example 1. Each of the four components in the generated language represents one of the nonterminals in Example 1 with a certain *offset* (8, 15, 11 and 9).

Note that the set from Example 2 is exponentially growing. At the same time, it has been proved that if a set grows faster than exponentially (for example, $\{n! \mid n \geqslant 1\}$), then it is not representable by univariate equations:

Proposition 1 (Okhotin, Rondogiannis [11]). *Let $S = \{n_1, n_2, \ldots, n_i, \ldots\}$ with $0 \leqslant n_1 < n_2 < \ldots < n_i < \ldots$ be an infinite set of numbers, for which $\liminf_{i\to\infty} \frac{n_i}{n_{i+1}} = 0$. Then S is not the least solution of any equation $X = \varphi(X)$.*

On the other hand, it is known that unary conjunctive grammars can generate a set that grows faster than any given recursive set:

Proposition 2 (Jeż, Okhotin [3]). *For every recursively enumerable set of natural numbers S there exists a conjunctive grammar G over an alphabet $\{a\}$, such that the set $\widehat{S} = \{n \mid a^n \in L(G)\}$ grows faster than S, in the sense that the n-th smallest number of $\widehat{S}$ is greater than the n-th smallest number of S, for all $n \geqslant 1$.*

Thus one-nonterminal conjunctive grammars are weaker in power than arbitrary unary conjunctive grammars. However, even though one-nonterminal conjunctive grammars cannot generate *all* unary conjunctive languages, it will now be demonstrated that they can represent a certain encoding of every conjunctive language.

3 One-Nonterminal Conjunctive Grammars

The goal is to simulate an arbitrary conjunctive grammar over $\{a\}$ by a conjunctive grammar with a single nonterminal symbol. The construction formalizes and elaborates the intuitive idea of Example 2, making it provably work for any grammar.

The first step towards the construction is a small refinement of the known normal form for conjunctive grammars. It is known that every conjunctive language

$L \subseteq \Sigma^+$ over any alphabet Σ can be generated by a conjunctive grammar in the *binary normal form* [9,10], with all rules of the form $A \to B_1C_1\& \ldots \& B_nC_n$ with $n \geqslant 1$ or $A \to a$. The following stronger form is required by the below construction.

Lemma 1. *For every conjunctive grammar $G = (\Sigma, N, P, S)$ with $\varepsilon \notin L(G)$ there exists a conjunctive grammar $G' = (\Sigma, N', P', S')$ generating the same language, in which every rule is of the form $A \to a$ with $a \in \Sigma$, or*

$$A \to B_1C_1\& \ldots \& B_nC_n \quad (\textit{with } n \geqslant 2),$$

in which the sets $\{B_1, C_1\}$, …, $\{B_n, C_n\}$ are pairwise disjoint.

A grammar in the binary normal form can be easily converted this stronger form by making multiple copies of each nonterminal.

The next theorem is the core result of this paper.

Theorem 1. *For every unary conjunctive grammar $G = (\{a\}, \{A_1, \ldots, A_m\}, P, A_1)$ of the form given in Lemma 1 there exist numbers $0 < d_1 < \ldots < d_m < p$ depending only on m and a conjunctive grammar $G' = (\{a\}, \{B\}, P', B)$ generating the language $L(G') = \{a^{np-d_i} \mid 1 \leqslant i \leqslant m,\ a^n \in L_G(A_i)\}$.*

Accordingly, the corresponding equation $X = \varphi(X)$ over sets of natural numbers has a unique solution $S = \bigcup_{i=1}^{m} S_i$, where $S_i = \{np - d_i \mid a^n \in L_G(A_i)\}$.

Let $p = 4^{m+2}$ and let $d_i = \frac{p}{4} + 4^i$ for every nonterminal A_i. For every number $t \in \{0, \ldots, p\}$, the set $\{np - t \mid n \geqslant 0\}$ is called *track number* t. The goal of the construction is to represent each set S_i in the track d_i. The rest of the tracks should be empty.

For every rule $A_i \to A_{j_1}A_{k_1}\& \ldots \& A_{j_\ell}A_{k_\ell}$ in G, the new grammar G' contains the rule

$$B \to a^{d_{j_1}+d_{k_1}-d_i}BB\& \ldots \& a^{d_{j_\ell}+d_{k_\ell}-d_i}BB, \tag{1}$$

and for every rule $A_i \to a$ in G, let G' have a rule

$$B \to a^{p-d_i}.$$

The proof of correctness of the construction will be done in terms of equations over sets of numbers. The task is to prove that the unique solution of the equation corresponding to G' is $S = \bigcup_i S_i$, where $S_i = \{np - d_i \mid a^n \in L_G(A_i)\}$.

Each time X appears in the right-hand side of the equation, it is used in the context of an expression $\psi(X) = X + X + (d_i + d_j - d_k)$. The proof of the theorem is based upon the following property of these expressions.

Lemma 2. *Let $i, j, k, \ell \in \{1, \ldots, m\}$ with $\{i, j\} \cap \{k, \ell\} = \varnothing$. Then*

$$(S + S + d_i + d_j) \cap (S + S + d_k + d_\ell) = (S_i + S_j + d_i + d_j) \cap (S_k + S_\ell + d_k + d_\ell).$$

Proof. As addition is distributive over union and union is distributive over intersection,

$$(S + S + d_i + d_j) \cap (S + S + d_k + d_\ell) = \\ = \bigcup_{i',j',k',\ell'} (S_{i'} + S_{j'} + d_i + d_j) \cap (S_{k'} + S_{\ell'} + d_k + d_\ell)$$

It is sufficient to prove that if $\{i', j'\} \neq \{i, j\}$ or $\{k', \ell'\} \neq \{k, \ell\}$, then the intersection is empty. Consider any such intersection

$$(S_{i'} + S_{j'} + d_i + d_j) \cap (S_{k'} + S_{\ell'} + d_k + d_\ell) = \\ \big(\{np \mid a^n \in L(A_{i'})\} - d_{i'} + \{np \mid a^n \in L(A_{j'})\} - d_{j'} + d_i + d_j\big) \cap \\ \big(\{np \mid a^n \in L(A_{k'})\} - d_{k'} + \{np \mid a^n \in L(A_{\ell'})\} - d_{\ell'} + d_k + d_\ell\big),$$

and suppose it contains any number, which must consequently be equal to $d_i + d_j - d_{i'} - d_{j'}$ modulo p and to $d_k + d_\ell - d_{k'} - d_{\ell'}$ modulo p. As each d_t satisfies $\frac{p}{4} < d_t \leqslant \frac{p}{2}$, both offsets are strictly between $-\frac{p}{2}$ and $\frac{p}{2}$, and therefore they must be equal to each other:

$$d_i + d_j - d_{i'} - d_{j'} = d_k + d_\ell - d_{k'} - d_{\ell'}.$$

Equivalently, $d_i + d_j + d_{k'} + d_{\ell'} = d_k + d_\ell + d_{i'} + d_{j'}$, and since each d_t is defined as $\frac{p}{4} + 4^t$, this holds if and only if

$$4^i + 4^j + 4^{k'} + 4^{\ell'} = 4^k + 4^\ell + 4^{i'} + 4^{j'}.$$

Consider the largest of these eight numbers, let its value be d. Without loss of generality, assume that it is on the left-hand side. Then the left-hand side is greater than d. On the other hand, if no number on the right-hand side is d, then the sum is at most $4 \cdot \frac{d}{4} = d$. Thus at least one number on the right-hand side must be equal to d as well. Removing those two numbers and giving the same argument for the sum of 3, 2 and 1 summands yields that

$$\{d_i, d_j, d_{k'}, d_{\ell'}\} = \{d_k, d_\ell, d_{i'}, d_{j'}\}.$$

Then, by the assumption that $\{i, j\} \cap \{k, \ell\} = \varnothing$,

$$\{d_i, d_j\} = \{d_{i'}, d_{j'}\} \quad \text{and} \quad \{d_{k'}, d_{\ell'}\} = \{d_k, d_\ell\},$$

and since the addition is commutative,

$$i = i', \quad j = j', \quad k = k' \quad \text{and} \quad \ell = \ell'.$$

This completes the proof of the lemma. □

With this property established, it can be verified that every rule for every A_i in G is correctly simulated by the corresponding rule of G', and that the data from different tracks is never mixed.

Consider the equation $X = \varphi(X)$ over sets of numbers corresponding to G'. Every "long" rule $A \to \mathscr{A}$ in G, where $\mathscr{A} = A_{j_1}A_{k_1}\& \ldots \& A_{j_\ell}A_{k_\ell}$, is represented in the new grammar by a rule (1), which contributes the following subexpression to φ

$$\varphi_{i,\mathscr{A}}(S) = \bigcap_{t=1}^{\ell}(d_{j_t} + d_{k_t} - d_i) + S + S = \bigcap_{t=1}^{\ell}(d_{j_t} + d_{k_t} - d_i) + S_{j_t} + S_{k_t}$$

Altogether, the equation $X = \varphi(X)$ takes the following form:

$$X = \bigcup_{A_i \to \mathscr{A} \in P} \varphi_{i,\mathscr{A}}(X) \cup \bigcup_{A_i \to a \in P} \{p - d_i\}$$

Now the task is to prove that the unique solution of this equation is $S = \bigcup_i S_i$, where $S_i = \{np - d_i \mid a^n \in L_G(A_i)\}$.

Consider each "long" rule $A_i \to \mathscr{A}$ with $\mathscr{A} = A_{j_1}A_{k_1}\& \ldots \& A_{j_t}A_{k_t}$. Then

$$\varphi_{i,\mathscr{A}}(S) = \bigcap_{t=1}^{\ell}(d_{j_t} + d_{k_t} - d_i) + S + S = \bigcap_{t=1}^{\ell}(d_{j_t} + d_{k_t} - d_i) + S_{j_t} + S_{k_t}$$

by Lemma 2, and it is easy to calculate that

$$\bigcap_{t=1}^{\ell}(d_{j_t} + d_{k_t} - d_i) + S_{j_t} + S_{k_t} = \{np - d_i \mid a^n \in L_G(\mathscr{A})\}.$$

Similarly, for a "short" rule $A_i \to a$, $\{p - d_i\} = \{np - d_i \mid a^n \in L_G(\{a\})\}$, and altogether,

$$\varphi(S) = \bigcup_i \bigcup_{A_i \to \mathscr{B} \in P} \{np - d_i \mid a^n \in L_G(\mathscr{B})\} = \bigcup_i \{np - d_i \mid a^n \in L_G(A_i)\} = S,$$

where the inner equality is due to the fact that $(\ldots, L_G(A_i), \ldots)$ is the solution of the system of language equations associated to G. This completes the proof of Theorem 1.

For an example of this transformation, consider the four-nonterminal grammar from Example 1. It satisfies the condition in Lemma 1, but it is not precisely in the binary normal form, as it contains rules $A_2 \to aa$ and $A_3 \to aaa$. However, these rules do not affect the general construction, and one can extend the transformation of Theorem 1 to this grammar as follows:

Example 3. For the grammar in Example 1, the constants are $m = 4$, $p = 4^{m+2} = 4^6 = 4096$, $d_1 = 1028$, $d_2 = 1040$, $d_3 = 1088$ and $d_4 = 1280$, and the transformation yields the grammar

$$S \to a^{1088}SS\&a^{1052}SS \mid a^{1016}SS\&a^{1280}SS \mid a^{980}SS\&a^{1472}SS \mid \\ a^{788}SS\&a^{896}SS \mid a^{3068} \mid a^{7152} \mid a^{11200}$$

generating the language

$$\{a^{4^n-1028} \mid n \geqslant 6\} \cup \{a^{2\cdot 4^n-1040} \mid n \geqslant 6\} \cup \{a^{3\cdot 4^n-1088} \mid n \geqslant 6\} \cup \{a^{6\cdot 4^n-1280} \mid n \geqslant 6\}.$$

4 Complexity of the Membership Problem

The *(general) membership problem* for a family of grammars is stated as follows: "Given a string w and a grammar G, determine whether $w \in L(G)$". The membership problem is P-complete both for context-free and conjunctive grammars.

A variant of this problem, which received considerable attention in the recent years, is the *compressed membership problem* [12], where the string w is given as a context-free grammar G_w generating $\{w\}$. This problem is PSPACE-complete for context-free grammars [7,12] and EXPTIME-complete for conjunctive grammars [4]. For unary languages, the compressed representation of a^n is its binary notation of length $\Theta(\log n)$. In this case the problem is NP-complete for context-free grammars [1,12], but still EXPTIME-complete for conjunctive grammars [4].

Proposition 3 (Jeż, Okhotin [4]). *There exists an EXPTIME-complete set of numbers $S \subseteq \mathbb{N}$, such that the language $L = \{a^n \mid n \in S\}$ is generated by a conjunctive grammar. The compressed membership problem for conjunctive grammars over a unary alphabet is EXPTIME-complete.*

To show that both results still hold for one-nonterminal unary conjunctive grammars, it is sufficient to take the grammar generating L and to transform it according to Theorem 1.

Theorem 2. *There exists an EXPTIME-complete set of numbers $S \subseteq \mathbb{N}$, such that the language $L = \{a^n \mid n \in S\}$ is generated by a one-nonterminal conjunctive grammar. The compressed membership problem for one-nonterminal unary conjunctive grammars is EXPTIME-complete.*

For unary context-free grammars, the compressed membership problem has different complexity depending on the number of nonterminals. For multiple nonterminals it is NP-complete [12]. However, in the one-nonterminal case an efficient algorithm for solving this problem can be obtained using the following property:

Lemma 3. *Let $G = (\{a\}, \{S\}, P, S)$ be a one-nonterminal context-free grammar with m rules and with the longest right-hand side of a rule of length k. Then $L(G)$ is periodic starting from $\widehat{n} \leqslant 2mk^3(2k+1)$ with a period at most $2k^2$.*

First it is shown that from any single pair of rules $S \to S^{k_i}a^{\ell_i}$ and $S \to a^{\ell_j}$ one can deduce that $p = (k_i - 1)\ell_j + \ell_i$ is a period of $L(G)$, which gives an upper bound on the least period. As the language $L(G)$ is over a unary alphabet, a derivation is determined by the number of occurrences of each rule, and it is shown that a derivation exists if and only if those numbers satisfy a certain numerical condition. Using this representation, the derivation of every sufficiently long string can be reduced to obtain a derivation of a shorter string, which has the same length modulo p. Thus the periodicity starts from a short string.

Theorem 3. *The compressed membership problem for one-nonterminal unary CFGs is in NLOGSPACE.*

Table 1. Complexity of general membership problems

	uncompressed	compressed	fully compressed
Context-free			
general case	P-complete	PSPACE-complete [12]	n/a
$\Sigma = \{a\}$, any N	P-complete	NP-complete [1]	NP-complete [1]
$\Sigma = \{a\}$, $N = \{S\}$	**in NLOGSPACE**	**in NLOGSPACE**	**NP-complete**
Conjunctive			
general case	P-complete	EXPTIME-complete [4]	n/a
$\Sigma = \{a\}$, any N	P-complete	EXPTIME-complete [4]	EXPTIME-compl. [4]
$\Sigma = \{a\}$, $N = \{S\}$	in P	**EXPTIME-complete**	EXPTIME-compl. [4]

By Lemma 3, the period of $L(G)$ is small. Then the input string can be replaced with a shorter string of a logarithmic length, and a nondeterministic algorithm can guess its derivation, storing the compressed sentential form in logarithmic space.

Note that the grammars are supplied to the above algorithm uncompressed, that is, rules of the form $S \to a^\ell S^k$ are stored using $\ell + k$ symbols. Consider another variant of the problem, the *fully compressed membership problem*, in which the grammar is compressed as well, and a rule $S \to a^\ell S^k$ is given by binary notations of ℓ and k. This problem is hard already for one nonterminal:

Theorem 4. *The fully compressed membership problem for one-nonterminal unary CFGs is NP-complete.*

As in the NP-hardness result for integer expressions by Stockmeyer and Meyer [13, Thm. 5.1], the proof is by encoding an instance of the *knapsack problem* in a grammar, with the sizes of the objects compressed. The knapsack problem is stated as follows: "Given integers $b_1, \ldots, b_n$ and z in binary notation, determine whether there exist $c_1, \ldots, c_n \in \{0, 1\}$ with $\sum_{i=1}^{n} b_i c_i = z$". Given an instance of this problem, assume that $z > \max_i b_i$, and let m be the least power of two with $m \geqslant \max\{z, 2^n\} + 2$. Then the grammar G with the following rules is constructed: $S \to S^n$, $S \to a^{m^2+2^{i-1}}$ and $S \to a^{m^2+mb_i+2^{i-1}}$ for $1 \leqslant i \leqslant n$. It can be proved that the string $w = a^{nm^2+zm+(2^n-1)}$ is in $L(G)$ if and only if the numbers $c_1, \ldots, c_n \in \{0, 1\}$ with $\Sigma_{i=1}^{n} b_i c_i = z$ do exist.

The complexity of different cases is summarized in Table 1.

5 Decision Problems

Having established the complexity of the membership problem, let us now consider some other basic properties of one-nonterminal unary conjunctive grammars. In the case of multiple nonterminals, most properties are undecidable:

Proposition 4 (Jeż, Okhotin [3]). *For every fixed unary conjunctive language $L_0 \subseteq a^*$, the problem of whether a given conjunctive grammar over $\{a\}$ generates the language L_0 is Π_1-complete.*

In contrast, in the case of one-nonterminal grammars, the equality to any fixed ultimately periodic set is clearly decidable: it is sufficient to substitute it into the equation and check whether it is turned into an equality. This approach extends to a larger class of fixed languages:

Theorem 5. *There exists an algorithm, which, given an equation $X = \varphi(X)$ using union, intersection, addition and ultimately periodic constants, and a finite automaton M over an alphabet $\Sigma_k = \{\mathtt{0}, \mathtt{1}, \ldots, k-1\}$, determines whether $\{n \mid$ the k-ary notation of n is in $L(M)\}$ is the least solution of the equation.*

In particular, the theorem applies to such languages as in Examples 1–3, which are recognized by finite automata in base-4 notation.

The algorithm works by substituting the set of numbers defined by M into the equation. The value of each subexpression is computed in the form of a finite automaton over Σ_k representing base-k notation. For Boolean operations this is clearly possible, while the addition of sets can be done symbolically on finite automata for their base-k representation according to the following lemma:

Lemma 4. *Let L_1 and L_2 be regular languages over $\Sigma_k = \{\mathtt{0}, \mathtt{1}, \ldots, k-1\}$, with $L_1 \cap \mathtt{0}\Sigma_k^* = L_2 \cap \mathtt{0}\Sigma_k^* = \varnothing$. Then the language $\{$the k-ary notation of $n_1 + n_2 \mid$ the k-ary notation of n_i is in $L_i\}$ over Σ_k is (effectively) regular.*

This shows that equality to a fixed language is decidable for one-nonterminal conjunctive grammars for a fairly large class of constants. At the same time, the more general problem of equivalence of two given grammars is undecidable.

Theorem 6. *The equivalence problem for one-nonterminal unary conjunctive grammars is Π_1-complete.*

The equivalence problem for unary conjunctive grammars with multiple nonterminals is Π_1-complete and the proof is by reduction from this problem. Two grammars G_1 and G_2 are combined into G, such that $L(G_1) = L_G(A_1)$ and $L(G_2) = L_G(A_2)$. Let G' be G with A_1 and A_2 exchanged. Once the construction of Theorem 1 is applied to G and G', the two resulting one-nonterminal grammars are equivalent if and only if the original grammars generate the same language.

Theorem 7. *The co-finiteness problem for one-nonterminal unary conjunctive grammars is Σ_1-complete.*

The problem is in Σ_1 by Theorem 5: an algorithm solving this problem can nondeterministically guess an NFA N recognizing a co-finite language and test whether the grammar generates $L(N)$. The hardness is proved by reduction from the non-emptiness problem for unary conjunctive grammars. The transformation of Theorem 1 is applied to a given grammar, resulting in a one-nonterminal grammar G', and then the following p extra rules are added to G':

$$S \to a^{d_1+i}X \& a^{d_2+i}X \quad (0 \leqslant i \leqslant p-1)$$

Table 2. Decision problems for grammars over $\{a\}$

	equiv. to reg. L_0	equivalence	finiteness	co-finiteness
Context-free				
general case	undecidable	undecidable	decidable	undecidable
$\Sigma = \{a\}$, any N	decidable	decidable	decidable	decidable
$\Sigma = \{a\}$, $N = \{S\}$	decidable	decidable	decidable	decidable
Conjunctive				
general case	undecidable	undecidable	undecidable	undecidable
$\Sigma = \{a\}$, any N	Π_1-complete [3]	Π_1-complete [3]	undecidable [3]	$\Sigma_1 \leqslant \cdot \leqslant \Sigma_2$
$\Sigma = \{a\}$, $N = \{S\}$	**decidable**	Π_1**-complete**	Σ_1**-complete**	Σ_1**-complete**

If any string appears on track d_1, these rules will generate all longer strings, thus "spamming" the language to make it co-finite.

Theorem 8. *The finiteness problem for one-nonterminal unary conjunctive grammars is Σ_1-complete.*

The problem is in Σ_1 for the same reason as in Theorem 7. The Σ_1-hardness is proved by reduction from the problem of whether $L(G) \neq a^+$ for a given unary conjunctive grammar G. Once G is transformed to a one-nonterminal grammar G' (as in the previous proof), an additional conjunct is added to each rule as follows: for every rule

$$S \to a^{\ell_1} SS \& \ldots \& a^{\ell_k} SS$$

created in Theorem 7, which generates a subset of $\{a^{np-d_i} \mid n \geqslant 1\}$ for some number i, the final grammar has the rule

$$S \to a^{\ell_1} SS \& \ldots \& a^{\ell_k} SS \& a^{p+d_1-d_i} S.$$

This additional conjunct acts as a filter: if any string in track d_1 is missing, then no strings of any greater length can be generated, making the generated language finite.

The level of undecidability of both finiteness and co-finiteness problems for conjunctive grammars with multiple nonterminals remains open, see the summary in Table 2.

References

1. Huynh, D.T.: Commutative grammars: the complexity of uniform word problems. Information and Control 57(1), 21–39 (1983)
2. Jeż, A.: Conjunctive grammars can generate non-regular unary languages. International Journal of Foundations of Computer Science 19(3), 597–615 (2008)
3. Jeż, A., Okhotin, A.: Conjunctive grammars over a unary alphabet: undecidability and unbounded growth. Theory of Computing Systems (to appear)
4. Jeż, A., Okhotin, A.: Complexity of equations over sets of natural numbers. In: 25th Annual Symposium on Theoretical Aspects of Computer Science (STACS 2008), Bordeaux, France, February 21-23, pp. 373–383 (2008)

5. Jeż, A., Okhotin, A.: On the computational completeness of equations over sets of natural numbers. In: 35th International Colloquium on Automata, Languages and Programming (ICALP 2008), Reykjavik, Iceland, July 7-11, pp. 63–74 (2008)
6. Kunc, M.: What do we know about language equations? In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) DLT 2007. LNCS, vol. 4588, pp. 23–27. Springer, Heidelberg (2007)
7. Lohrey, M.: Word problems and membership problems on compressed words. SIAM Journal on Computing 35(5), 1210–1240 (2006)
8. McKenzie, P., Wagner, K.W.: The complexity of membership problems for circuits over sets of natural numbers. Computational Complexity 16, 211–244 (2007)
9. Okhotin, A.: Conjunctive grammars. Journal of Automata, Languages and Combinatorics 6(4), 519–535 (2001)
10. Okhotin, A.: Nine open problems for conjunctive and Boolean grammars. Bulletin of the EATCS 91, 96–119 (2007)
11. Okhotin, A., Rondogiannis, P.: On the expressive power of univariate equations over sets of natural numbers. In: IFIP Intl. Conf. on Theoretical Computer Science (TCS 2008), Milan, Italy, September 8-10. IFIP, vol. 273, pp. 215–227 (2008)
12. Plandowski, W., Rytter, W.: Complexity of language recognition problems for compressed words. In: Karhumäki, J., Maurer, H.A., Păun, G., Rozenberg, G. (eds.) Jewels are Forever, pp. 262–272. Springer, Heidelberg (1999)
13. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time. In: STOC 1973, pp. 1–9 (1973)

Concatenation of Regular Languages and Descriptional Complexity[⋆]

Galina Jirásková

Mathematical Institute, Slovak Academy of Sciences,
Grešákova 6, 040 01 Košice, Slovakia
jiraskov@saske.sk

Abstract. We investigate the deterministic and nondeterministic state complexity of languages that can be obtained as the concatenation of two regular languages represented by deterministic and nondeterministic finite automata. In the nondeterministic case, we show that the whole range of complexities from 1 to $m + n$ can be obtained using a binary alphabet. In the deterministic case, we get the whole range of complexities from 1 to $m \cdot 2^n - 2^{n-1}$, however, to describe appropriate automata we use a growing alphabet.

1 Introduction

The state complexity of a regular language is the smallest number of states in any deterministic finite automaton (dfa) accepting the language. The nondeterministic state complexity of a regular language is defined as the smallest number of states in any nondeterministic finite automaton (nfa) for the language. The (nondeterministic) state complexity of an operation on regular languages is the smallest number of states that are sufficient and necessary in the worst case for a dfa (an nfa, respectively) to accept the language resulting from the operation.

Some early results on the state complexity of union, concatenation, star, and cyclic shift on languages defined by partial dfa's have been presented by Maslov in 1970 [20]. In the paper, some small inaccuracies occur, and, moreover, no rigorous formal proofs are provided. The systematic study of the state complexity of operations on regular languages began with works by Leiss [21], Birget [2,3], and Yu *et al.* [28]. A lot of results in this area have been obtained. In particular, the state complexity of operations on finite languages has been investigated by Campeanu *et al.* [4] and of operations on unary language by Pighizzini and Shallit [22]. Domaratzki [5] has studied proportional removals, and Salomaa *et al.* [24] have investigated reversals in different cases. The first results on the nondeterministic state complexity of operation on regular language have been published by Holzer and Kutrib [10], some of them have been obtained for binary languages in [18]. A comprehensive survey for the deterministic case has been given by Yu in [27].

⋆ Research supported by VEGA grant 2/0111/09.

Iwama *et al.* [13] stated the question of whether there always exists a minimal nondeterministic finite automaton of n states whose equivalent minimal deterministic finite automaton has exactly α states for all integers n and α satisfying $n \leqslant \alpha \leqslant 2^n$. The question has also been considered in [14], where an integer Z with $n < Z < 2^n$ is called a "magic number" if no dfa of Z states can be simulated by any nfa of n states. In [17] it has been shown that there are no magic numbers, that is, appropriate automata have been described for all integers n and α. However, the constructions use a growing alphabet of an exponential size. Later, in [7], the size of the alphabet has been reduced to $n + 2$, and finally, in [16], the result has been proved using a fixed four-letter alphabet. On the other hand, it is known that there are a lot of magic numbers in the unary case [8]. The cases of binary and ternary alphabets are still open.

A similar problem for complements of regular languages has been examined in [15]. Using a growing alphabet it has been proved that all values in the range from $\log n$ to 2^n can be obtained as the nondeterministic state complexity of the complement of an n-state nfa language. In [19], this result has been improved by showing that it holds also for a fixed five-letter alphabet. In the same paper, the deterministic and nondeterministic state complexity of stars and reversals of regular languages has been examined. In all cases, the whole range of complexities up to the known upper bounds have been obtained.

In this paper, we investigate the deterministic and nondeterministic state complexity of languages that can be obtained as the concatenation of two regular languages represented by deterministic and nondeterministic finite automata. We show that the nondeterministic state complexity of the concatenation of two languages represented by nfa's of m and n states, respectively, can be an arbitrary value in the range from 1 to $m+n$. To prove the result we use a binary alphabet (in fact, except for the case of $m + n$, the languages are even unary). In the deterministic case, we get the whole range of complexities from 1 to m in the case of $n = 1$, and the whole range from 1 to $m \cdot 2^n - 2^{n-1}$ in the case of $n \geqslant 2$. However, to describe suitable automata we use a growing alphabet.

To conclude this section, let us mention some other related works. Magic numbers for symmetric difference nfa's have been studied by Zijl [29]. In [11], it has been shown that the deterministic and nondeterministic state complexity of union and intersection of regular languages may reach each value from 1 up to the upper bounds mn or $m + n + 1$. Similar results for the nonterminal complexity of some operations on context-free languages have been recently obtained by Dassow and Stiebe [6].

2 Preliminaries

In this section, we give some basic definitions, notations, and preliminary results used throughout the paper. For further details, we refer to [25,26].

Let Σ be a finite alphabet and Σ^* the set of all strings over the alphabet Σ including the empty string ε. A *language* is any subset of Σ^*. We denote the cardinality of a finite set A by $|A|$ and its power-set by 2^A.

A *deterministic finite automaton* (dfa) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is a finite input alphabet, δ is the transition function that maps $Q \times \Sigma$ to Q, q_0 is the initial state, $q_0 \in Q$, and F is the set of accepting states, $F \subseteq Q$. In this paper, all dfa's are assumed to be complete, that is, the next state $\delta(q, a)$ is defined for each state q in Q and each symbol a in Σ. The transition function δ is extended to a function from $Q \times \Sigma^*$ to Q in a natural way. A string w in Σ^* is accepted by the dfa M if the state $\delta(q_0, w)$ is an accepting state of the dfa M. The *language accepted by* the dfa M, denoted $L(M)$, is the set of strings $\{w \in \Sigma^* \mid \delta(q_0, w) \in F\}$.

A *nondeterministic finite automaton* (nfa) is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where Q, Σ, q_0 and F are defined in the same way as for a dfa, and δ is the nondeterministic transition function that maps $Q \times \Sigma$ to 2^Q. It can be naturally extended to the domain $Q \times \Sigma^*$. A string w in Σ^* is accepted by the nfa M if the set $\delta(q_0, w)$ contains an accepting state of the nfa M. The *language accepted by* the nfa M is the set $L(M) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \varnothing\}$.

Two automata are said to be *equivalent* if they accept the same language. A dfa (an nfa) M is called *minimal* if all dfa's (all nfa's, respectively) that are equivalent to M have at least as many states as M. It is well-known that a dfa $M = (Q, \Sigma, \delta, q_0, F)$ is minimal if *(i)* all states are reachable from the initial state q_0, and *(ii)* no two different states are equivalent.

The *(deterministic) state complexity* of a regular language is the number of states in its minimal dfa. The *nondeterministic state complexity* of a regular language is defined as the number of states in a minimal nfa (with a single initial state) accepting this language. A regular language with deterministic (nondeterministic) state complexity n is called an n-state dfa language (an n-state nfa language, respectively).

Every nondeterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ can be converted to an equivalent deterministic finite automaton $M' = (2^Q, \Sigma, \delta', \{q_0\}, F')$ using an algorithm known as the "subset construction" [23]. The transition function δ' is defined by $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$, and a state R in 2^Q is an accepting state of the dfa M' if it contains at least one accepting state of the nfa M. The dfa M' need not be minimal since some states may be unreachable or equivalent. We call the dfa M' the *subset automaton* corresponding to the nfa M.

To prove that an nfa is minimal we use a fooling-set lower-bound technique [1,2,3,9,12]. We recall the definition of a fooling set, and the lemma from [2] describing this lower-bound technique.

Definition 1. *A set of pairs of strings $\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}$ is said to be a* fooling set *for a language L if for every i and j in $\{1, 2, \ldots, n\}$,*

(F1) *the string $x_i y_i$ is in the language L, and*

(F2) *if $i \neq j$, then at least one of the strings $x_i y_j$ and $x_j y_i$ is not in L.*

Lemma 1 (Birget [2]). *Let $\mathcal{A}$ be a fooling set for a regular language L. Then every nfa recognizing the language L requires at least $|\mathcal{A}|$ states.* □

3 Concatenation and Nondeterministic State Complexity

The nondeterministic state complexity of concatenation of regular languages is $m + n$, and this bound can be reached by the concatenation of the binary languages $\{a^m\}^*$ and $\{b^n\}^*$ [10]. Our first lemma shows that the nondeterministic state complexity of the concatenation of an m-state and an n-state unary nfa languages may reach an arbitrary value from 1 to $m+n-1$. Thus we obtain the whole range of complexities from 1 to $m+n$ in the nondeterministic case.

Lemma 2. *For all integers m, n and α such that $1 \leqslant \alpha \leqslant m+n-1$, there exist a minimal unary nfa A of m states and a minimal unary nfa B of n states such that every minimal nfa for the language $L(A)L(B)$ has α states.*

Proof. We will define automata A and B over a unary alphabet. Therefore, $L(A)L(B) = L(B)L(A)$, and, without loss of generality, we can assume $m \leqslant n$. Let us consider two cases: (1) $1 \leqslant \alpha < m$, and (2) $m \leqslant \alpha \leqslant m+n-1$.

(1) Let $1 \leqslant \alpha < m$. Let A be the m-state nfa for the language $\{\varepsilon\} \cup \{a^k \mid k \geqslant m\}$ and B be the n-state nfa accepting the language $\{\varepsilon\} \cup \{a^k \mid \alpha \leqslant k \leqslant n-1\}$ shown in Fig. 1. The set $\{(a^i, a^{m-i}) \mid i = 0, 1, \dots, m-1\}$ is a fooling set for the language $L(A)$ since the string a^m is in the language $L(A)$, while each string a^r with $0 < r < m$ is not in this language. Therefore, by Lemma 1, the nfa A is minimal. The nfa B is minimal as well since it accepts the string a^{n-1} but does not accept any longer string. We have $L(A)L(B) = \{\varepsilon\} \cup \{a^k \mid k \geqslant \alpha\}$. Every minimal nfa for the concatenation has α states.

(2) Let $m \leqslant \alpha \leqslant m+n-1$, that is, $\alpha = m+\beta$, where $0 \leqslant \beta \leqslant n-1$. Consider an m-state nfa for the language $\{a^k \mid k \geqslant m-1\}$, and an n-state nfa for the language $\{a^k \mid \beta \leqslant k \leqslant n-1\}$. The concatenation of these two languages is the language $\{a^k \mid k \geqslant m+\beta-1\}$ that requires $m+\beta$ nondeterministic states. □

As a corollary, we get the following result.

Theorem 1. *For all integers m, n and α such that $1 \leqslant \alpha \leqslant m+n$, there exist a minimal nondeterministic finite automaton A of m states and a minimal nondeterministic finite automaton B of n states such that every minimal nondeterministic finite automaton for the language $L(A)L(B)$ has α states. All values from 1 to $m+n-1$ can be reached by the concatenation of unary languages, the upper bound $m+n$ is attained by the concatenation of binary languages.* □

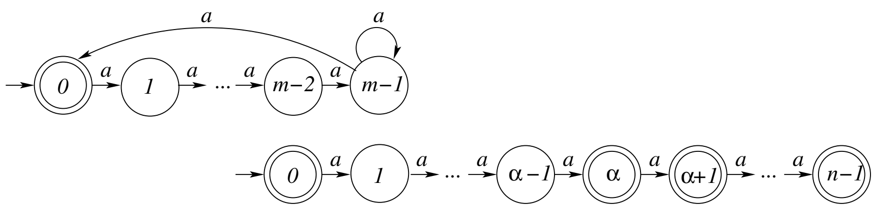


Fig. 1. The nondeterministic finite automata A and B; $1 \leqslant \alpha < m$

4 Concatenation and State Complexity

We now turn our attention to the state complexity of languages that can be obtained as the concatenation of an m-state and n-state dfa languages. The upper bound is known to be m in the case of $n = 1$ and $m \cdot 2^n - 2^{n-1}$ in the case of $n \geqslant 2$ [28]. These upper bounds can be reached by the concatenation of binary languages [15]. In this section, we prove that the whole range of complexities, up to these upper bounds, can be produced in the deterministic case. Our three lemmata cover the following three cases:

(1) $1 \leqslant \alpha \leqslant m$,
(2) $m < \alpha \leqslant m \cdot 2^{n-1}$,
(3) $m \cdot 2^{n-1} < \alpha \leqslant m \cdot 2^n - 2^{n-1} = (m-1) \cdot 2^n + 2^{n-1}$.

Proposition 1. *For all integers m, n and α with $1 \leqslant \alpha \leqslant m+n-1$, there exist a minimal unary dfa A of m states and a minimal unary dfa B of n states such that the minimal dfa for the language $L(A)L(B)$ has α states.*

Proof. Since automata A and B will be defined over a unary alphabet, we may assume $m \leqslant n$.

First, let $1 \leqslant \alpha < m$. Let A be the minimal m-state dfa for the language $\{\varepsilon\} \cup \{a^k \mid k \geqslant m-1\}$, and let B be the minimal n-state dfa for the language $\{\varepsilon\} \cup \{a^k \mid \alpha - 1 \leqslant k \leqslant n-2\}$. Then $L(A)L(B) = \{\varepsilon\} \cup \{a^k \mid k \geqslant \alpha - 1\}$. The minimal dfa for this concatenation has α states.

Now, let $m \leqslant \alpha \leqslant m+n-1$, that is $\alpha = m + \beta$, where $0 \leqslant \beta \leqslant n-1$. Let A be the minimal m-state dfa for the language $\{a^k \mid k \geqslant m-1\}$, and let B be the minimal n-state dfa for the language $\{a^k \mid \beta \leqslant k \leqslant n-2\}$, or, in the case of $\beta = n-1$, for the language $\{a^k \mid k \geqslant n-1\}$. The language $L(A)L(B) = \{a^k \mid k \geqslant m+\beta-1\}$ needs $m+\beta$ deterministic states. □

Thus we have shown the following lemma.

Lemma 3. *For all integers m, n and α with $1 \leqslant \alpha \leqslant m$, there exist a minimal unary dfa A of m states and a minimal unary dfa B of n states such that the minimal dfa for the language $L(A)L(B)$ has α states.* □

The next lemma deals with the case of $n \geqslant 2$ and $m < \alpha \leqslant m \cdot 2^{n-1}$.

Lemma 4. *For all integers m, n and α such that $n \geqslant 2$ and $m < \alpha \leqslant m \cdot 2^{n-1}$, there exist a minimal dfa A of m states and a minimal dfa B of n states, both defined over a growing alphabet, such that the minimal dfa for the language $L(A)L(B)$ has α states.*

Proof. If $m < \alpha \leqslant m \cdot 2^{n-1}$, then there is an integer k with $1 \leqslant k \leqslant m$, such that $(m-k) \cdot 2^{n-1} + k < \alpha \leqslant (m-(k-1)) \cdot 2^{n-1} + (k-1)$. Then

$$\alpha = (m-k) \cdot 2^{n-1} + k + \beta,$$

where $1 \leqslant k \leqslant m$ and $1 \leqslant \beta \leqslant 2^{n-1} - 1$.

We are going to define an m-state dfa A, and an n-state dfa B over a growing alphabet $\{a, b, c, d, f, e_1, e_2, \ldots, e_\beta\}$. The states in the dfa A will be divided into two groups, the first one will contain $m - k$ states $q_1, q_2, \ldots, q_{m-k}$, the second one will contain k states $p_1, p_2, \ldots, p_k$. The dfa B will have n states $1, 2, \ldots, n$. Then, we will construct an nfa C for the language $L(A)L(B)$ in a standard way, and apply the subset construction to this nfa to get a dfa C' for $L(A)L(B)$. The transitions in automata A and B will be defined in such a way that the dfa C' will have exactly $(m-k) \cdot 2^{n-1} + k + \beta$ (that is, α) reachable and pairwise inequivalent states. We will define transitions on symbols a and b so that all subsets $\{q_1\} \cup S$, where $S \subseteq \{1, 2, \ldots, n\}$ and $1 \in S$, will be reachable in the dfa C'. Transitions by symbol c will be used to reach all $\{q_i\} \cup S$ $(2 \leqslant i \leqslant m-k)$, while transitions by symbol d will be used to reach k subsets $\{p_j, 2\}$ $(1 \leqslant j \leqslant k)$. Finally, we will use β symbols $e_1, e_2, \ldots, e_\beta$ to reach β special sets $\{p_1\} \cup S_\ell$ $(1 \leqslant \ell \leqslant \beta)$. The last symbol f will be used to prove the inequivalence of these reachable subsets.

To this aim, let $S_1, S_2, \ldots, S_{2^{n-1}}$ be all subsets of $\{1, 2, \ldots, n\}$ containing 1, which are ordered in such a way that if $|S_i| < |S_j|$ then $i < j$. Thus, $S_1 = \{1\}$.

Let $\Sigma = \{a, b, c, d, f, e_1, e_2, \ldots e_\beta\}$.

Define an m-state dfa $A = (Q_A, \Sigma, \delta_A, q_1, \{q_1\})$, where $Q_A = \{q_1, q_2, \ldots, q_{m-k}\} \cup \{p_1, p_2, \ldots, p_k\}$, and the transitions are as follows:

by a and b, each state goes to state q_1;
by c; there is a circle $(q_1 q_2 \;..\; q_{m-k})$, and each states p_j goes to itself;
by d, there is a circle $(q_1 p_1 \;..\; p_k)$, and each other state goes to q_1;
by f, state p_k goes to state q_1, and each other state goes to p_1;
by e_ℓ, each state goes to p_1 for $\ell = 1, 2, \ldots, \beta$.

Fig. 2 shows the states and the transitions on a, b, c, d, f in the deterministic automaton A. Transitions on symbols e_ℓ, which go from each state to state p_1, are omitted in the figure.

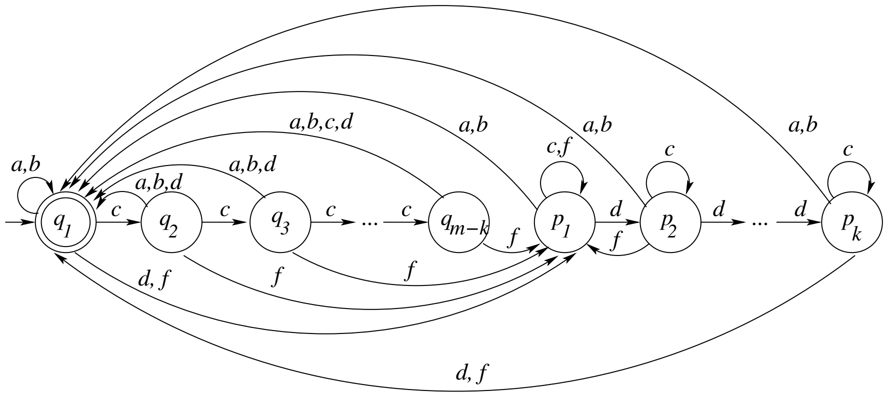


Fig. 2. The deterministic finite automaton A; transitions on a, b, c, d, f

Define an n-state dfa $B = (Q_B, \Sigma, \delta_B, 1, \{n\})$, where $Q_B = \{1, 2, \ldots, n\}$, and the transitions are defined as follows:

by a, there is a circle $(1 \, .. \, n)$;
by b, states 1 and n go to state 1, and each other state j goes to state $j + 1$;
by c, each state goes to itself;
by d and f, each state goes to state 2;
by e_ℓ, each state in the set S_ℓ goes to itself, and each other state goes to state 1 for all $\ell = 1, 2, \ldots, \beta$. The states and the transitions on a, b, c, d, f in the deterministic automaton B are shown in Fig. 3. Transitions by symbols $e_1, e_2, \ldots, e_\beta$, that depend on subsets $S_1, S_2, \ldots, S_\beta$, are again omitted.

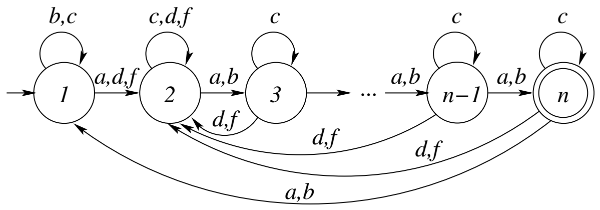


Fig. 3. The deterministic finite automaton B; transitions on a, b, c, d, f

Construct a nondeterministic automaton $C = (Q_A \cup Q_B, \Sigma, \delta, \{q_1, 1\}, \{n\})$ for the language $L(A)L(B)$ from the dfa's A and B by adding the transition from a state q in Q_A to state 1 on a symbol s whenever $\delta_A(q, s) = q_1$. Let C' be the corresponding subset automaton. Let $\mathcal{R}$ be the following system of subsets of the state set $Q_A \cup Q_B$:

$$\begin{aligned}\mathcal{R} = \{&\{q_i\} \cup S \mid 1 \leqslant i \leqslant m - k,\ S \subseteq \{1, 2, \ldots, n\} \text{ and } 1 \in S\} \cup \\ &\{\{p_j, 2\} \mid 1 \leqslant j \leqslant k\} \cup \\ &\{\{p_1\} \cup S_\ell \mid 1 \leqslant \ell \leqslant \beta\}.\end{aligned}$$

There are $(m - k) \cdot 2^{n-1} + k + \beta$ (that is, α) subsets in $\mathcal{R}$. Let us show that the system $\mathcal{R}$ is the system of all reachable and pairwise inequivalent states of the deterministic finite automaton C'.

We are going to show that (i) each set in $\mathcal{R}$ is a reachable state of the DFA C', (ii) no other subsets are reachable in the DFA C', (iii) the reachable states of the DFA C' are pairwise inequivalent. The lemma then follows immediately.

(i) Let us show that each subset in $\mathcal{R}$ is a reachable state in the DFA C'. The set $\{q_1, 1\}$ is the initial state of the DFA C', and so is reachable. Next, the set $\{q_1\} \cup \{1, r_2, r_3, \ldots, r_t\}$, where $2 \leqslant r_2 < r_3 < \cdots < r_t \leqslant n$, of size t, can be reached from the set $\{q_1\} \cup \{1, r_3 - r_2 + 1, \ldots, r_t - r_2 + 1\}$ of size $t - 1$ by the string $ab^{r_2 - 2}$ since the set $\{q_1\} \cup \{1, r_3 - r_2 + 1, \ldots, r_t - r_2 + 1\}$ goes the set $\{q_1, 1\} \cup \{2, r_3 - r_2 + 2, \ldots, r_t - r_2 + 2\}$ by a, and then to the set $\{q_1, 1\} \cup \{r_2, r_3, \ldots, r_t\}$ by $b^{r_2 - 2}$.

Thus, the reachability of sets $\{q_1\} \cup S$, where $S \subseteq \{1, 2, \ldots, n\}$ and $1 \in S$, follows by induction. For each i with $2 \leqslant i \leqslant m - k$, the set $\{q_i\} \cup S$ can be reached from the set $\{q_1\} \cup S$ by the string c^{i-1}. Next, the initial state $\{q_1, 1\}$ goes to state $\{p_1, 2\}$ by f, and then to state $\{p_j, 2\}$ by d^{j-1} for $j = 2, 3, \dots, k$. Finally, state $\{q_1\} \cup S_\ell$ goes to state $\{p_1\} \cup S_\ell$ by e_ℓ for $l = 1, 2, \dots, \beta$. Hence all sets in $\mathcal{R}$ are reachable in the DFA C'.

(ii) Let us prove that no other subset of $Q_A \cup Q_B$ is reachable in the DFA C'. Since the initial state of the DFA C' is in $\mathcal{R}$, it is sufficient to show that for each set R in $\mathcal{R}$ and each symbol s in Σ, the set $\delta(R, s)$ is in the system $\mathcal{R}$. Let $R = \{q\} \cup S$ be a set in $\mathcal{R}$, that is, either $R = \{q_i\} \cup S$ where $1 \leqslant i \leqslant m - k$ and S is a subset of Q_B containing state 1, or $R = \{p_j, 2\}$ where $1 \leqslant j \leqslant k$, or $R = \{p_1\} \cup S_\ell$ where $1 \leqslant \ell \leqslant \beta$. Then

- by a and b, the set R goes to a set $\{q_1\} \cup S'$, where S' is a subset of Q_B containing state 1. Each such subset is in $\mathcal{R}$;
- by c, R goes to $\{q_1\} \cup S$ if $q = q_{m-k}$, to $\{q_{i+1}\} \cup S$ if $q = q_i$, or to itself;
- by d, R goes either to $\{p_j, 2\}$ for some j with $1 \leqslant j \leqslant k$, or to the set $\{q_1, 1, 2\}$;
- by f, the set R goes either to the set $\{p_1, 2\}$ or to the set $\{q_1, 1, 2\}$;
- by e_ℓ, where $1 \leqslant \ell \leqslant \beta$, the set R goes to the set $\{p_1\} \cup S_\ell$ in the case of $S_\ell \subseteq S$, and to a set $\{p_1\} \cup S'$ such that $|S'| < |S_\ell|$, otherwise. In the latter case, since the sets $S_1, S_2, \dots, S_\beta$ are ordered according to their cardinality, it holds that $S' = S_t$ for some $t < \ell$, and so $\{p_1\} \cup S'$ is in $\mathcal{R}$. Thus each set in $\mathcal{R}$ goes to a set in $\mathcal{R}$ by each symbol, which means that $\mathcal{R}$ is the system of *all* reachable states in the DFA C'.

(iii) To prove inequivalence notice that the string fa^{n-1} is accepted by the NFA C starting in state p_k, but is not accepted by C starting in any other state. Next, the string $d^{k-j}fa^{n-1}$, where $1 \leqslant j \leqslant k-1$, is accepted by the NFA C only from state p_j, and the string $d^k fa^{n-1}$ is accepted by the NFA C only from state q_1. Finally, the string $c^{m-k-i+1}d^k fa^{n-1}$, where $2 \leqslant i \leqslant m-k$, is accepted by the NFA C only from state q_i. This means that two subsets $\{p\} \cup S$ and $\{q\} \cup T$ with $p \neq q$ are inequivalent. Consider two different reachable subsets $\{q\} \cup S$ and $\{q\} \cup T$. Then either $q \in \{q_1, q_2, \dots, q_{m-k}\}$ or $q = p_1$ (since in $\mathcal{R}$ there is only one subset containing p_j for $j = 2, 3, \dots, k$). In the former case, the subsets S and T must differ in a state r with $r \geqslant 2$, and the string a^{n-r} distinguishes the two states $\{q_i\} \cup S$ and $\{q_i\} \cup T$. In the latter case, the subsets S and T differ in a state r in $\{1, 2, \dots, n\}$ and the string a^{n-r} distinguishes $\{p_1\} \cup S$ and $\{p_1\} \cup T$. Thus the minimal DFA for the language $L(A)L(B)$ has $(m-k) \cdot 2^{n-1} + k + \beta$, that is α, states, and the lemma follows.

Remark. In the case of $k = m$, we should be little bit more careful. In this case, we do not use transitions on b, and the initial and the only accepting state will be p_1. The reachable and parwise inequivalent states in the dfa C' will be $\{p_1, 1\}, \{p_1, 1, 2\}, \dots, \{p_1, 1, 2, \dots, n\}$, $\{p_j, 2\}$ for $j = 1, 2, \dots, m$, and $\{p_1\} \cup S_\ell$ for $\ell = 1, 2, \dots, \beta - n$, where $S_1, S_2, \dots$ are the remaining subsets of $\{1, 2, \dots, n\}$ containing 1 ordered according to their cardinality. In this way we get all values from $m + n$ to $m + 2^{n-1}$. The values from m to $m + n - 1$ are covered by proposition 1. □

Our last lemma covers the remaining cases.

Lemma 5. *For all integers m, n and α such that $n \geqslant 2$ and $m \cdot 2^{n-1} < \alpha \leqslant (m-1) \cdot 2^n + 2^{n-1}$, there exist a minimal dfa A of m states and a minimal dfa B of n states, both defined over a growing alphabet, such that the minimal dfa for the language $L(A)L(B)$ has α states.*

Proof. If $m \cdot 2^{n-1} < \alpha \leqslant (m-1) \cdot 2^n + 2^{n-1}$, then there is an integer k with $2 \leqslant k \leqslant m$ such that $(m-k) \cdot 2^n + k \cdot 2^{n-1} < \alpha \leqslant (m-(k-1)) \cdot 2^n + (k-1) \cdot 2^{n-1}$. Then

$$\alpha = (m-k) \cdot 2^n + k \cdot 2^{n-1} + \beta,$$

where $2 \leqslant k \leqslant m$ and $1 \leqslant \beta \leqslant 2^{n-1}$.

We will proceed in a similar way as in the previous lemma. The dfa's A and B will have a similar structure as before. However, now we will define transitions on symbols in Σ in such a way, so that the following sets will be reachable and pairwise inequivalent in the dfa C' for the language $L(A)L(B)$:

- all subsets $\{q_i\} \cup S$, where $1 \leqslant i \leqslant m-k$ and S is a subset of $\{1, 2, \ldots, n\}$,
- all subsets $\{p_j\} \cup T$, where $1 \leqslant j \leqslant k$ and $T \subseteq \{1, 2, \ldots, n\}$ and $1 \in T$,
- some special subsets $\{p_k\} \cup S_\ell$ for $\ell = 1, 2, \ldots, \beta$.

To this aim, let $S_1, S_2, \ldots, S_{2^{n-1}}$ be all subsets of the set $\{2, 3, \ldots, n\}$ ordered in such a way, that if $|S_i| < |S_j|$, then $i < j$.

Let $\Sigma = \{a, b, c, f, e_1, e_2, \ldots, e_\beta\}$.

Define an m-state dfa $A = (Q_A, \Sigma, \delta_A, q_1, \{p_1\})$, where $Q_A = \{q_1, q_2, \ldots, q_{m-k}\} \cup \{p_1, p_2, \ldots, p_k\}$, the transitions on symbols a, b, c are shown in Fig. 4, by f, state q_{m-k} goes to state p_1, and each other state goes to state q_1; and by e_ℓ, state q_1 goes to state p_k, and each other state goes to state q_1 for all $\ell = 1, 2, \ldots, \beta$.

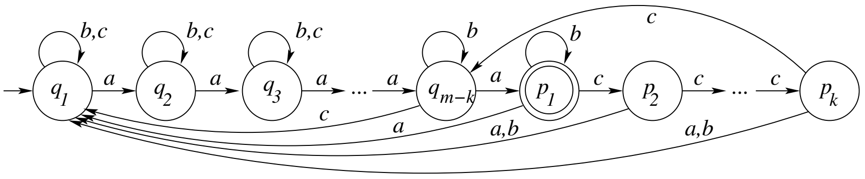


Fig. 4. The deterministic finite automaton A; transitions on a, b, c

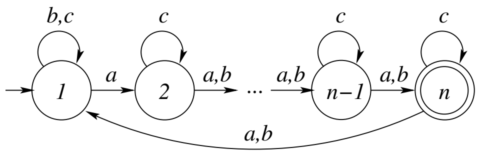


Fig. 5. The deterministic finite automaton B; transitions on a, b, c

Define an n-state dfa $B = (Q_B, \Sigma, \delta_B, 1, \{n\})$, where $Q_B = \{1, 2, \ldots, n\}$, transitions on symbols a, b, c are shown in Fig. 5, by f, each state goes to state 2, and by e_ℓ, each state in the set S_ℓ goes to itself, and each other state goes to $\min S_\ell$ for all $\ell = 1, 2, \ldots, \beta$.

Construct a nondeterministic finite automaton $C = (Q_A \cup Q_B, \Sigma, \delta, \{q_1\}, \{n\})$ from the dfa's A and B by adding the transition from a state q in Q_A to state 1 on a symbol s whenever $\delta_A(q, s) = q_1$. Let C' be the dfa obtained from the nfa C by the subset construction. Let $\mathcal{R}$ be the following system of subsets of the state set $Q_A \cup Q_B$:

$$\begin{aligned}\mathcal{R} = \{&\{q_i\} \cup S \mid 1 \leqslant i \leqslant m-k,\ S \subseteq \{1, 2, \ldots, n\}\} \cup \\ &\{\{p_j\} \cup T \mid 1 \leqslant j \leqslant k, T \subseteq \{1, 2, \ldots, n\} \text{ and } 1 \in T\}\ cup \\ &\{\{p_k\} \cup S_\ell \mid 1 \leqslant \ell \leqslant \beta\}.\end{aligned}$$

There are $(m-k) \cdot 2^n + k \cdot 2^{n-1} + \beta$ subsets in $\mathcal{R}$. Let us show that the system $\mathcal{R}$ consists of all reachable and pairwise inequivalent states of the dfa C'.

The reachability of subsets $\{q_i\} \cup S$ and $\{p_1\} \cup T$, where $1 \leqslant i \leqslant m-k$, $S \subseteq \{1, 2, \ldots, n\}$, and $T \subseteq \{1, 2, \ldots, n\}$ with $1 \in T$, can be shown in the same way as in [15, Theorem 1] since the transitions on a, b in states $q_1, q_2, \ldots, q_{m-k}, p_1$ and in states in Q_B are the same as in [15]. Each set $\{p_1\} \cup T$ goes to the set $\{p_j\} \cup T$ $(2 \leqslant j \leqslant k)$ by the string c^{j-1}. The set $\{q_1\} \cup S_\ell$ goes to the set $\{p_k\} \cup S_\ell$ for $\ell = 1, 2, \ldots, \beta$. Thus, all subsets in $\mathcal{R}$ are reachable in the DFA C'.

To prove that no other subsets are reachable, again, the initial state $\{q_1\}$ is in $\mathcal{R}$, and for each subset R in the system $\mathcal{R}$ and each symbols s in Σ, the set $\delta(R, s)$ is in the system $\mathcal{R}$ as well.

For the proof of the inequivalence of these states notice that the string fa^{n-1} is accepted by the NFA C only from state q_{m-k}, the string $c^{k-j+1} f a^{n-1}$, where $1 \leqslant j \leqslant k$, is accepted only from state p_j, and the string $a^{m-k-i} f a^{n-1}$, where $1 \leqslant i < m-k$, is accepted only from state q_i. The proof of inequivalence now proceeds in a similar way as in Lemma 4.

In the case of $k = m$, states p_j go to the initial and the only final state p_1 by a, by c, state p_k goes to p_1, by f state p_k goes to p_1 while the other states go to p_k, by e_ℓ, p_1 goes to p_k and the other states go to p_1. The reachable and inequivalent states in this case are $\{p_j\} \cup S$ where $1 \leqslant j \leqslant m$ and S is a subset of Q_B containing state 1, and the sets $\{p_k\} \cup S_\ell$ where $1 \leqslant \ell \leqslant \beta$. □

Hence we have shown the following theorem.

Theorem 2. *For all integers m, n and α such that either $n = 1$ and $1 \leqslant \alpha \leqslant m$, or $n \geqslant 2$ and $1 \leqslant \alpha \leqslant m \cdot 2^n - 2^{n-1}$, there exist a minimal deterministic finite automaton A of m states and a minimal deterministic finite automaton B of n states, both defined over a growing alphabet, such that the minimal deterministic finite automaton for the language $L(A)L(B)$ has exactly α states.* □

5 Conclusions

We have investigated the deterministic and nondeterministic state complexity of languages that can be obtained as the concatenation of two regular languages represented by deterministic and nondeterministic finite automata, respectively.

The nondeterministic state complexity of concatenation has been known to be $m + n$ and this bound is attained by the concatenation of binary languages $\{a^m\}^*$ and $\{b^n\}^*$ [10]. We have shown that all values from 1 to $m+n-1$ can be obtained as the size of a minimal nfa accepting the concatenation of an m-state and an n-state unary nfa languages. Whether or not the complexity of $m+n$ can be reached by the concatenation of unary languages remains open. We conjecture that the answer is negative.

In the deterministic case, we have used a growing alphabet to define, for each α with $1 \leqslant \alpha \leqslant m \cdot 2^n - 2^{n-1}$, an m-state and an n-state deterministic finite automata recognizing languages that need exactly α states for their concatenation. It remains open whether the whole range of complexities for concatenation of dfa languages can be obtained using a fixed alphabet.

Acknowledgements

I would like to thank Professor Geffert for the interesting discussion on concatenation of unary nfa languages. I am also grateful to anonymous referees of CSR 2009 for their corrections and suggestions.

References

1. Aho, A.V., Ullman, J.D., Yannakakis, M.: On notions of information transfer in VLSI circuits. In: Proc. 15th STOC, pp. 133–139 (1983)
2. Birget, J.-C.: Intersection and union of regular languages and state complexity. Inform. Process. Lett. 43, 185–190 (1992)
3. Birget, J.-C.: Partial orders on words, minimal elements of regular languages, and state complexity. Theoret. Comput. Sci. 119, 267–291 (1993)
4. Câmpeanu, C., Culik II, K., Salomaa, K., Yu, S.: State complexity of basic operations on finite languages. In: Boldt, O., Jürgensen, H. (eds.) WIA 1999. LNCS, vol. 2214, pp. 60–70. Springer, Heidelberg (2001)
5. Domaratzki, M.: State complexity and proportional removals. J. Autom. Lang. Comb. 7, 455–468 (2002)
6. Dassow, J., Stiebe, R.: Nonterminal complexity of some operations on context-free languages. Fundam. Inform. 83, 35–49 (2008)
7. Geffert, V. (Non)determinism and the size of one-way finite automata. In: Mereghetti, C., Palano, B., Pighizzini, G., Wotschke, D. (eds.) 7th International Workshop on Descriptional Complexity of Formal Systems, pp. 23–37. University of Milano, Italy (2005)
8. Geffert, V.: Magic numbers in the state hierarchy of finite automata. Inform. Comput. 205, 1652–1670 (2007)
9. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inform. Process. Lett. 59, 75–77 (1996)

10. Holzer, M., Kutrib, M.: Nondeterministic descriptional complexity of regular languages. Internat. J. Found. Comput. Sci. 14, 1087–1102 (2003)
11. Hricko, M., Jirásková, G., Szabari, A.: Union and intersection of regular languages and descriptional complexity. In: Mereghetti, C., Palano, B., Pighizzini, G., Wotschke, D. (eds.) 7th International Workshop on Descriptional Complexity of Formal Systems, pp. 170–181. University of Milano, Italy (2005)
12. Hromkovič, J.: Communication Complexity and Parallel Computing. Springer, Heidelberg (1997)
13. Iwama, K., Kambayashi, Y., Takaki, K.: Tight bounds on the number of states of DFAs that are equivalent to n-state NFAs. Theoret. Comput. Sci. 237, 485–494 (2000)
14. Iwama, K., Matsuura, A., Paterson, M.: A family of NFAs which need $2^n - \alpha$ deterministic states. Theoret. Comput. Sci. 301, 451–462 (2003)
15. Jirásek, J., Jirásková, G., Szabari, A.: State complexity of concatenation and complementation. Internat. J. Found. Comput. Sci. 16, 511–529 (2005)
16. Jirásek, J., Jirásková, G., Szabari, A.: Deterministic blow-ups of minimal nondeterministic finite automata over a fixed alphabet. Internat. J. Found. Comput. Sci. 19, 617–631 (2005)
17. Jirásková, G.: Note on minimal finite automata. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, pp. 421–431. Springer, Heidelberg (2001)
18. Jirásková, G.: State complexity of some operations on binary regular languages. Theoret. Comput. Sci. 330, 287–298 (2005)
19. Jirásková, G.: On the State Complexity of Complements, Stars, and Reversals of Regular Languages. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 443–454. Springer, Heidelberg (2008)
20. Maslov, A.N.: Estimates of the number of states of finite automata. Soviet Math. Dokl. 11, 1373–1375 (1970)
21. Leiss, E.: Succinct representation of regular languages by Boolean automata. Theoret. Comput. Sci. 13, 323–330 (1981)
22. Pighizzini, G., Shallit, J.: Unary language operations, state complexity and Jacobsthal's function.Internat. J. Found. Comput. Sci. 13, 145–159 (2002)
23. Rabin, M., Scott, D.: Finite automata and their decision problems. IBM Res. Develop. 3, 114–129 (1959)
24. Salomaa, A., Wood, D., Yu, S.: On the state complexity of reversals of regular languages. Theoret. Comput. Sci. 320, 315–329 (2004)
25. Sipser, M.: Introduction to the theory of computation. PWS Publishing Company, Boston (1997)
26. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, ch. 2, vol. I, pp. 41–110. Springer, Heidelberg (1997)
27. Yu, S.: State complexity: Recent results and open problems. Fundam. Inform. 64, 471–480 (2005)
28. Yu, S., Zhuang, Q., Salomaa, K.: The state complexity of some basic operations on regular languages. Theoret. Comput. Sci. 125, 315–328 (1994)
29. Zijl, L.: Magic numbers for symmetric difference NFAs. Internat. J. Found. Comput. Sci. 16, 1027–1038 (2005)

Approximability of the Maximum Solution Problem for Certain Families of Algebras

Peter Jonsson[⋆] and Johan Thapper

Department of Computer and Information Science
Linköpings universitet
SE-581 83 Linköping, Sweden
{petej,johth}@ida.liu.se

Abstract. We study the approximability of the *maximum solution problem*. This problem is an optimisation variant of the constraint satisfaction problem and it captures a wide range of interesting problems in, for example, integer programming, equation solving, and graph theory. The approximability of this problem has previously been studied in the two-element case [Khanna et al, 'The approximability of constraint satisfaction', SIAM Journal on Computing 23(6), 2000] and in some algebraically motivated cases [Jonsson et al, 'MAX ONES generalized to larger domains', SIAM Journal on Computing 38(1), 2008]. We continue this line of research by considering the approximability of MAX SOL for different types of constraints. Our investigation combined with the older results strengthens the hypothesis that MAX SOL exhibits a pentachotomy with respect to approximability.

Keywords: Optimisation, approximability, constraint satisfaction, algebra, computational complexity.

1 Introduction

We study the *maximum solution problem*, a problem perhaps most intuitively described as a generalisation of MAX ONES. The latter problem is that of finding an assignment from a set of variables to a domain $\{0, 1\}$ such that a given set of constraints are satisfied and the number of variables assigned to 1 are maximised. This type of constraint satisfaction problems are commonly parametrised by a *constraint language* Γ, i.e., a set of relations describing the structure of the constraints that are allowed to appear. Among the problems realisable by MAX ONES(Γ) one finds the MAX INDEPENDENT SET-problem for graphs and certain variants of MAX 0/1 PROGRAMMING. The maximum solution problem, or MAX SOL(Γ) for short, generalises the domain of the variable assignment from $\{0, 1\}$ to an arbitrary finite subset of the natural numbers. The measure of a solution is now the sum of a variable weight times its assigned value, taken over all variables. This allows us to capture a wider array of problems, including certain problems in integer linear programming, problems in multi-valued logic [8], and in equation solving over various algebraic structures [11]. The problem has also been studied (with respect to computational complexity) on undirected graphs [9], i.e., when Γ consists

[⋆] 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.

of a single, symmetric binary relation. However, it is quite obvious that the systematic study of MAX SOL is still in its infancy; for example, given an arbitrary constraint language Γ, no plausible conjecture has been suggested for the approximability (or even the complexity) of MAX SOL(Γ).

The situation is quite different for MAX ONES: for any constraint language Γ, MAX ONES(Γ) is either polynomial-time solvable, **APX**-complete, **poly-APX**-complete, it is **NP**-hard to obtain a solution of nonzero measure, or it is **NP**-hard to obtain any solution. This classification and the borderlines between the different cases were presented by Khanna et al. in [10]. For MAX SOL(Γ), a similar classification of approximability for homogeneous constraint languages Γ was obtained in [7], together with a (conjectured complete) classification of the approximability for maximal constraint languages. Such classifications are obviously interesting from a theoretical point of view, but also from a practical point of view, where they can help identifying families of tractable constraints and algorithms for them.

A constraint language can be extended in a complexity-preserving way to a larger set of relations called a *relational clone*. Furthermore, the relational clones can be described by algebraic operations so it makes sense talking about MAX SOL$(\mathbf{A})$ where $\mathbf{A}$ is an algebra. In this way, we can consider complexity-theoretic problems from an algebraic angle and this algebraic approach [3,6] has proved to be very fruitful when studying constraint problems. The study of MAX SOL using the algebraic approach was initiated in [7] and we continue the algebraic study of MAX SOL in this paper. We begin by providing approximability results for certain affine algebras and 2-element algebras; these results are used extensively in the 'main' classification results. These results appear to be useful for studying other algebras as well. For instance, a classification of paraprimal algebras probably hinges on a classification of all affine algebras (via Corollary 4.12 in [14]). Our proof is based on combinatorial properties in the subspace lattices of finite vector spaces.

The first classification determines the approximability of MAX SOL$(\mathbf{A})$ when $\mathbf{A}$ is a strictly simple surjective algebra. Such an algebra has a very 'simple' structure: all its smaller homomorphic images and all its proper subalgebras are one-element. These algebras can be viewed as building blocks for more complex algebras and they are well-studied in the literature; an understanding of such algebras is probably needed in order to make further progress using the algebraic approach. We note, for example, that the proof of our second classification result is partly based on the results for strictly simple surjective algebras. Concrete examples include when $\mathbf{A}$ is a finite field of prime order or a Post algebra. Furthermore, these algebras generalise the two-element case nicely since every surjective two-element algebra is strictly simple. Our proof is based on Szendrei's characterisation of strictly simple surjective algebras. In each case of the characterisation, we can either use results from [7] or our new results for affine and 2-element algebras in order to determine the approximability. The corresponding classification of the CSP problem was carried out in [3].

The second classification considers algebras that are symmetric in the sense of [15]. Examples include algebras whose automorphism group contains the alternating group (i.e. the permutation group containing only even permutations) and certain three-element algebras with cyclic automorphism groups [15]. Well-known examples are the *homogeneous* algebras; an algebra $\mathbf{A}$ is homogeneous if its automorphism group Aut($\mathbf{A}$) is the full symmetric group. The approximability of MAX SOL($\mathbf{A}$) is known for all homogeneous algebras [7] and our result generalises this result. The proof is basically a mix of Szendrei's classification for symmetric algebras [15] and our approximability results for affine and strictly simple surjective algebras. It should be noted that our proof is considerably simpler than the original proof for homogeneous algebras (which is a fairly tedious case analysis). As a by-product of the proof, we also get a classification of CSP($\mathbf{A}$) for symmetric algebras (Theorem 12.)

In order to concretise, consider the equation $x = y + 1 \pmod 3$ over the domain $A = \{0, 1, 2\}$. For brevity, we define $R = \{(x, y) \mid x = y + 1 \pmod 3\}$ and note that $R = \{(x, \sigma(x)) \mid x \in A\}$ for the permutation $\sigma(0) = 2$, $\sigma(1) = 0$, and $\sigma(2) = 1$. Let Γ be *any* relational clone containing R. It is known that for every permutation $\pi : A \to A$, $\{(x, \pi(x)) \mid x \in A\} \in \Gamma$ if and only if $\pi(x) \in$ Aut($\mathsf{Pol}(\Gamma)$) where $\mathsf{Pol}(\Gamma)$ denotes the algebra with universe A and the functions that preserve Γ. It is now easy to see that Aut($\mathsf{Pol}(\Gamma)$) contains every even permutation on A: the identity is always an automorphism and σ^{-1} is generated by σ. Thus, $\mathsf{Pol}(\Gamma)$ is symmetric and the approximability of MAX SOL(Γ) can be determined using Theorem 11.

2 Preliminaries

This section is divided into two parts: we begin by giving the formal definition of the constraint satisfaction and the maximum solution problems, and continue by reviewing algebraic techniques for analysing relations. We will assume basic familiarity with complexity and approximability classes (such as **PO**, **NPO**, **APX** and **poly-APX**), and reductions (such as AP-, S-, and L-reductions) [1,10].

We formally define constraint satisfaction as follows: let A (*the domain*) be a finite set and let R_A denote the set of all finitary relations over A. A constraint language A is a subset $\Gamma \subseteq R_A$. The constraint satisfaction problem over the constraint language Γ, denoted CSP(Γ), is the decision problem with instance (V, A, C). Here, V is a set of variables, A is a finite set of values, and C is a set of constraints $\{C_1, \ldots, C_q\}$, in which each constraint C_i is a pair (s_i, ϱ_i) with s_i a list of variables of length m_i, called the constraint scope, and ϱ_i an m_i-ary relation over the set A, belonging to Γ, called the constraint relation. The question is whether or not there exists a function from V to A such that, for each constraint in C, the image of the constraint scope is a member of the constraint relation.

We define the maximum solution problem over a constraint language Γ (MAX SOL(Γ)) as the maximisation problem with

Instance. A tuple (V, A, C, w), where A is a finite subset of $\mathbb{N}$, (V, A, C) is a CSP instance over Γ, and $w : V \to \mathbb{Q}^+$ is a weight function.
Solution. An assignment $f : V \to A$ such that all constraints are satisfied.
Measure. $\sum_{v \in V} w(v) \cdot f(v)$

Next, we consider clones and operations. As usual, let A be a domain. An operation on A is an arbitrary function $f : A^k \to A$ and the set of all finitary operations on A is denoted by O_A. A k-ary operation $f \in O_A$ can be extended to an operation on n-tuples $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}$ by $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) =$

$$(f(\boldsymbol{t_1}[1], \boldsymbol{t_2}[1], \ldots, \boldsymbol{t_k}[1]), f(\boldsymbol{t_1}[2], \boldsymbol{t_2}[2], \ldots, \boldsymbol{t_k}[2]), \ldots f(\boldsymbol{t_1}[n], \boldsymbol{t_2}[n], \ldots, \boldsymbol{t_k}[n])),$$

where $\boldsymbol{t_j}[i]$ is the i-th component of $\boldsymbol{t_j}$. Let $\varrho \in R_A$. If f is an operation such that for all $\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k} \in \varrho_i$ $f(\boldsymbol{t_1}, \boldsymbol{t_2}, \ldots, \boldsymbol{t_k}) \in \varrho_i$, then ϱ is *preserved* by f. If all constraint relations in Γ are preserved by f, then Γ is preserved by f. An operation f which preserves Γ is called a polymorphism of Γ and the set of polymorphisms is denoted $\mathsf{Pol}(\Gamma)$. Given a set of operations F, the set of all relations that are preserved by the operations in F is denoted $\mathsf{Inv}(F)$.

Sets of operations of the form $\mathsf{Pol}(\Gamma)$ are known as *clones*, and they are well-studied objects in algebra (cf. [12,14]). We remark that the operators Inv and Pol form a Galois correspondence between the set of relations over A and the set of operations on A. A comprehensive study of this correspondence can be found in [12].

A first-order formula φ over a constraint language Γ is said to be *primitive positive* (we say φ is a pp-formula for short) if it is of the form $\exists \mathbf{x} : (\varrho_1(\mathbf{x}_1) \land \ldots \land \varrho_k(\mathbf{x}_k))$ where $\varrho_1, \ldots, \varrho_k \in \Gamma$ and $\mathbf{x}_1, \ldots, \mathbf{x}_k$ are vectors of variables of size equal to the arity of the corresponding relation. Note that a pp-formula φ with m free variables defines an m-ary relation $\varrho \subseteq A^m$; ϱ is the set of all tuples satisfying the formula φ.

We define a closure operation $\langle \cdot \rangle$ such that $\varrho \in \langle \Gamma \rangle$ if and only if the relation ϱ can be obtained from Γ by pp-formulas. Sets of relations of the form $\langle \Gamma \rangle$ are called *relational clones*. The following theorem states that we have access to a handy S-reduction from MAX SOL over finite subsets of $\langle \Gamma \rangle$ to MAX SOL over Γ itself.

Theorem 1 ([7]). *Let Γ be a constraint language and $\Gamma' \subseteq \langle \Gamma \rangle$ a finite subset. Then,* MAX SOL(Γ') *is S-reducible to* MAX SOL(Γ).

The concept of a core of a constraint language Γ has previously shown its value when classifying the complexity of CSP(Γ). The analogous concept of a *max-core* for the optimization problem MAX SOL(Γ) was defined in [8]: a constraint language Γ is a max-core if and only if there is no noninjective unary operation f in $\mathsf{Pol}(\Gamma)$ such that $f(a) \geq a$ for all $a \in A$. A constraint language Γ' is a max-core of Γ if and only if Γ' is a max-core and $\Gamma' = f(\Gamma)$ for some unary operation $f \in \mathsf{Pol}(\Gamma)$ such that $f(a) \geq a$ for all $a \in A$. We have the following result:

Lemma 2 ([8]). *If Γ' is a max-core of Γ, then* MAX SOL(Γ) *and* MAX SOL(Γ') *are equivalent under S-reductions.*

We will now introduce the concept of an algebra and some of the terminology related to it. For a coherent treatment of this subject, we refer to [14]. Let A be a domain. An *algebra* $\mathbf{A}$ over A is a tuple $(A; F)$, where $F \subseteq O_A$ is a family of operations on A. For the purposes of this paper, all algebras will be finite, i.e., the set A will be finite. An operation $f \in O_A$ is called a *term operation* of $\mathbf{A}$ if $f \in \mathsf{Pol}(\mathsf{Inv}(F))$. The set of all term operations of $\mathbf{A}$ will be denoted by $\mathsf{Term}(\mathbf{A})$. Two algebras over the same universe are called *term equivalent* if they have the same set of term operations. An operation f is called *idempotent* if $f(a, \ldots, a) = a$ for all $a \in A$. The set of all idempotent term operations of $\mathbf{A}$ will be denoted by $\mathsf{Term}_{id}(\mathbf{A})$. For a domain A, let $\mathcal{C}_A$ denote the constraint language consisting of all constant, unary constraints, i.e., $\mathcal{C}_A = \{\{(a)\} \mid a \in A\}$. We use algebras to specify constraint languages and we will often write $\textsc{Max Sol}(\mathbf{A})$ for the problem $\textsc{Max Sol}(\mathsf{Inv}(\mathsf{Term}(\mathbf{A})))$.

Next, we present some operations that will be important in the sequel. If $\overline{A}$ is an abelian group, then the *affine operation* $a_{\overline{A}}(a,b,c) : A^3 \to A$ satisfies $a_{\overline{A}}(a,b,c) = a - b + c$. The *discriminator operation* $t : A^3 \to A$ satisfies $t(a,b,c) = c$ if $a = b$ and $t(a,b,c) = a$ otherwise. The *dual discriminator operation* $d : A^3 \to A$ satisfies $d(a,b,c) = a$ if $a = b$ and $d(a,b,c) = c$ otherwise. Finally, the *switching operation* $s : A^3 \to A$ satisfies $s(a,b,c) = c$ if $a = b$, $s(a,b,c) = b$ if $a = c$, and $s(a,b,c) = a$ otherwise. We remind the reader that $\textsc{Csp}(\mathsf{Inv}(r)) \in \mathbf{P}$ when $r \in \{a_{\overline{A}}, t, d, s\}$, cf. [3,6]. The following proposition is a summary of the results from [7] which we will need:

Proposition 3. *(1)* $\textsc{Max Sol}(\mathsf{Inv}(t))$ *is in* **PO**. *(2) Let* $R = \{(a,a),(a,b),(b,a)\}$ *with* $a, b \in A$ *and* $0 < a < b$. *Then,* $\textsc{Max Sol}(\{R\})$ *is* **APX**-*complete. (3) Let* $R = \{(0,0),(0,b),(b,0)\}$ *with* $b \in A$ *and* $0 < b$. *Then,* $\textsc{Max Sol}(\{R\})$ *is* **poly-APX**-*complete.*

3 Affine Algebras and Two-Element Algebras

In this section, we look at certain affine algebras and constraint languages over 2-element domains. Let $\overline{A} = (A; +)$ be a finite abelian group. The finite-dimensional vector space on $\overline{A}$ over K will be denoted $_K\overline{A} = (A; +, K)$. An algebra $\mathbf{A}$ is said to be *affine with respect to an abelian group* $\overline{A}$ if (1) $\mathbf{A}$ and $\overline{A}$ have the same universe, (2) the 4-ary relation $Q_{\overline{A}} = \{(a,b,c,d) \in A^4 \mid a - b + c = d\}$ is in $\mathsf{Inv}(\mathbf{A})$, and (3) $a_{\overline{A}}$ is a term operation of $\mathbf{A}$. It is known that $\textsc{Max Sol}(\mathbf{A})$ is in **APX** for all affine algebras $\mathbf{A}$ and that $\textsc{Max Sol}(\mathbf{A})$ is **APX**-complete for the affine algebra $\mathbf{A} = (A; a_{\overline{A}})$ [7]. Here, we will extend the latter result to cover some affine algebras with a larger set of term operations, where the underlying group is a finite vector space.

Let $_K\overline{A}$ be an n-dimensional vector space over a finite field K of size q. Let $\Lambda_0(_K\overline{A})$ be the constraint language consisting of all relations $\{(x_1, \ldots, x_n) \mid \sum_{i=1}^{n} c_i x_i = d\}$, for some $c_i \in K, d \in A$ and with $\sum_{i=1}^{n} c_i = 0$. The algebra over A with operations $\mathsf{Pol}(\Lambda_0(_K\overline{A}))$ is affine and we have the following result:

Theorem 4. $\textsc{Max Sol}(\Lambda_0(_K\overline{A}))$ *is* **APX**-*hard for any finite dimensional vector space* $_K\overline{A}$ *over a finite field* K *of size* $q \geq 2$.

The proof of this theorem relies on Lemma 5 which will be presented below. The lemma uses the subspace structure in finite vector spaces and we will need some terminology and notation: an *affine hyperplane* in $_K\overline{A}$ is a coset $a + S$, where $a \in A$ and S is a codimension 1 subspace of $_K\overline{A}$. Let $\mathcal{H}$ be the set of affine hyperplanes in $_K\overline{A}$. Denote by V the q^n-dimensional vector space over $\mathbb{Q}$ with basis A. For any subset $B \subseteq A$, let $\chi(B)$ denote the characteristic vector of B, i.e., $\chi(B) = \sum_{a \in B} a$. Let $g : A \to \mathbb{Q}$ be any function from A to the rational numbers. We can then extend g to a linear transformation $g : V \to \mathbb{Q}$ by letting $g(v) = \sum_i v_i g(a_i)$, when $v = \sum_i v_i a_i$. In particular, $g(\chi(B)) = \sum_{a \in B} g(a)$.

Lemma 5. *If $g(\chi(H)) = C$ for all $H \in \mathcal{H}$ and some constant C, then $g(a) = C/q^{n-1}$ for all $a \in A$.*

Proof. We will show that the set $X = \{\chi(H) \mid H \in \mathcal{H}\}$ spans V. From this it follows that g is uniquely determined by its values on X. The *q-binomial coefficients* are defined by

$$b_q(n,k) = \frac{(q^n-1)(q^{n-1}-1)\cdots(q^{n-k+1}-1)}{(q^k-1)(q^{k-1}-1)\cdots(q-1)}.$$

They count, among other things, the number of k-dimensional subspaces in an n-dimensional vector space over a finite field of size q. The number of codimension 1 subspaces (hyperplanes) containing a fixed 1-dimensional subspace (line) is counted by $b_q(n-1,1)$ (cf. [4].) We let $a \in A$ be fixed and for each $v \in A$, $v \neq a$, we count the number of (affine) hyperplanes through a that also contain v. They are exactly the hyperplanes containing the unique line through a and v and this number is $b_q(n-1,1) = \frac{q^{n-1}-1}{q-1}$. Thus, $\sum_{a \in H \in \mathcal{H}} \chi(H) = b_q(n,n-1) \cdot a + b_q(n-1,1) \cdot \chi(A-a) = q^{n-1} \cdot a + b_q(n-1,1) \cdot \chi(A)$. Now, choose an arbitrary affine hyperplane $H^1 \in \mathcal{H}$, let $H^2, \ldots, H^q$ denote its translations and note that $\sum_{i=1}^q \chi(H^i) = \chi(A)$. This implies that a can be written as the following linear combination of vectors in X: $a = q^{1-n}\left(\sum_{a \in H \in \mathcal{H}} \chi(H) - b_q(n-1,1) \cdot \sum_{i=1}^q \chi(H^i)\right)$. Apply g to both sides and use linearity to obtain $g(a) = q^{1-n}\left(b_q(n,n-1) \cdot C - b_q(n-1,1) \cdot q \cdot C\right) = q^{1-n}\left(\frac{q^n-1}{q-1} - \frac{q^n-q}{q-1}\right) \cdot C$, from which the lemma follows. □

The proof of Theorem 4 is a reduction from the problem MAX q-CUT. In this problem, one is given a graph $G = (V,E)$, and a solution σ to G is an assignment from V to some set K of size q. The objective is to maximise the number of edges $(u,v) \in E$ such that $\sigma(u) \neq \sigma(v)$. It is well known that MAX q-CUT is **APX**-complete for $q \geq 2$.

Proof (of Theorem 4). Let ${}_K\overline{A} = (A; +, K)$ be a vector space of size $|A| = q^n$. We present an L-reduction from MAX q-CUT to MAX SOL$(\Lambda_0({}_K\overline{A}))$, which proves that the latter is **APX**-hard.

Remember that we view the set of elements, A, as a subset of the natural numbers. In order to avoid ambiguity, we will use $a+b$ and $a-b$ to denote addition and subtraction in the group and $a \oplus b$ for the addition of the values of the group elements $a, b \in A$. Define $g : A \to \mathbb{N}$ as follows:

$$g(a) = \max_{x,y \in A}\{x \oplus y \mid x - y = a\} = \max_{x \in A}\{x \oplus (x-a)\}.$$

Extend g to arbitrary subsets $B \subseteq A$ through $g(B) = \sum_{a \in B} g(a)$. Let 0_A denote the zero vector and note that $g(0_A) = 2 \cdot \max A > g(a)$ for all $a \in A \setminus \{0_A\}$. Hence, g is nonconstant on A, and by Lemma 5, g must be nonconstant on the set of affine hyperplanes in ${}_K\overline{A}$. Therefore, we can let $H \subseteq A$ be a hyperplane (through the origin) in ${}_K\overline{A}$ and $e \in A$ an element such that $g(e+H) = \min\{g(a+H) \mid a \in A\} < g(A)/q$.

Let $I = (V,E)$ be an instance of MAX q-CUT. We create an instance $F(I)$ of MAX SOL$(\Lambda_0({}_K\overline{A}))$ as follows. For each $v_i \in V$, we create $|A|/q$ variables x_i^s, $s \in H$ and add the $(|A|/q)-1$ equations $x_i^s - x_i^{0_A} = s$ to $F(I)$, for $s \in H \setminus \{0_A\}$. These equations ensure that in a solution σ' to $F(I)$ it must hold that $\{\sigma'(x_i^s) \mid s \in H\} = a + H$ for some $a \in A$. For each edge $(v_i, v_j) \in E$, we create $2|A|^2/q$ new variables $y_{ij}^{st,k}$ and $z_{ij}^{st,k}$ for $s, t \in H$, $k \in K$, and add the following $|A|^2/q$ equations:

$$e + k(x_i^s - x_j^t) = y_{ij}^{st,k} - z_{ij}^{st,k}. \tag{1}$$

Finally, we let $w(x_i^s) = 0$ and $w(y_{ij}^{(s,t)}) = w(z_{ij}^{(s,t)}) = 1$. From a solution σ' to $F(I)$, we derive a solution σ to I by fixing an element $h_\perp \in A \setminus H$ and letting $\sigma(v_i) = k \in K$ when $\{\sigma'(x_i^s) \mid s \in H\} = kh_\perp + H$. Note that the measure of σ is independent of the choice of $h_\perp$.

We will now determine the measure of σ'. Note that in any solution, due to (1) and the definition of g, we have

$$\sigma'(y_{ij}^{st,k}) \oplus \sigma'(z_{ij}^{st,k}) \leq g(e + k(\sigma'(x_i^s) - \sigma'(x_j^t))). \tag{2}$$

Assume that $\{\sigma'(x_i^s) \mid s \in H\} = a + H$ and $\{\sigma'(x_j^t) \mid t \in H\} = b + H$. Then,

$$\{e + k(\sigma'(x_i^s) - \sigma'(x_j^t)) \mid s \in H\} = e + k(a - b) + H,$$

for any fixed $t \in H$ and $k \in K$. Therefore,

$$\sum_{s,t \in H, k \in K} \sigma'(y_{ij}^{st,k}) \oplus \sigma'(z_{ij}^{st,k}) \leq \sum_{s,t \in H, k \in K} g(e + k(\sigma'(x_i^s) - \sigma'(x_j^t))) = q^{n-1} \sum_{k \in K} g(e + k(a - b) + H)$$

If $\sigma(v_i) = \sigma(v_j)$, then $a - b \in H$, so the right-hand side equals $C = q^n g(e + H)$. Otherwise, $a - b \notin H$ and the right-hand side equals $D = q^{n-1} g(A)$. Now, assume that the q-cut determined by σ contains $m(\sigma)$ edges. Then, the measure m' of the solution σ' to $F(I)$ is bounded by

$$m'(\sigma') \leq |E| \cdot C + m(\sigma) \cdot (D - C). \tag{3}$$

When σ' is an optimal solution, the inequality in (2) can be replaced by an equality and it follows that

$$\textsc{opt}(F(I)) = |E| \cdot C + \textsc{opt}(I) \cdot (D - C). \tag{4}$$

By a straightforward probabilistic argument, it follows that $\textsc{opt}(I) \geq |E| \cdot (1 - 1/q)$, which in turn implies that

$$\textsc{opt}(F(I)) = \textsc{opt}(I) \left(\frac{|E| \cdot C}{\textsc{opt}(I)} + (D - C) \right) \leq \textsc{opt}(I) \cdot (C/(q-1) + D). \tag{5}$$

Note that both C and D are independent of the instance I. By subtracting (3) from (4) we get

$$\textsc{opt}(F(I)) - m'(\sigma') \geq (\textsc{opt}(I) - m(\sigma)) \cdot (D - C). \tag{6}$$

By the choice of H and e, we have $C = q^n g(e + H) < q^n g(A)/q = q^{n-1} g(A) = D$. Consequently, (5) and (6) shows that F is the desired L-reduction. □

The following consequence of Theorem 4 will be needed in the forthcoming proofs. The endomorphism ring of ${}_K\overline{A}$, i.e., the ring of linear transformations on ${}_K\overline{A}$, will be denoted $\mathsf{End}\ {}_K\overline{A}$. One can consider $\overline{A}$ as a module over $\mathsf{End}\ {}_K\overline{A}$ and this module will be denoted ${}_{(\mathsf{End}\ {}_K\overline{A})}\overline{A}$. The group of translations $\{x + a \mid a \in A\}$ will be denoted $T(\overline{A})$.

Corollary 6. *Let ${}_K\overline{A}$ be a finite dimensional vector space over a finite field K. Then,* MAX SOL($\mathsf{Inv}(\mathsf{Term}_{id}({}_K\overline{A}))$) *and* MAX SOL($\mathsf{Inv}(\mathsf{Term}_{id}({}_{(\mathsf{End}\ {}_K\overline{A})}\overline{A}) \cup T(\overline{A}))$) *are* **APX***-complete.*

We now use Theorem 4 to extend the classification of MAX ONES(Γ) by Khanna et al. [10]; this result will be needed several times in the sequel. Khanna et al. have proved a complete classification result for $D = \{0, 1\}$ and their proof is easy to generalise to the case when $D = \{0, a\}$, $a > 0$. If $D = \{a, b\}$, $0 < a < b$, then it is possible to exploit *Post's lattice* [13] for proving a similar result. The next lemma follows without difficulties by combining this lattice with results from Proposition 3 and Theorem 4.

Lemma 7. *Let Γ be a constraint language over a 2-element domain. Then,* MAX SOL(Γ) *is either in* **PO**, **APX***-complete,* **poly-APX***-complete, it is* **NP***-hard to find a nonzero solution or it is* **NP***-hard to find a feasible solution.*

4 Strictly Simple Surjective Algebras

The strictly simple surjective algebras were classified by Szendrei in [16], and the complexity of constraint satisfaction over such algebras was studied in [3]. Here, we do the corresponding classification of the approximability of MAX SOL. First, we will need a few definitions to be able to state Szendrei's theorem.

Let $\mathbf{A} = (A; F)$ be an algebra and let $B \subseteq A$. Let $f|_B$ denote the restriction of f to B and let $F|_B = \{f|_B \mid f \in F\}$. If for every $f \in F$, it holds that $f|_B(B) \subseteq B$, then $\mathbf{B} = (B; F|_B)$ is called a *subalgebra* of $\mathbf{A}$ and B is said to *support* this subalgebra. If $|B| < |A|$, then $\mathbf{B}$ is called a *proper subalgebra* of $\mathbf{A}$.

Let I be an index set and let $\mathbf{A} = (A; F)$ and $\mathbf{B} = (B; F')$ be two algebras with $F = \{f_i : i \in I\}$ and $F' = \{f'_i : i \in I\}$ such that f_i and f'_i have the same arity k_i for all $i \in I$. Then, a map $h : A \to B$ is called a *homomorphism* if for all $i \in I$, $h(f_i(a_1, \ldots, a_{k_i})) = f'_i(h(a_1), \ldots, h(a_{k_i}))$. When h is surjective, $\mathbf{B}$ is called a *homomorphic image* of $\mathbf{A}$. An algebra is called *simple* if all its smaller homomorphic images are trivial (one-element) and *strictly simple* if, in addition, all its proper subalgebras are one-element. An algebra is called *surjective* if all of its term operations are surjective.

Let $\mathcal{I}$ be a family of bijections between subsets of a set A. By $\mathcal{R}(\mathcal{I})$ we denote the set of operations on A which preserve each relation of the form $\{(a, \pi(a)) \mid a \in A\}$ for $\pi \in \mathcal{I}$. By $\mathcal{R}_{id}(\mathcal{I})$ we denote the set of idempotent operations in $\mathcal{R}(\mathcal{I})$.

Let G be a permutation group on A. Then, G is called *transitive* if, for any $a, b \in A$, there exists $g \in G$ such that $g(a) = b$. G is called *regular* if it is transitive and each nonidentity member has no fixed point. G is called *primitive*, or is said to act primitively on A, if it is transitive and the algebra $(A; G)$ is simple.

Let a be some fixed element in A, and define the relation $X_k^a = \{(a_1, \ldots, a_k) \in A^k \mid a_i = a$ for at least one i, $1 \leq i \leq k\}$. Let $\mathcal{F}_k^a$ denote the set of all operations preserving X_k^a, and let $\mathcal{F}_\omega^a = \bigcap_{k=2}^{\infty} \mathcal{F}_k^a$.

Theorem 8 ([16]). *Let* $\mathbf{A}$ *be a finite strictly simple surjective algebra. If* $\mathbf{A}$ *has no one-element subalgebras, then* $\mathbf{A}$ *is term equivalent to one of the following: (a)* $(A; \mathcal{R}(G))$ *for a regular permutation group* G *acting on* A*; (b)* $(A; \mathsf{Term}_{id}(_{(\mathsf{End}\ _K\overline{A})}\overline{A}) \cup T(\overline{A}))$ *for some vector space* $_K\overline{A} = (A; +, K)$ *over a finite field* K*; or (c)* (A, G) *for a primitive permutation group* G *on* A*. If* $\mathbf{A}$ *has one-element subalgebras, then* $\mathbf{A}$ *is idempotent and term equivalent to one of the following algebras:*

- *(a°)* $(A; \mathcal{R}_{id}(G))$ *for a permutation group* G *on* A *such that every nonidentity member of* G *has at most one fixed point;*
- *(b°)* $(A; \mathsf{Term}_{id}(_{(\mathsf{End}\ _K\overline{A})}\overline{A}))$ *for some vector space* $_K\overline{A}$ *over a finite field* K*;*
- *(d)* $(A; \mathcal{R}_{id}(G) \cap \mathcal{F}_k^a)$ *for some* k $(2 \leq k \leq \omega)$*, some element* $a \in A$*, and some permutation group* G *acting on* A *such that* a *is the unique fixed point of every nonidentity member of* G*;*
- *(e)* $(A; F)$ *where* $|A| = 2$ *and* F *contains a semilattice operation; or*
- *(f) a two-element algebra with an empty set of basic operations.*

Using the results from Section 3, we can give the following classification of approximability of MAX SOL$(\mathbf{A})$ for finite strictly simple surjective algebras $\mathbf{A}$.

Theorem 9. *Let* $\mathbf{A}$ *be a finite strictly simple surjective algebra. Then,* MAX SOL$(\mathbf{A})$ *is either in* **PO***, it is* **APX***-complete, it is* **poly-APX***-complete, or it is* **NP***-hard to find a solution.*

Proof (sketch). If $\mathbf{A}$ is of type (c) or (f), then CSP$(\mathbf{A})$ is **NP**-complete [3]. If $\mathbf{A}$ is of type (a) or $(a°)$, then the discriminator operation $t(x, y, z)$ is a term operation of $\mathbf{A}$ and tractability follows from Proposition 3(1). If $\mathbf{A}$ is of type (b) or $(b°)$, then **APX**-completeness follows from Corollary 6. If $\mathbf{A}$ is of type (d), then from [2,3,7], one can deduce tractability when $a = \max A$, membership in **poly-APX** when $a = 0 \in A$ and membership in **APX** otherwise. To prove **APX**- and **poly-APX**-hardness in the relevant cases, note that $X_k^a \in \mathsf{Inv}(\mathbf{A})$ for some $k \geq 2$ and that $\mathbf{A}$ is idempotent. The relation $r = (A \times \{a\}) \cup (\{a\} \times A)$ is pp-definable in $\{X_k^a\} \cup \mathcal{C}_A$ via $r(x, y) \equiv_{pp} \exists z : X_k^a(x, y, z, \ldots, z) \wedge \{\max A\}(z)$ and the max-core of r is the relation $\{(a, a), (a, \max A), (\max A, a)\}$. Thus, **APX**- and **poly-APX**-hardness follows from Lemma 2 combined with Proposition 3(2) and 3(3). Finally, if $\mathbf{A} = (A; F)$ is of type (e), then since $|A| = 2$, the result follows from Lemma 7. □

5 Symmetric Algebras

A bijective homomorphism from $\mathbf{A}$ to itself is called an automorphism. An algebra $\mathbf{A}$ is *symmetric* (in the sense of Szendrei [15]) if for every subalgebra $\mathbf{B} = (B; F)$

of $\mathbf{A}$, (1) the automorphism group of $\mathbf{B}$ acts primitively on B; and (2) for any set $C \subseteq A$ with $|C| = |B|$, C supports a subalgebra of $\mathbf{A}$ isomorphic to $\mathbf{B}$. Examples of symmetric algebras include homogeneous algebras and algebras whose automorphism group contains the alternating group. Condition (2) on symmetric algebras implies that if $\mathbf{B} = (B; F|_B)$ is a proper subalgebra of $\mathbf{A}$, then $(C; F|_C)$ is a subalgebra of $\mathbf{A}$ whenever C is a subset of A with $|C| \leq |B|$. Consequently, we can assign a number $\nu(\mathbf{A})$, $0 \leq \nu(\mathbf{A}) \leq |A| - 1$, to every symmetric algebra such that a proper subset $B \subset A$ is the universe of a subalgebra of $\mathbf{A}$ if and only if $|B| \leq \nu(\mathbf{A})$. One may note that $\nu(\mathbf{A}) \geq 1$ if and only if $\mathbf{A}$ is idempotent.

We need some notation for describing symmetric algebras: a bijective homomorphism is called an *isomorphism* and an isomorphism between two subalgebras of an algebra $\mathbf{A}$ is called an *internal isomorphism* of $\mathbf{A}$. The set of all internal isomorphisms will be denoted $\mathsf{Iso}\ \mathbf{A}$. A $k \times l$ *cross* $(k, l \geq 2)$ is a relation on A^2 of the form $X(B_1, B_2, b_1, b_2) = (B_1 \times \{b_2\}) \cup (\{b_1\} \times B_2)$, where $b_1 \in B_1$, $b_2 \in B_2$, $|B_1| = k$, and $|B_2| = l$. Let $\mathcal{D}_1$ denote the clone of all idempotent operations on A, and let $\mathcal{E}_1$ denote the subclone of $\mathcal{D}_1$ consisting of all operations which in addition preserve every relation $L_{a,b} = \{(a,a,a), (a,b,b), (b,a,b), (b,b,a)\}$ where $a, b \in A$ and $a \neq b$. For $2 \leq m \leq |A|$, let $\mathcal{D}_m$ be the clone of all operations in $\mathcal{D}_1$ preserving every $m \times 2$ cross. For $2 \leq m \leq |A|$, let $\mathcal{E}_m$ be the clone consisting of all operations $f \in \mathcal{D}_1$ for which there exists a projection p agreeing with f on every m-element subset B of A.

Theorem 10 ([15]). *Let $\mathbf{A}$ be a finite symmetric algebra. If $\mathbf{A}$ is not idempotent, then $|A|$ is prime and there is a cyclic group $\overline{A} = (A; +)$ such that $\mathbf{A}$ is term equivalent to $(A; \mathcal{R}(T(\overline{A})))$, $(A; \mathsf{Term}_{id}(\overline{A}) \cup T(\overline{A}))$, or $(A; T(\overline{A}))$. If $\mathbf{A}$ is idempotent, then $\mathbf{A}$ is term equivalent to one of the following algebras:*

1. $(A; \mathcal{R}(\mathsf{Iso}\ \mathbf{A}) \cap \mathcal{D}_m)$ *for some m with $1 \leq m \leq \nu(\mathbf{A})$ or $m = |A|$;*
2. $(A; \mathcal{R}(\mathsf{Iso}\ \mathbf{A}) \cap \mathcal{E}_m)$ *for some m with $1 \leq m \leq \nu(\mathbf{A})$ or $m = |A|$;*
3. $(A; \mathsf{Term}_{id}({}_K\overline{A}))$ *for a 1-dimensional vector space ${}_K\mathbf{A} = (A; +, K)$ over a finite field K; or*
4. $(A; \mathsf{Term}_{id}(\overline{A}))$ *for a 4-element abelian group $\overline{A} = (A; +)$ of exponent 2.*

Theorem 11. *Let $\mathbf{A}$ be a symmetric algebra. Then,* Max Sol$(\mathbf{A})$ *is either in* **PO**, *it is* **APX**-*complete, it is* **poly-APX**-*complete, or it is* **NP**-*hard to find a solution.*

Proof (sketch). If $\mathbf{A}$ is not idempotent, then one can show that $\mathbf{A}$ is strictly simple and surjective. In this case, the the result follows from Theorem 9.

Assume instead that $\mathbf{A}$ is idempotent; cases 3 and 4 are now immediately covered by Corollary 6 so we suppose that $\mathbf{A} = (A; \mathcal{R}(\mathsf{Iso}\ \mathbf{A}) \cap \mathcal{D}_m)$. If $m = 1$, then t is a term operation and tractability follows from Proposition 3(1). When $m > 1$, then d is a term operation and the membership results in **APX** and **poly-APX** follow. Furthermore, all $m \times 2$ crosses are in $\mathsf{Inv}\mathbf{A}$ and hardness can be shown by utilising max-cores.

Finally assume that $\mathbf{A} = (A; \mathcal{R}(\mathsf{Iso}\ \mathbf{A}) \cap \mathcal{E}_m)$ for some $1 \leq m \leq \nu(\mathbf{A})$ or $m = |A|$. When $m = 1$, then membership in **APX** can be shown based on the fact that s is a term operation; hardness can be proved by using Lemma 7. If $m > 1$, then $\mathbf{A}$ contains 2-element subalgebras $\{a, b\}$ such that each operation in $\mathbf{A}$ restricted to $\{a, b\}$ is a projection. Consequently, Max Sol$(\mathbf{A})$ is **NP**-hard for all $m \geq 2$. □

By following the proof of Theorem 11, the complexity of $\textsc{Csp}(\mathbf{A})$ can be determined, too. We note that this result agrees with Conjecture 7.5 in [3] on the source of intractability in finite, idempotent algebras.

Theorem 12. *Let* $\mathbf{A}$ *be an idempotent symmetric algebra. If there exists a nontrivial homomorphic image* $\mathbf{B}$ *of a subalgebra of* $\mathbf{A}$ *such that the operations of* $\mathbf{B}$ *are all projections, then* $\textsc{Csp}(\mathsf{Inv}(\mathbf{A}))$ *is* **NP***-complete. Otherwise,* $\textsc{Csp}(\mathsf{Inv}(\mathbf{A}))$ *is in* **P**.

6 Discussion

The results in this paper together with the approximability classifications in [7,10] provide support for the following conjecture: for every constraint language Γ over a finite domain $D \subseteq \mathbb{N}$, $\textsc{Max Sol}(\Gamma)$ is either polynomial-time solvable, **APX**-complete, **poly-APX**-complete, it is **NP**-hard to obtain a solution of nonzero measure, or it is **NP**-hard to obtain any solution. Where the exact borderlines between the cases lie is largely unknown, though, and a plausible conjecture seems remote for the moment. Therefore, we present a selection of questions that may be considered before attacking the 'main' approximability classification for $\textsc{Max Sol}$.

It has been observed that classifying the complexity of $\textsc{Csp}$ for all strictly simple algebras could be seen as a possible *'base case for induction'* [3]. This is due to the necessary condition that for a tractable algebra, all of its subalgebras and homomorphic images must be tractable. Furthermore, it is sufficient to study surjective algebras with respect to $\textsc{Csp}$ since the application of a unary polymorphism to a set of relations does not change the complexity of the set [5]. For $\textsc{Max Sol}$, however, it is possible to turn an **APX**-hard problem into a problem in **PO** by applying a unary polymorphism f, unless it satisfies some additional condition, such as $f(a) \geq a$ for all $a \in A$. It therefore looks appealing to replace the property of the algebra being surjective by that of the constraint language being a max-core. It should be noted that being a max-core is not a purely algebraic property and that we do not know how, or if, it is possible to obtain a usable characterisation of such algebras.

Kuivinen [11] has given tight inapproximability bounds (provided that $\mathbf{P} \neq \mathbf{NP}$) for the problem of solving systems of equations with integer coefficients over an arbitrary abelian group. In [7], this problem was shown to be **APX**-hard for cyclic groups of prime order. Theorem 4 extends this result to show **APX**-hardness when the underlying group is a finite vector space and the sum of the coefficients is 0. The next step would be to prove **APX**-hardness for arbitrary abelian groups. We note that the proof of Theorem 4 relies on a result which utilises the subspace structure of finite vector spaces. Informally, this is needed to be able to distinguish one affine hyperplane from the average of the others. Unfortunately, it is not hard to find an abelian group on $A = \{0, 1, 2, 4, 5, 6\}$ in which the sum of the elements in all cosets of the same nontrivial subgroup is the same: let $\overline{A}$ be the abelian group defined by the isomorphism $f : \mathbb{Z}_6 \to \overline{A}$, where $f(0) = 0, f(1) = 1, f(2) = 4, f(3) = 6, f(4) = 5$ and $f(5) = 2$. It is easy to check that if $a + H$ is a coset with $|H| > 1$, then $\sum_{a' \in a+H} a' = \frac{|H|}{|A|} \sum_{a \in A} a = 3 \cdot |H|$. A deeper analysis of these abelian groups is thus needed to settle this case.

Algebras preserving relations obtained from unrestricted systems of equations over abelian groups are full idempotent reducts of affine algebras. Corollary 6 covers some nonidempotent cases as well, due to the restriction of the coefficients in the equations defining $\Lambda_0({}_K\overline{A})$. However, in the proof of Lemma 7 we find an example of a constraint language corresponding to an affine algebra which is $\max A$-valid. The question which arises is, whether or not MAX SOL$(\mathbf{A})$ can be shown to be **APX**-hard for all affine algebras $\mathbf{A}$ such that $\mathsf{Inv}(\mathbf{A})$ is not $\max A$-valid. As observed in the introduction, this is probably a key question when it comes to deciding the approximability of MAX SOL$(\mathbf{A})$ when $\mathbf{A}$ is, for example, para-primal.

References

1. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation. Springer, Heidelberg (1999)
2. Bulatov, A.: Combinatorial problems raised from 2-semilattices. Journal of Algebra 298, 321–339 (2006)
3. Bulatov, A., Jeavons, P., Krokhin, A.: Classifying the computational complexity of constraints using finte algebras. SIAM Journal on Computing 34(3), 720–742 (2005)
4. Goldman, J., Rota, G.-C.: On the foundations of combinatorial theory IV. Finite vector spaces and Eulerian generating functions. Studies in Appl. Math. 49, 239–258 (1970)
5. Jeavons, P.: On the algebraic structure of combinatorial problems. Theoretical Computer Science 200(1-2), 185–204 (1998)
6. Jeavons, P., Cohen, D., Gyssens, M.: Closure properties of constraints. Journal of the ACM 44(4), 527–548 (1997)
7. Jonsson, P., Kuivinen, F., Nordh, G.: MAX ONES generalized to larger domains. SIAM Journal on Computing 38(1), 329–365 (2008)
8. Jonsson, P., Nordh, G.: Generalised integer programming based on logically defined relations. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 549–560. Springer, Heidelberg (2006)
9. Jonsson, P., Nordh, G., Thapper, J.: The maximum solution problem on graphs. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 228–239. Springer, Heidelberg (2007)
10. Khanna, S., Sudan, M., Trevisan, L., Williamson, D.P.: The approximability of constraint satisfaction problems. SIAM Journal on Computing 30(6), 1863–1920 (2000)
11. Kuivinen, F.: Tight approximability results for the maximum solution equation problem over Z_p. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 628–639. Springer, Heidelberg (2005)
12. Pöschel, R., Kalužnin, L.: Funktionen- und Relationenalgebren. DVW, Berlin (1979)
13. Post, E.: The two-valued iterative systems of mathematical logic. Annals of Mathematical Studies 5, 1–122 (1941)
14. Szendrei, Á.: Clones in Universal Algebra. Seminaires de Mathématiques Supérieures, vol. 99. University of Montreal (1986)
15. Szendrei, Á.: Symmetric algebras. In: Contributions to General Algebra 6, pp. 259–280. Verlag Hölder-Pichler-Tempsky, Wien and Verlag Teubner, Stuttgart (1989)
16. Szendrei, Á.: Simple surjective algebras having no proper subalgebras. J. Austral. Math. Soc. ser. A 48, 434–454 (1990)

Complete Complexity Classification of Short Shop Scheduling[⋆]

Alexander Kononov[1,2], Sergey Sevastyanov[1,2], and Maxim Sviridenko[3]

[1] Sobolev Institute of Mathematics, Novosibirsk, Russia
[2] Novosibirsk State University, Novosibirsk, Russia
[3] IBM T.J. Watson Research Center, Yorktown Heights, USA

Abstract. In this paper a comprehensive complexity analysis of classical shop scheduling problems (open shop, job shop and mixed shop) is presented subject to joint constraints on several problem parameters, such as the maximum processing time of an operation, the maximum number of operations per job, and the schedule length. Thus, our research continues the research line on the complexity analysis of short scheduling initiated for the open shop and job shop problems in the paper by Williamson et al. (1997). We improve upon some of the results in that paper.

1 Introduction

The open shop, job shop and mixed shop scheduling problems are among the classical and well-studied scheduling models. These problems appear ubiquitously in the modelling of many real-life phenomena such as industrial manufacturing production lines, packet exchanges in communication networks, timetabling etc. All these problems received a lot of attention from researchers in different communities [8]. One of the most important questions is to understand the boundary between efficiently (polynomially) solvable cases of the problem and NP-complete ones. The shop scheduling problems are well-known to be notoriously difficult both from practical and theoretical perspectives. There are only few efficient algorithms known in very restricted settings such as constant number of jobs or two machines, unit processing times and very special operation precedence structure. Most other special cases are known to be NP-complete [3,12]. A practical proof of this intractability is that a small example with 10 jobs and 10 machines posed by Fisher and Thompson [5] in 1963 remained open for over 20 years until it was solved by Carlier and Pinson [2].

One of the popular ways to draw a boundary between easy and hard scheduling problems with an integer valued objective function $F(S)$ is to show that checking the existence of a schedule S with value $F(S) \leq K-1$ is a polynomially solvable problem while the problem of deciding if there exists a schedule

[⋆] Research of the first and second authors is partially supported by RFBR grant no. 08-01-00370, Russian-Taiwan grant no. 08-06-92000. Research of the first author is partially supported by ADTP grant 2.1.1.3235.

S with $F(S) \leq K$ is NP-complete. The first such example in scheduling is the proof that deciding if there exists a schedule of length three for identical parallel machine scheduling with precedence constraints is an NP-complete problem due to Lenstra and Rinnoy Kan [13]. Later this approach was applied to unrelated parallel machine scheduling [14], scheduling with communication delays [9], job shop and open shop scheduling [20], no-wait job shop scheduling [1] and many other scheduling models. Note also that proving such an NP-completeness result for some integer constant K and an integer valued objective function implies a non-approximability result, that the existence of an approximation algorithm with performance guarantee better than $(K+1)/K$ implies $P = NP$.

We now formally define the shop scheduling problems.

Problem statement. A *job shop* is a multi-stage production process with the property that a given set of jobs $\{J_1, \ldots, J_n\} \doteq \mathcal{J}$ have to be processed on a given set of *machines* $\{M_1, \ldots, M_m\} \doteq \mathcal{M}$, and each job may have to pass through several machines. More exactly, each job J_j is a chain of m_j *operations* $O_{1j}, \ldots, O_{m_j j}$. Every operation O_{ij} is preassigned to a machine M_{ij} on which it has to be processed for p_{ij} time units. The value p_{ij} is called its *processing time* or *length*. In any feasible schedule for those n jobs, at any moment in time every job is processed by at most one machine and every machine executes at most one job. Furthermore, for each job J_j, the operation $O_{i-1,j}$ is always completed before starting the operation O_{ij}, and each operation is processed without interruption on the machine to which it is assigned. In an *open shop* the ordering of the operations in a job is not fixed and may be chosen by the scheduler. In a *mixed shop* there are jobs of the "open shop type" with any order between operations and jobs of the "job shop type" with chain precedence constraints between operations. For further references and a survey of the area, see Lawler, Lenstra, Rinnooy Kan, and Shmoys [12] and Chen, Potts, and Woeginger [3].

Our goal is to find a feasible schedule that minimizes the *makespan* (or *length*) C_{max} of the schedule, i.e., the maximum completion time among all jobs. The minimum makespan among all feasible schedules is denoted by C^*_{max}. Following, the standard three-field scheduling notations we denote job shop, open shop and mixed shop scheduling problems as $J||C_{max}$, $O||C_{max}$ and $X||C_{max}$. The middle entry in the three-field notation will be used either to constrain the maximal number of operations per job, denoted by *op* (e.g., "*op* ≤ 3" will mean "at most three operations per job"), or to constrain the set of possible processing times (e.g., "$p_{ij} = 1$").

Known Results. Williamson et al. [20] proved that deciding if there exists a schedule of length three and finding such a schedule can be solved in polynomial time for the open shop and job shop scheduling problems. On the negative side they showed that for the job shop scheduling with at most three operations per job and unit processing times, i.e., $J|p_{ij} = 1, op \leq 3|C_{max}$, deciding if there exists a schedule of length four is an NP-complete problem. They also proved that for the open shop scheduling with at most three operations per job and processing times $p_{ij} \in \{1, 2\}$, i.e., $O|p_{ij} \in \{1, 2\}, op \leq 3|C_{max}$, deciding if there exists a schedule of length four is an NP-complete problem. Note that the open shop problem with $p_{ij} \in \{0,1\}$ is just the bipartite edge-coloring problem and therefore is a polynomially solvable problem.

The comprehensive survey paper by Shakhlevich et al. [18] reviews many results on the complexity of the mixed shop scheduling problem. The complexity of the problem was analyzed in the dependence of restrictions imposed on some problem parameters, such as the number of machines, the number of the "job-shop-type" jobs, and the number of the "open-shop-type" jobs. However, most of these results are not directly related to the topic of this paper that is concentrated on the complexity of finding "short schedules".

In the papers [10],[11] the authors provide a similar multi-parametric analysis of the complexity for the connected list coloring and the generalized open shop problem.

Our Results. In this paper we consider the three classical shop scheduling models: the mixed shop, the job shop and the open shop scheduling problems. We find the exact boundary for the schedule length K such that finding a schedule with the makespan $\leq K-1$ is "easy" and finding a schedule with the makespan K is NP-complete. We also classify easy and hard instances depending on their set of allowable processing times and maximum number of non-zero operations per job. More precisely, we show that

1. deciding if there exists a schedule of length four for the open shop scheduling problem with at most two operations with processing times $p_{ij} \in \{1,2\}$, i.e., $O|p_{ij} \in \{1,2\}, op \leq 2|C_{max} \leq 4$, is an NP-complete problem (Theorem 11) (this result tightens the result from [20]);
2. the job shop scheduling problem with at most two operations per job and unit processing times, i.e., $J|p_{ij} = 1, op \leq 2|C_{max}$, is polynomially solvable (Theorem 1) (The algorithm is based on the bipartite edge coloring that produces an infeasible schedule which is later transformed into a feasible one.);
3. even more general mixed shop scheduling problem $X|p_{ij} = 1, op \leq 2|C_{max}$ is polynomially solvable (Theorem 2) (The algorithm is a combination of a very efficient graph orientation algorithm and the previous algorithm for the job shop scheduling problem. Although the general graph orientation problem can be reduced to a Max Flow computation we show that for the instances corresponding to the mixed shop scheduling problem there exists a linear time algorithm.);
4. deciding if there exists a schedule of length four for the job shop scheduling problem with at most two operations per job with processing times $p_{ij} \in \{1,2\}$, i.e., $J|p_{ij} \in \{1,2\}, op \leq 2|C_{max} \leq 4$, is an NP-complete problem (Theorem 8) (The proof of this result is highly technical and uses new type of gadgets, different from the ones used in [20].);
5. deciding if there exists a schedule of length two for the mixed shop scheduling problem, i.e., $X||C_{max} \leq 2$, is a polynomially solvable problem (Theorem 3);
6. the problem of deciding whether an instance of $X|p_{ij} = 1, op \leq 3|C_{max}$, i.e., mixed shop scheduling problem with at most three unit operations per job has a schedule of length at most three is NP-complete (Theorem 4);

7. the problem of deciding whether an instance of $X|p_{ij} \in \{1,2\}, op \leq 2|C_{max}$ has a schedule of length at most three is NP-complete (Theorem 5).

2 Polynomial Time Algorithm for the Mixed Shop Scheduling Problem with at Most Two Unit Operations per Job

We start with the reformulation of the multigraph edge coloring theorem due to Melnikov and Vizing [15] (Theorem 2.2.1) for the job shop scheduling with unit processing times and then show how to generalize it to the mixed shop scheduling problem. Note that this result was first proved in [15], then generalized by Pyatkin [16]. Finally, a very simple proof was provided by Vizing [19].

We partition the set of jobs into two subsets $\mathcal{J} = \overrightarrow{\mathcal{J}} \cup \mathcal{J}_e$. The subset $\overrightarrow{\mathcal{J}}$ is the set of jobs with exactly two non-zero operations that must be processed on different machines, and $\mathcal{J}_e$ is the set of the remaining ("easy") jobs that either consist of at most one non-zero operation or have two consecutive operations on the same machine. Note that the original proof from [15] works in the case when $\mathcal{J}_e = \emptyset$ but can be easily generalized to handle jobs from $\mathcal{J}_e$.

Let $\Delta_k = \sum_{O_{ij}|M_{ij}=M_k} p_{ij}$ be the total load of machine M_k, and let $\Delta_{1k} = \sum_{J_j \in \overrightarrow{\mathcal{J}}|M_{1j}=M_k} p_{1j}$ and $\Delta_{2k} = \sum_{J_j \in \overrightarrow{\mathcal{J}}|M_{2j}=M_k} p_{2j}$ be the total length of the first and second operations of jobs from $\overrightarrow{\mathcal{J}}$ on machine M_k respectively. Obviously, $\Delta_k \geq \Delta_{1k} + \Delta_{2k}$. Since in this section we consider shop scheduling problems with unit processing times, the load of a machine is equal to the number of operations that must be processed on that machine. Since each machine can process at most one operation at a time and since every job can be processed by at most one machine at a time, we obtain the following lower bound on the optimal makespan $C^*_{max} \geq L = \max_k \max\{\Delta_k, \Delta_{1k}+1, \Delta_{2k}+1\}$. The next theorem shows that this lower bound is tight for a special case of the job shop scheduling problem.

Theorem 1. *For any instance of the job shop scheduling problem $\langle J|p_{ij} = 1, op \leq 2|C_{max}\rangle$ there exists a schedule of length $C^*_{max} = L$ which can be found in polynomial time.*

Proof. Consider the following directed multigraph $\overrightarrow{G} = (\mathcal{M}, A)$. The vertex set of $\overrightarrow{G}$ corresponds to the set of machines and the set of arcs corresponds to jobs from $\overrightarrow{\mathcal{J}}$, i.e., jobs with two unit operations on different machines. A directed arc (M_i, M_j) corresponds to a job with the first operation on M_i and the second operation on M_j. Thus, Δ_{1k} and Δ_{2k} correspond to the outdegree and indegree of the vertex $M_k \in \mathcal{M}$. A *linear factor* in multigraph $\overrightarrow{G}$ is a subgraph such that indegree and outdegree of every vertex is at most one. Let $\delta = \max_k \max\{\Delta_{1k}, \Delta_{2k}\}$ be the maximal semi-degree in graph $\overrightarrow{G}$.

By the standard application of König's Edge Coloring Theorem we could find a decomposition of graph $\overrightarrow{G}$ into δ linear factors. Such a decomposition corresponds to an infeasible schedule of length $\delta + 1$ for the set of jobs $\overrightarrow{\mathcal{J}}$ which can be constructed as follows.

Take an arbitrary linear factor in the decomposition and schedule all corresponding first operations in the time period $[0, 1]$ and all second operations in the time interval $[1, 2]$. Repeat the procedure iteratively, taking at step τ an arbitrary remaining linear factor and scheduling all first operations in the time period $[\tau - 1, \tau]$ and all second operations in the time period $[\tau, \tau + 1]$.

This infeasible schedule is easy to make feasible if we process all first operations on machine M_k in the time interval $[0, \Delta_{1k}]$ and all second operations in the time interval $[L - \Delta_{2k}, L]$ in the order defined by the previous infeasible schedule. The feasibility of this schedule follows from the fact that first and second operations on one machine do not overlap and the relative order of the first and the second operations of the same job cannot be violated during the transformation since we shift first operations toward the time 0 and we shift second operations toward the time L. Finally, the jobs from the set $\mathcal{J}_e$ can be inserted into idle time periods left by the schedule for jobs in $\overrightarrow{\mathcal{J}}$ in a greedy way.

The running time of this algorithm is dominated by that of the algorithm of computing the linear factor decomposition in multigraph $\overrightarrow{G}$, which is equivalent to computing δ unweighted bipartite matchings in a graph with $2|\mathcal{M}|$ vertices and $|\mathcal{J}|$ edges. One of the best algorithm to solve it is due to Cole, Ost and Schirra [4] and has the running time $O(|\mathcal{J}| \log \delta)$.

We now consider the mixed shop scheduling problem $\langle X|p_{ij} = 1, op \leq 2|C_{max}\rangle$. In this problem the set of jobs is partitioned into three subsets $\mathcal{J} = \bar{\mathcal{J}} \cup \overrightarrow{\mathcal{J}} \cup \mathcal{J}_e$. The set $\overrightarrow{\mathcal{J}}$ consists of "job-shop type" jobs with exactly two unit length operations that must be processed on different machines in a prescribed order between the two operations. The set $\bar{\mathcal{J}}$ consists of "open-shop type" jobs with exactly two unit length operations that must be processed on different machines without a predefined order between those operations. And finally, the set $\mathcal{J}_e$ consists of the remaining "easy" jobs, including the jobs with both operations on the same machine and the jobs with at most one nonzero operation.

Consider some optimal schedule for $\langle X|p_{ij} = 1, op \leq 2|C_{max}\rangle$. In this schedule the jobs from $\bar{\mathcal{J}}$ have some orientation in which one operation of each job precedes the other one. Given this orientation ($\mathcal{O}$), the schedule length is equal to $L(\mathcal{O}) = \max_k \max\{\Delta_k, \Delta_{1k}(\mathcal{O}) + 1, \Delta_{2k}(\mathcal{O}) + 1\}$ by Theorem 1, where the values of Δ_{1k} and Δ_{2k} depend on orientation $\mathcal{O}$.

The orientation problems are among the well-studied problems of combinatorial optimization, see the survey in [17] (Chapter 61). The classical way to solve orientation problems is by a reduction of an orientation problem to the Hoffman's Circulation Theorem (Theorem 61.2 [17]). Finding a circulation is equivalent to one maximum flow computation (Theorem 11.3 [17]). In the next lemma we provide a more efficient way to compute the orientation $\mathcal{O}$ minimizing $L(\mathcal{O})$.

Lemma 1. *The problem of minimizing the function $L(\mathcal{O})$ over all possible orientations $\mathcal{O}$ of jobs in $\bar{\mathcal{J}}$ is solvable in $O(|\mathcal{J}|)$ time.*

Proof. Let $\Delta = \max_k \Delta_k$ be the maximum machine load. Machine M_k is called *critical* under orientation $\mathcal{O}$ if $\Delta_{1k}(\mathcal{O}) = \Delta$ or $\Delta_{2k}(\mathcal{O}) = \Delta$. By Theorem 1, under any orientation $\mathcal{O}$ of jobs in $\bar{\mathcal{J}}$ we have $\Delta \le L(\mathcal{O}) \le \Delta + 1$, and the value $L(\mathcal{O}) = \Delta$ is attained if and only if there are no critical machines under orientation $\mathcal{O}$. Thus, our aim is to find an orientation $\mathcal{O}$ which provides no critical machines (a *positive answer*), or to establish that such orientation does not exist (a *negative answer*). In particular, we have the negative answer at once if there exists a critical machine for the initial (empty) orientation of jobs in $\bar{\mathcal{J}}$. Since this can be checked immediately (by a single scanning of the input data), we can further assume that the initial orientation contains no critical machines.

Consider the following multigraph $G = (\mathcal{M}, E)$. The set of vertices $\mathcal{M}$ corresponds to the set of machines, and the set of edges E corresponds to the set of jobs from $\bar{\mathcal{J}}$; $d(M_k)$ denotes the degree of vertex $M_k \in G$. Let us formulate a few obvious propositions.

Proposition 1. *The orientation problem for edges E in G can be solved independently for each connected component of graph G, and the positive answer for the whole graph is attained if and only if it is attained for every connected component.*

Thus, while solving the orientation problem, we can assume that G is connected. Machine M_k (and the corresponding vertex $M_k \in G$) is called a *no-problem machine* (*no-problem* vertex) for a given partial orientation $\mathcal{O}'$ of jobs in $\bar{\mathcal{J}}$ (edges in E) if for no expansion $\mathcal{O}$ of orientation $\mathcal{O}'$ machine M_k can become a critical machine.

Proposition 2. *Machine M_k is a no-problem machine for a given (partial) orientation $\mathcal{O}'$ if and only if it meets one of the following properties:*
(a) machine M_k contains operations of "easy" jobs;
(b) $\Delta_k < \Delta$;
(c) $\Delta_{1k}(\mathcal{O}') > 0$ and $\Delta_{2k}(\mathcal{O}') > 0$.

Proposition 3. *If G is connected and contains a no-problem vertex then the orientation problem has a positive answer which can be obtained in linear time.*

Proof. First observe that we have $\Delta \ge 2$, because in case $\Delta = 1$ the graph G consists of a single edge and cannot contain a no-problem vertex.

Let M_0 be a no-problem vertex in $G = (\mathcal{M}, E)$. By scanning the set E, we construct a spanning tree $T = (\mathcal{M}, E_T)$, compute the *tree-degree* $d_T(M)$ of each vertex M in the tree, and orient all edges from $E \setminus E_T$ arbitrarily. Scan the set of vertices $\mathcal{M}$ and make up the list $\mathcal{L}$ of *leaves* of T (i.e., vertices $M \in \mathcal{M}$ with $d_T(M) = 1$) except M_0. Let $M \in \mathcal{L}$ be the first leaf in the list, and let $e = (M, M')$ be the only edge of T incident to M. If M is a no-problem vertex, orient e arbitrarily. Otherwise, M is incident to an arc $e' \in A$ in $\overrightarrow{G}$, and we make M a no-problem vertex by orienting e in the direction opposite to e'. Delete M from tree T and decrease $d_T(M')$ by 1. If M' becomes a leaf ($d_T(M') = 1$), we add it to the end of list $\mathcal{L}$ and continue the above procedure, until vertex M_0

remains the only vertex of the tree. Once the only remaining vertex is M_0, we obtain the desired orientation with the positive answer.

It is clear that all steps of the procedure can be implemented in time linear in the number of edges in G.

Due to Proposition 3 we can further assume that graph G contains no no-problem vertices. Vertex M of graph G (and the corresponding machine $M \in \mathcal{M}$) is called *sub-critical* if it has the unit degree $d(M) = 1$ and one of two possible orientations of the edge incident to M makes machine M critical.

Suppose that graph G contains a vertex M of degree 1. It is clear that M may be either a no-problem vertex or a sub-critical vertex. Since we agreed that the first case is excluded, we should only consider the case when there is a sub-critical vertex M in G. The single edge e incident to M has got the unique "positive" orientation under which the positive answer is still possible. This conditions the following step of the orientation algorithm.

Eliminating the unit-degree vertices. Compute the degrees of all vertices of graph G and make the list SC of all sub-critical vertices. Take the first vertex $M' \in SC$, choose the unique "positive" orientation of its single edge $e = (M', M'')$, remove vertex M' from the list and from graph G, decrease by 1 the degree of vertex M''. If $d(M'')$ became equal to zero, this means that e was the last edge in graph G, and that the process of orientation of edges came to an end. At that, if machine M'' became critical, this means that the answer is **negative** and we could not avoid this. Otherwise, we have a **positive** answer.

Alternatively, if $d(M'')$ became equal to 1, there may be two possible cases: M'' may become either a no-problem machine or a sub-critical machine. In the first case we can complete the orientation process with the positive answer, due to Proposition 3. In the second case we add machine M'' to the end of the list SC and continue the process.

The described process may finish with two possible outcomes. Either it exhausts all edges in G (with the positive or negative answer), or it exhausts all unit degree vertices. We claim that the second case also provides the positive answer.

Finding an orientation with positive answer in the case when the degree of each vertex in G is at least 2. Starting with an arbitrary vertex $M \in G$ and an arbitrary edge incident to M, we easily find a cycle C in graph G. Let us orient all edges of C "clockwise", thereby removing them from graph G. Then each vertex of C will get one additional incoming arc and one additional outgoing arc, thus becoming a no-problem vertex for the obtained partial orientation. After removing the edges of C the resulting graph G' may become disconnected. But since G was connected, each vertex of G' is connected with a (no-problem) vertex of cycle C, and we can apply the algorithm of Proposition 3 providing a positive answer.

Obviously, all parts of the described above orientation algorithm can be implemented in time linear in the number of jobs, which completes the proof of Lemma 1.

Theorem 1 and Lemma 1 imply the following

Theorem 2. *The mixed shop scheduling problem* $\langle X|p_{ij} = 1, op \leq 2|C_{max}\rangle$ *can be solved in time* $O(n \log n)$.

High Multiplicity encoding. Note that the algorithms from the Theorems 1 and 2 are polynomial only under the unary encoding of the input. Under the more efficient encoding when we have at most Λ job types and we just keep the number of jobs λ_k for each type $k = 1, \ldots, \Lambda$ in the binary representation, the number of jobs $n = |\mathcal{J}|$ might not be polynomial in the input size. Fortunately, there are algorithms for the bipartite edge coloring problem that perform well in such situations [17]. The algorithm of Gonzalez and Sahni [7] has running time $O(\Lambda^2)$ and the algorithm of Gabow and Kariv [6] has the running time $O(m\Lambda \log \mu)$ where μ is the maximum job multiplicity.

The algorithm of the Theorem 2 is easy to implement in the time $O(\Lambda)$. We just need to notice that if we have at least two parallel edges between vertices u and v in the graph G or equivalently at least two identical "open shop type" jobs then we could orient them in the opposite directions. Both vertices u and v become no-problem vertices (machines) after such orientation. Therefore, it is enough to consider the case when there is a unique job of "open shop type" for every pair of machines.

3 Complexity Results for Short Shop Scheduling

In the beginning of the Section we consider mixed shop scheduling problems, providing the tight complexity classification for those problems. Next we obtain new complexity results for job shop and open shop scheduling problems. Due to the space limitation we omit the proofs from this extended abstract.

Theorem 3. *The decision problem* $\langle X || C_{max} \leq 2\rangle$, *i.e., the problem of deciding whether a given instance of the mixed shop problem has a schedule of length at most 2 is solvable in linear time.*

Theorem 4. *The decision problem* $\langle X|p_{ij} = 1, op \leq 3|C_{max} \leq 3\rangle$, *i.e., the problem of deciding whether a given instance of the mixed shop scheduling problem with at most 3 operations per job and unit processing times has a schedule of length at most 3, is NP-complete.*

Theorem 5. *The decision problem* $\langle X|p_{ij} \in \{1, 2\}, op \leq 2|C_{max} \leq 3\rangle$, *i.e., the problem of deciding whether a given instance of the mixed shop scheduling problem with at most 2 operations per job and processing times 1 or 2 has a schedule of length at most 3, is NP-complete.*

Theorem 6 (Williamson et al. [20]). *The decision problem* $\langle J|p_{ij} = 1, op \leq 3|C_{max} \leq 4\rangle$, *i.e., the problem of deciding whether a given instance of the job shop scheduling problem with at most 3 operations per job and unit processing times has a schedule of length at most 4, is NP-complete.*

The matching positive result is obtained by a reduction to the well-known 2SAT problem.

Theorem 7 (Williamson et al. [20]). *The decision problem* $\langle J||C_{max} \le 3\rangle$ *is polynomially solvable.*

Theorems 1, 6 and 7 raise a natural question. What is the complexity status of the job shop scheduling problem with at most two operations per job and non-unit processing times?

Theorem 8. *The decision problem* $\langle J\,|\,p_{ij} \in \{1,2\}, op \le 2\,|\,C_{max} \le 4\rangle$*, i.e., the problem of deciding whether a given instance of the job shop problem with at most 2 operations per job and processing times 1 or 2 has a schedule of length at most 4, is NP-complete.*

Theorem 9 (Williamson et al. [20]). *The decision problem* $\langle O|p_{ij} \in \{1,2\}, op \le 3|C_{max} \le 4\rangle$*, i.e., the problem of deciding whether a given instance of the open shop problem with at most 3 operations per job and processing times one or two has a schedule of length at most 4, is NP-complete.*

The corresponding positive result is obtained by using an algorithm for weighted bipartite matching.

Theorem 10 (Williamson et al. [20]). *The decision problem* $\langle O||C_{max} \le 3\rangle$ *is polynomially solvable.*

The following theorem is a refinement of the above negative result to a more restrictive instances with at most two operations per job.

Theorem 11. *The decision problem* $\langle O|p_{ij} \in \{1,2\}, op \le 2|C_{max} \le 4\rangle$*, i.e., the problem of deciding whether a given instance of the open shop problem with at most 2 operations per job and processing times one or two has a schedule of length at most 4, is NP-complete.*

4 Conclusion

In this paper we provided a complete complexity classification of the decision problems for job shop, open shop and mixed shop scheduling models with respect to different combinations of joint constraints on problem parameters, such as: the maximum number of operations per job, the maximum processing time of an operation, and the upper bound on schedule length. We established that the family of subproblems has a basis system consisting of ten subproblems, five of which are polynomially solvable and another five are NP-complete.

The natural question is to extend our classification to additional problem parameters, such as the maximum number of operations per machine, the number of machines, and the number of jobs.

It would be also of interest to obtain a similar problem classification for the preemptive shop scheduling problems. The most difficult problem is to understand the preemptive mixed shop scheduling case, since we could have optimal schedules with preemptions in non-integral time moments for very simple instances of this problem.

References

1. Bansal, N., Mahdian, M., Sviridenko, M.: Minimizing makespan in no-wait job shops. Math. Oper. Res. 30(4), 817–831 (2005)
2. Carlier, J., Pinson, E.: An algorithm for solving the job-shop problem. Management Sci. 35(2), 164–176 (1989)
3. Chen, B., Potts, C., Woeginger, G.: A review of machine scheduling: complexity, algorithms and approximability. Handbook of combinatorial optimization, vol. 3, pp. 21–169. Kluwer Acad. Publ., Boston (1998)
4. Cole, R., Ost, K., Schirra, S.: Edge-coloring bipartite multigraphs in $O(E \log D)$ time. Combinatorica 21, 5–12 (2001)
5. Fischer, J., Thompson, G.: Industrial Scheduling. Prentice Hall, Englewood Cliffs (1963)
6. Gabow, H., Kariv, O.: Algorithms for edge coloring bipartite graphs and multigraphs. SIAM J. Comput. 11, 117–129 (1982)
7. Gonzalez, T., Sahni, S.: Open shop scheduling to minimize finish time. J. Assoc. Comput. Mach. 23(4), 665–679 (1976)
8. Handbook of scheduling. In: Leung, J.Y.-T. (ed.) Algorithms, models, and performance analysis. Chapman & Hall/CRC Computer and Information Science Series. Chapman & Hall/CRC, Boca Raton (2004)
9. Hoogeveen, J., Lenstra, J., Veltman, B.: Three, four, five, six, or the complexity of scheduling with communication delays. Oper. Res. Lett. 16(3), 129–137 (1994)
10. Kashyrskikh, K.N., Sevastianov, S.V., Tchernykh, I.D.: 4-parameter complexity analysis of the open shop problem (in Russian). Diskret. Analiz. i Issled. Oper. 7(4), 59–77 (2000)
11. Kononov, A., Sevastianov, S.: On the complexity of the connected list vertex-coloring problem (in Russian). Diskret. Analiz. i Issled. Oper. 7(2) (Ser. 1), 21–46 (2000)
12. Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B.: Sequencing and scheduling: algorithms and complexity. In: Graves, S., Rinnooy Kan, A.H.G., Zipkin, P. (eds.) Handbooks in Operations Research and Management Science. Logistics of Production and Inventory, vol. 4, pp. 445–522. North-Holland, Amsterdam (1993)
13. Lenstra, J., Rinnooy Kan, A.H.G.: Complexity of scheduling under precedence constraints. Operations Research 26(1), 22–35 (1978)
14. Lenstra, J., Shmoys, D., Tardos, E.: Approximation algorithms for scheduling unrelated parallel machines. Math. Programming 46(3) (Ser. A), 259–271 (1990)
15. Melnikov, L., Vizing, V.: The edge chromatic number of a directed/mixed multigraph. J. Graph Theory 31(4), 267–273 (1999)
16. Pyatkin, A.: Incidentor Colorings and their applications, Ph. D. Thesis (1999)
17. Schrijver, A.: Combinatorial optimization. Polyhedra and efficiency. In: Algorithms and Combinatorics, vol. 24, B. Springer, Berlin (2003)
18. Shakhlevich, N.V., Sotskov, Y.N., Werner, F.: Complexity of mixed shop scheduling problems: A survey. European Journal of Operational Research 120, 343–351 (2000)
19. Vizing, V.: A bipartite interpretation of a directed multigraph in problems of the coloring of incidentors (Russian). Diskretn. Anal. Issled. Oper. 9(1) (Ser. 1), 27–41 (2002)
20. Williamson, D., Hall, L., Hoogeveen, J., Hurkens, C., Lenstra, J.K., Sevastianov, S., Shmoys, D.: Short shop schedules. Oper. Res. 45(2), 288–294 (1997)

Compressed Word Problems in HNN-Extensions and Amalgamated Products

Niko Haubold and Markus Lohrey[⋆]

Institut für Informatik, Universität Leipzig
{haubold,lohrey}@informatik.uni-leipzig.de

Abstract. It is shown that the compressed word problem for an HNN-extension $\langle H, t \mid t^{-1}at = \varphi(a) (a \in A)\rangle$ with A finite is polynomial time Turing-reducible to the compressed word problem for the base group H. An analogous result for amalgamated free products is shown as well.

1 Introduction

Since it was introduced by Dehn in 1910, the *word problem* for groups has emerged to a fundamental computational problem linking group theory, topology, mathematical logic, and computer science. The word problem for a finitely generated group G asks, whether a given word over the generators of G represents the identity of G, see Section 2 for more details. Dehn proved the decidability of the word problem for surface groups. On the other hand, 50 years after the appearance of Dehn's work, Novikov and independently Boone proved the existence of a finitely presented group with undecidable word problem, see [11] for references. However, many natural classes of groups with decidable word problem are known, as for instance finitely generated linear groups, automatic groups and one-relator groups. With the rise of computational complexity theory, also the complexity of the word problem became an active research area. This development has gained further attention by potential applications of combinatorial group theory for secure cryptographic systems [12].

In order to prove upper bounds on the complexity of the word problem for a group G, a "compressed" variant of the word problem for G was introduced in [7,8,15]. In the *compressed word problem* for G, the input word over the generators is not given explicitly but succinctly via a *straight-line program* (SLP for short). This is a context free grammar that generates exactly one word, see Section 2. Since the length of this word may grow exponentially with the size (number of productions) of the SLP, SLPs can be seen indeed as a succinct string representation. SLPs turned out to be a very flexible compressed representation of strings, which are well suited for studying algorithms for compressed data. In [8,15] it was shown that the word problem for the automorphism group Aut(G) of G can be reduced in polynomial time to the *compressed* word problem for G. In [7], it was shown that the compressed word problem for a finitely generated

[⋆] The second author is supported by the DFG research project ALKODA.

free group F can be solved in polynomial time. Hence, the word problem for $\mathrm{Aut}(F)$ turned out to be solvable in polynomial time [15], which solved an open problem from [6]. In [8], this result was also generalized to graph groups (also known as right-angled Artin groups).

In this paper, we prove a transfer theorem for the compressed word problem of *HNN-extensions* [3]. For a *base group* H, two isomorphic subgroups $A, B \leq H$, and an isomorphism $\varphi : A \to B$, the corresponding HNN-extension is the group

$$G = \langle H, t \mid t^{-1}at = \varphi(a)\,(a \in A)\rangle. \tag{1}$$

Intuitively, it is obtained by adding to H a new generator t (the *stable letter*) in such a way that conjugation of A by t realizes φ. The subgroups A and B are also called the *associated subgroups*. A related operation is that of the *amalgamated free product* of two groups H_1 and H_2 with isomorphic subgroups $A_1 \leq H_1$, $A_2 \leq H_2$ and an isomorphism $\varphi : A_1 \to A_2$. The corresponding amalgamated free product is the group $\langle H_1 * H_2 \mid a = \varphi(a)\,(a \in A_1)\rangle$. Intuitively, it results from the free product $H_1 * H_2$ by identifying every element $a \in A_1$ with $\varphi(a) \in A_2$. The subgroups A_1 and A_2 are also called the *amalgamated subgroups*.

HNN-extensions were introduced by Higman, Neumann, and Neumann in 1949 [3]. They proved that H embeds into the group G from (1). Modern proofs of the above mentioned Novikov-Boone theorem use HNN-extensions as the main tool for constructing finitely presented groups with an undecidable word problem [11]. In particular, arbitrary HNN-extensions do not preserve good algorithmic properties of groups like decidability of the word problem. In this paper, we restrict to HNN-extensions (resp. amalgamated free products) with finite associated (resp. amalgamated) subgroups, which is an important subcase. Stallings [16] proved that a group has more than one end if and only if it is either an HNN-extension with finite associated subgroups or an amalgamated free product with finite amalgamated subgroups. Moreover, a group is virtually-free (i.e., has a free subgroup of finite index) if and only if it can be built up from finite groups using amalgamated free products with finite amalgamated subgroups and HNN-extensions with finite associated subgroups [2].

It is not hard to see that the word problem for an HNN-extension (1) with A finite can be reduced in polynomial time to the word problem of the base group H. The main result of this paper extends this transfer theorem to the compressed setting: the compressed word problem for (1) with A finite can be reduced in polynomial time to the compressed word problem for H. In fact, we prove a slightly more general result, which deals with HNN-extensions with several stable letters $t_1, \ldots, t_n$, where the number n is part of the input. For each stable letter t_i the input contains a *partial* isomorphism φ_i from the fixed finite subgroup $A \leq H$ to the fixed finite subgroup $B \leq H$ and we consider the multiple HNN-extension $G = \langle H, t_1, \ldots, t_n \mid t_i^{-1}at_i = \varphi_i(a)\ (1 \leq i \leq n, a \in \mathrm{dom}(\varphi_i))\rangle$. Our polynomial time reduction consists of a sequence of polynomial time reductions. In a first step, we reduce the compressed word problem for G to the same problem for *reduced sequences*. These are strings (over the generators of H and the symbols $t_1, t_1^{-1}, \ldots, t_n, t_n^{-1}$) that do not contain a substring of the form $t_i^{-1}wt_i$ (resp.

$t_i w t_i^{-1}$), where the string w represents a group element from the domain (resp. range) of φ_i. In a second step, we reduce the number n of stable letters to a constant δ, which only depends on the size of the fixed subgroup A. The main step of the paper reduces the compressed word problem for reduced sequences over an HNN-extension with $k \leq \delta$ many stable letters (and associated partial isomorphisms from A to B) into two simpler problems: (i) the same problem but with only $k-1$ many stable letters and (ii) the same problem (with at most δ many stable letters) but with associated subgroups that are strictly smaller than A. By iterating this procedure, we arrive after a constant number of iterations (where each iteration is a polynomial time reduction) at a compressed word problem for which we directly know the existence of a polynomial time reduction to the compressed word problem for the base group H. Since the composition of a constant number of polynomial time reductions is again a polynomial time reduction, our main result follows.

The main reduction step in our algorithm uses techniques similar to those from [9], where a transfer theorem for solving equations over HNN-extensions with finite associated subgroups was shown.

From the close relationship of HNN-extensions with amalgamated free products, a polynomial time reduction from the compressed problem for an amalgamated free product $\langle H_1 * H_2 \mid a = \varphi(a)\,(a \in A_1)\rangle$ (with A_1 finite) to the compressed word problems of H_1 and H_2 is deduced in the final Section 4. This result generalizes a corresponding result for free products from [8].

A full version of this paper can be found at [4].

2 Preliminaries

Groups and the word problem. For background in combinatorial group theory see [11]. For a group G and two elements $x, y \in G$ we denote with $x^y = y^{-1}xy$ the conjugation of x by y. Let G be a *finitely generated group* and let Σ be a finite *group generating set* for G. Hence, $\Sigma^{\pm 1} = \Sigma \cup \Sigma^{-1}$ is a finite *monoid generating set* for G and there exists a canonical monoid homomorphism $h : (\Sigma^{\pm 1})^* \to G$, which maps a word $w \in (\Sigma^{\pm 1})^*$ to the group element represented by w. For $u, v \in (\Sigma^{\pm 1})^*$ we will also say that $u = v$ in G in case $h(u) = h(v)$. The *word problem for G w.r.t. Σ* is the following decision problem:

INPUT: A word $w \in (\Sigma^{\pm 1})^*$.
QUESTION: $w = 1$ in G?

It is well known that if Γ is another finite generating set for G, then the word problem for G w.r.t. Σ is logspace many-one reducible to the word problem for G w.r.t. Γ. This justifies one to speak just of the word problem for the group G.

The *free group* $F(\Sigma)$ generated by Σ can be defined as the quotient monoid $F(\Sigma) = (\Sigma^{\pm 1})^* / \{aa^{-1} = \varepsilon \mid a \in \Sigma^{\pm 1}\}$, where ε denotes the empty word. A *group presentation* is a pair (Σ, R), where Σ is an alphabet of symbols and R is a set of *relations* of the form $u = v$, where $u, v \in (\Sigma^{\pm 1})^*$. The group defined by this presentation is denoted by $\langle \Sigma \mid R \rangle$. It is defined as the quotient $F(\Sigma)/N(R)$, where $N(R)$ is the smallest normal subgroup of the free group $F(\Sigma)$ that contains all elements uv^{-1} with $(u = v) \in R$. In particular $F(\Sigma) = \langle \Sigma \mid \emptyset \rangle$. Of course, one can assume that all relations are of the form $r = 1$. In fact, usually the set of relations is given by a set of *relators* $R \subseteq (\Sigma^{\pm 1})^*$, which corresponds to the set $\{r = 1 \mid r \in R\}$ of relations.

The *free product* of two groups G_1 and G_2 is denoted by $G_1 * G_2$. If $G_i \simeq \langle \Sigma_i \mid R_i \rangle$ for $i \in \{1, 2\}$ with $\Sigma_1 \cap \Sigma_2 = \emptyset$, then $G_1 * G_2 \simeq \langle \Sigma_1 \cup \Sigma_2 \mid R_1 \cup R_2 \rangle$.

Straight-line programs. We are using straight-line programs as a compressed representation of strings with reoccuring subpatterns [14]. A *straight-line program (SLP) over the alphabet* Γ is a context free grammar $\mathbb{A} = (V, \Gamma, S, P)$, where V is the set of *nonterminals*, Γ is the set of *terminals*, $S \in V$ is the *initial nonterminal*, and $P \subseteq V \times (V \cup \Gamma)^*$ is the set of *productions* such that (i) for every $X \in V$ there is exactly one $\alpha \in (V \cup \Gamma)^*$ with $(X, \alpha) \in P$ and (ii) there is no cycle in the relation $\{(X, Y) \in V \times V \mid \exists \alpha : (X, \alpha) \in P, Y \text{ occurs in } \alpha\}$. A production (X, α) is also written as $X \to \alpha$. The language generated by the SLP $\mathbb{A}$ contains exactly one word $\mathrm{val}(\mathbb{A})$. Moreover, every nonterminal $X \in V$ generates exactly one word that is denoted by $\mathrm{val}(\mathbb{A}, X)$, or briefly $\mathrm{val}(X)$, if $\mathbb{A}$ is clear from the context. The size of $\mathbb{A}$ is $|\mathbb{A}| = \sum_{(X,\alpha) \in P} |\alpha|$. It can be seen easily that an SLP can be transformed in polynomial time into an SLP in *Chomsky normal form*, which means that all productions have the form $A \to BC$ or $A \to a$ for $A, B, C \in V$ and $a \in \Gamma$.

Let G be a finitely generated group and Σ a finite generating set for G. The *compressed word problem* for G w.r.t. Σ is the following decision problem:

INPUT: An SLP $\mathbb{A}$ over the terminal alphabet $\Sigma^{\pm 1}$.
OUTPUT: Does $\mathrm{val}(\mathbb{A}) = 1$ hold in G?

In this problem, the input size is $|\mathbb{A}|$. As for the ordinary word problem, the complexity of the compressed word problem does not depend on the chosen generating set. This allows one to speak of the compressed word problem for the group G. The compressed word problem for G is also denoted by $\mathrm{CWP}(G)$.

Polynomial time Turing-reductions. For two computational problems A and B, we write $A \leq_T^P B$ if A is polynomial time Turing-reducible to B. This means that A can be decided by a deterministic polynomial time Turing-machine that uses B as an oracle. Clearly, $\leq_T^P$ is transitive, and $A \leq_T^P B \in \mathsf{P}$ implies $A \in \mathsf{P}$. More generally, if $A, B_1, \dots, B_n$ are computational problems, then we write $A \leq_T^P \{B_1, \dots, B_n\}$ if $A \leq_T^P \bigcup_{i=1}^n (\{i\} \times B_i)$ (the set $\bigcup_{i=1}^n (\{i\} \times B_i)$ is basically the disjoint union of the B_i with every element from B_i marked by i).

HNN-extensions. Let $H = \langle \Sigma \mid R \rangle$ be a *base group* with isomorphic subgroups $A_i, B_i \leq H$ $(1 \leq i \leq n)$ and isomorphisms $\varphi_i : A_i \to B_i$. Let $h : (\Sigma^{\pm 1})^* \to H$ be the canonical morphism, which maps a word $w \in (\Sigma^{\pm 1})^*$ to the element of H it represents. We consider the *HNN-extension*

$$G = \langle H, t_1, \dots, t_n \mid a^{t_i} = \varphi_i(a) \ (1 \leq i \leq n, a \in A_i) \rangle. \tag{2}$$

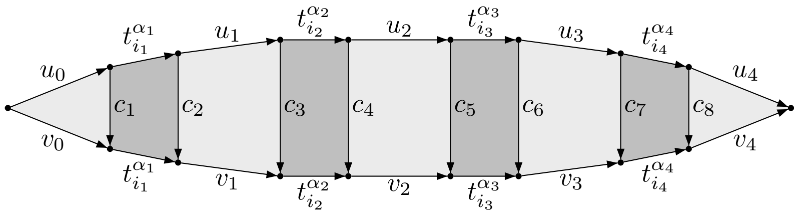
This means that $G = \langle \Sigma \cup \{t_1, \dots, t_n\} \mid R \cup \{a^{t_i} = \varphi_i(a) \mid 1 \leq i \leq n, a \in A_i\}\rangle$. It is known that the base group H naturally embeds into G [3]. In this paper, we will be only concerned with the case that all groups $A_1, \dots, A_n$ are finite and that Σ is finite. In this situation, we may assume that $\bigcup_{i=1}^{n}(A_i \cup B_i) \subseteq \Sigma$. We say that A_i and B_i are *associated subgroups* in the HNN-extension G. For the following, the notations $A_i(+1) = A_i$ and $A_i(-1) = B_i$ are useful. Note that $\varphi_i^{\alpha} : A_i(\alpha) \to A_i(-\alpha)$ for $\alpha \in \{+1, -1\}$.

A word $u \in (\Sigma^{\pm 1} \cup \{t_1, t_1^{-1}, \dots, t_n, t_n^{-1}\})^*$ is *reduced* if u does not contain a factor of the form $t_i^{-\alpha} w t_i^{\alpha}$ for $\alpha \in \{1, -1\}$, $w \in (\Sigma^{\pm 1})^*$ and $h(w) \in A_i(\alpha)$. With $\mathrm{Red}(H, \varphi_1, \dots, \varphi_n)$ we denote the set of all reduced words. For a word $u \in (\Sigma^{\pm 1} \cup \{t_1, t_1^{-1}, \dots, t_n, t_n^{-1}\})^*$ let us denote with $\pi_t(u)$ the projection of u to the alphabet $\{t_1, t_1^{-1}, \dots, t_n, t_n^{-1}\}$. The following Lemma provides a necessary and sufficient condition for equality of reduced strings in the group (2) [10]:

Lemma 2.1. *Let $u = u_0 t_{i_1}^{\alpha_1} u_1 \cdots t_{i_\ell}^{\alpha_\ell} u_\ell$ and $v = v_0 t_{j_1}^{\beta_1} v_1 \cdots t_{j_m}^{\beta_m} v_m$ be reduced words with $u_0, \dots, u_\ell, v_0, \dots, v_m \in (\Sigma^{\pm 1})^*$, $\alpha_1, \dots, \alpha_\ell, \beta_1, \dots, \beta_m \in \{1, -1\}$, and $i_1, \dots, i_\ell, j_1, \dots, j_m \in \{1, \dots, n\}$. Then $u = v$ in the HNN-extension G from (2) if and only if the following hold:*

(a) $\pi_t(u) = \pi_t(v)$ (i.e., $\ell = m$, $i_k = j_k$, and $\alpha_k = \beta_k$ for $1 \leq k \leq \ell$)
(b) there exist $c_1, \dots, c_{2m} \in \bigcup_{k=1}^{n}(A_k \cup B_k)$ such that:
  - *$u_k c_{2k+1} = c_{2k} v_k$ in H for $0 \leq k \leq \ell$ (here we set $c_0 = c_{2\ell+1} = 1$)*
  - *$c_{2k-1} \in A_{i_k}(\alpha_k)$ and $c_{2k} = \varphi_{i_k}^{\alpha_k}(c_{2k-1}) \in A_{i_k}(-\alpha_k)$ for $1 \leq k \leq \ell$.*

Condition (b) of the lemma can be visualized by a diagram of the following form (also called a Van Kampen diagram, see [11] for more details), where $\ell = m = 4$. Light-shaded (resp. dark-shaded) faces represent relations in H (resp. relations of the form $ct_i^{\alpha} = t_i^{\alpha}\varphi_i^{\alpha}(c)$ with $c \in A_i(\alpha)$).



Some simple compressed word problems. Plandowski [13] has shown that for two SLPs $\mathbb{A}$ and $\mathbb{B}$ it can be checked in polynomial time whether $\mathrm{val}(\mathbb{A}) = \mathrm{val}(\mathbb{B})$. In other words: the compressed word problem for a free monoid can be solved in polynomial time. In [7], this result was extended to free groups. A further generalization to free products $G_1 * G_2$ was shown in [8]:

Theorem 2.2. $\mathrm{CWP}(G_1 * G_2) \leq_T^P \{\mathrm{CWP}(G_1), \mathrm{CWP}(G_2)\}$.

For our reduction of the compressed word problem of an HNN-extension to the compressed word problem of the base group, we need the special case that in (2) we have $H = A_1 = \cdots = A_n = B_1 = \cdots = B_n$ (in particular, H is finite). In this case, we can even assume that the finite group H (represented by its multiplication table) is part of the input:

Lemma 2.3. *The following problem can be solved in polynomial time:*

INPUT: A finite group H, automorphisms $\varphi_i : H \to H$ $(1 \le i \le n)$, and an SLP $\mathbb{A}$ over the alphabet $H \cup \{t_1, t_1^{-1}, \dots t_n, t_n^{-1}\}$.
QUESTION: $\mathrm{val}(\mathbb{A}) = 1$ *in* $\langle H, t_1, \dots, t_n \mid h^{t_i} = \varphi_i(h)\ (1 \le i \le n, h \in H)\rangle$*?*

Proof. Let $s \in (H \cup \{t_1, t_1^{-1}, \dots t_n, t_n^{-1}\})^*$. From the defining equations of the group $G = \langle H, t_1, \dots, t_n \mid h^{t_i} = \varphi_i(h)\ (1 \le i \le n, h \in H)\rangle$ it follows that there exists a unique $h \in H$ with $s = \pi_t(s)h$ in G. Hence, $s = 1$ in G if and only if $\pi_t(s) = 1$ in the free group $F(t_1, \dots, t_n)$ and $h = 1$ in H.

Now, let $\mathbb{A}$ be an SLP over the alphabet $H \cup \{t_1, t_1^{-1}, \dots t_n, t_n^{-1}\}$. W.l.o.g. assume that $\mathbb{A}$ is in Chomsky normal form. It is straightforward to compute an SLP for the projection $\pi_t(\mathrm{val}(\mathbb{A}))$. Since by Theorem 2.2 the compressed word problem for the free group $F(t_1, \dots, t_n)$ can be solved in polynomial time, it suffices to compute for every nonterminal A of $\mathbb{A}$ the unique $h_A \in H$ with $\mathrm{val}(A) = \pi_t(\mathrm{val}(A))h_A$ in G. We compute the elements h_A bottom up. The case that the right-hand side for A is a terminal symbol from $H \cup \{t_1, t_1^{-1}, \dots t_n, t_n^{-1}\}$ is clear. Hence, assume that $A \to BC$ is a production of $\mathbb{A}$ and assume that $h_B, h_C \in H$ are already computed. Hence, in G we have $\mathrm{val}(A) = \mathrm{val}(B)\mathrm{val}(C) = \pi_t(\mathrm{val}(B))h_B\pi_t(\mathrm{val}(C))h_C$. Thus, it suffices to compute the unique $h \in H$ with $h_B\pi_t(\mathrm{val}(C)) = \pi_t(\mathrm{val}(C))h$ in G. Note that if $\pi_t(\mathrm{val}(C)) = t_{i_1}^{\alpha_1} t_{i_2}^{\alpha_2} \cdots t_{i_n}^{\alpha_n}$, then

$$h = \varphi_{i_n}^{\alpha_n}(\cdots \varphi_{i_2}^{\alpha_2}(\varphi_{i_1}^{\alpha_1}(h_B)) \cdots) = (\varphi_{i_1}^{\alpha_1} \circ \cdots \circ \varphi_{i_n}^{\alpha_n})(h_B).$$

The automorphism $f = \varphi_{i_1}^{\alpha_1} \circ \cdots \circ \varphi_{i_n}^{\alpha_n}$ can be easily computed from an SLP $\mathbb{C}$ for the string $\pi_t(\mathrm{val}(C))$ by replacing in $\mathbb{C}$ the terminal symbol t_i (resp. t_i^{-1}) by φ_i (resp. φ_i^{-1}). This allows to compute f bottom-up and then to compute $f(h_B)$. □

Note that the group $\langle H, t_1, \dots, t_n \mid h^{t_i} = \varphi_i(h)\ (1 \le i \le n, h \in H)\rangle$ is the semidirect product $H \rtimes_\varphi F$, where $F = F(t_1, \dots, t_n)$ is the free group generated by $t_1, \dots, t_n$ and the homomorphism $\varphi : F \to \mathrm{Aut}(H)$ is defined by $\varphi(t_i) = \varphi_i$.

3 Compressed Word Problem of an HNN-Extension

In this section we show that the compressed word problem for an HNN-extension of the form (1) is polynomial time Turing-reducible to the compressed word problem for H. In fact, we prove the existence of such a reduction for a slightly more general problem, which we introduce below.

For the further consideration, let us fix the group H together with the finite subgroups A and B. Let Σ be a finite generating set for H. These data are fixed, i.e., they will not belong to the input of computational problems. In the following, when writing down a multiple HNN-extension

$$\langle H, t_1, \dots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \le i \le n, a \in A)\rangle, \tag{3}$$

we allow implicitly that every φ_i is only partially defined on A. Thus, (3) is in fact an abbreviation for $\langle H, t_1, \dots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \le i \le n, a \in \mathrm{dom}(\varphi_i))\rangle$.

Note that there is only a fixed number of partial isomorphisms from A to B, but we allow $\varphi_i = \varphi_j$ for $i \neq j$ in (3).

Let us introduce several restrictions and extensions of CWP(G). Our most general problem is the following computational problem UCWP(H, A, B) (the letter "U" stands for "uniform", meaning that a list of partial isomorphisms from A to B is part of the input):

INPUT: Partial isomorphisms $\varphi_i : A \to B$ $(1 \leq i \leq n)$ and an SLP $\mathbb{A}$ over the alphabet $\Sigma^{\pm 1} \cup \{t_1, t_1^{-1}, \ldots, t_n, t_n^{-1}\}$.
QUESTION: $\mathrm{val}(\mathbb{A}) = 1$ in $\langle H, t_1, \ldots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \leq i \leq n, a \in A)\rangle$?

The restriction of this problem UCWP(H, A, B) to reduced input strings is denoted by RUCWP(H, A, B). It is formally defined as the following problem:

INPUT: Partial isomorphisms $\varphi_i : A \to B$ $(1 \leq i \leq n)$ and SLPs $\mathbb{A}, \mathbb{B}$ over the alphabet $\Sigma^{\pm 1} \cup \{t_1, t_1^{-1}, \ldots, t_n, t_n^{-1}\}$ such that $\mathrm{val}(\mathbb{A}), \mathrm{val}(\mathbb{B}) \in \mathrm{Red}(H, \varphi_1, \ldots, \varphi_n)$.
QUESTION: $\mathrm{val}(\mathbb{A}) = \mathrm{val}(\mathbb{B})$ in $\langle H, t_1, \ldots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \leq i \leq n, a \in A)\rangle$?

Let us now consider a fixed list of partial isomorphisms $\varphi_1, \ldots, \varphi_n : A \to B$. Then RCWP($H, A, B, \varphi_1, \ldots, \varphi_n$) is the following computational problem:

INPUT: Two SLPs $\mathbb{A}$ and $\mathbb{B}$ over the alphabet $\Sigma^{\pm 1} \cup \{t_1, t_1^{-1}, \ldots, t_n, t_n^{-1}\}$ such that $\mathrm{val}(\mathbb{A}), \mathrm{val}(\mathbb{B}) \in \mathrm{Red}(H, \varphi_1, \ldots, \varphi_n)$.
QUESTION: $\mathrm{val}(\mathbb{A}) = \mathrm{val}(\mathbb{B})$ in $\langle H, t_1, \ldots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \leq i \leq n, a \in A)\rangle$?

Our main result is:

Theorem 3.1. UCWP(H, A, B) $\leq_P^T$ CWP(H).

The rest of Section 3 sketches the main steps of our proof of Theorem 3.1. First, we state that we may restrict ourselves to SLPs that evaluate to reduced strings:

Lemma 3.2. UCWP(H, A, B) $\leq_P^T$ RUCWP(H, A, B). *More precisely, there is a polynomial time Turing-reduction from* UCWP(H, A, B) *to* RUCWP(H, A, B) *that on input* $(\varphi_1, \ldots, \varphi_n, \mathbb{A})$ *only asks* RUCWP(H, A, B)*-queries of the form* $(\varphi_1, \ldots, \varphi_n, \mathbb{A}', \mathbb{B}')$ *(thus, the list of partial isomorphisms is not changed).*

Lemma 3.3. *Let* $\varphi_1, \ldots, \varphi_n : A \to B$ *be fixed partial isomorphisms. Then* CWP($\langle H, t_1, \ldots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \leq i \leq n, a \in A)\rangle$) *is polynomial time Turing-reducible to* RCWP($H, A, B, \varphi_1, \ldots, \varphi_n$).

Roughly, the strategy in the proofs of Lemma 3.2 and Lemma 3.3 is to transform bottom up the rules of the input SLP $\mathbb{A}$ into new rules (for the same set of nonterminals) such that the new rules generate from a nonterminal A a reduced string, which is equivalent in the HNN-extension to the string that is generated from A in the original SLP $\mathbb{A}$. The same technique was applied in [7] in order to compute an SLP for irreducible normal form in free group for a given SLP-compressed string.

In a second step we show that the number of different stable letters can be reduced to a constant. For this, it is important to note that the associated subgroups $A, B \leq H$ do not belong to the input; so their size is a fixed constant.

Fix the constant $\delta = 2 \cdot |A|! \cdot 2^{|A|}$. Note that the number of HNN-extensions of the form $\langle H, t_1, \dots, t_k \mid a^{t_i} = \psi_i(a)\ (1 \le i \le k, a \in A)\rangle$ with $k \le \delta$ is constant. The following lemma says that $\mathrm{RUCWP}(H, A, B)$ can be reduced in polynomial time to one of the problems $\mathrm{RCWP}(H, A, B, \psi_1, \dots, \psi_k)$. Moreover, we can determine in polynomial time, which of these problems arises.

Lemma 3.4. *There exists a polynomial time algorithm for the following:*

INPUT: Partial isomorphisms $\varphi_1, \dots, \varphi_n : A \to B$ and SLPs $\mathbb{A}, \mathbb{B}$ over the alphabet $\Sigma^{\pm 1} \cup \{t_1, t_1^{-1}, \dots, t_n, t_n^{-1}\}$ such that $\mathrm{val}(\mathbb{A}), \mathrm{val}(\mathbb{B}) \in \mathrm{Red}(H, \varphi_1, \dots, \varphi_n)$.

OUTPUT: Partial isomorphisms $\psi_1, \dots, \psi_k : A \to B$ where $k \le \delta$ and SLPs $\mathbb{A}'$, $\mathbb{B}'$ over the alphabet $\Sigma^{\pm 1} \cup \{t_1, t_1^{-1}, \dots, t_k, t_k^{-1}\}$ such that:

- *For every $1 \le i \le k$ there exists $1 \le j \le n$ with $\psi_i = \varphi_j$.*
- $\mathrm{val}(\mathbb{A}'), \mathrm{val}(\mathbb{B}') \in \mathrm{Red}(H, \psi_1, \dots, \psi_k)$
- $\mathrm{val}(\mathbb{A}) = \mathrm{val}(\mathbb{B})$ *in $\langle H, t_1, \dots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \le i \le n, a \in A)\rangle$ if and only if* $\mathrm{val}(\mathbb{A}') = \mathrm{val}(\mathbb{B}')$ *in $\langle H, t_1, \dots, t_k \mid a^{t_i} = \psi_i(a)\ (1 \le i \le k, a \in A)\rangle$.*

Proof. Fix an input $(\varphi_1, \dots, \varphi_n, \mathbb{A}, \mathbb{B})$ for the problem $\mathrm{RUCWP}(H, A, B)$. In particular, $\mathrm{val}(\mathbb{A}), \mathrm{val}(\mathbb{B}) \in \mathrm{Red}(H, \varphi_1, \dots, \varphi_n)$. Define the function $\tau : \{1, \dots, n\} \to \{1, \dots, n\}$ by

$$\tau(i) = \min\{k \mid \varphi_k = \varphi_i\}.$$

This mapping can be easily computed in polynomial time from the sequence $\varphi_1, \dots, \varphi_n$. Assume w.l.o.g. that $\mathrm{ran}(\tau) = \{1, \dots, \gamma\}$ for some $\gamma \le n$. Note that $\gamma \le |A|! \cdot 2^{|A|} = \frac{\delta}{2}$. For every t_i $(1 \le i \le \gamma)$ we take two stable letters $t_{i,0}$ and $t_{i,1}$. Hence, the total number of stable letters is at most δ. Moreover, we define a sequential transducer T which, reading as input the word $u_0 t_{i_1}^{\alpha_1} u_1 \cdots t_{i_m}^{\alpha_m} u_m$ (with $u_0, \dots, u_m \in (\Sigma^{\pm 1})^+$ and $1 \le i_1, \dots, i_m \le n$) returns

$$T(w) = u_0\, t_{\tau(i_1),1}^{\alpha_1}\, u_1\, t_{\tau(i_2),0}^{\alpha_2}\, u_2\, t_{\tau(i_3),1}^{\alpha_3}\, u_3 \cdots t_{\tau(i_m),m \bmod 2}^{\alpha_m}\, u_m.$$

Finally, we define the HNN-extension

$$G' = \langle H, t_{1,0}, t_{1,1}, \dots, t_{\gamma,0}, t_{\gamma,1} \mid a^{t_{i,k}} = \varphi_i(a)\ (1 \le i \le \gamma, k \in \{0,1\}, a \in A)\rangle.$$

This HNN-extension has $2\gamma \le \delta$ many stable letters; it is the HNN-extension $\langle H, t_1, \dots, t_k \mid a^{t_i} = \psi_i(a)\ (1 \le i \le k, a \in A)\rangle$ from the lemma.

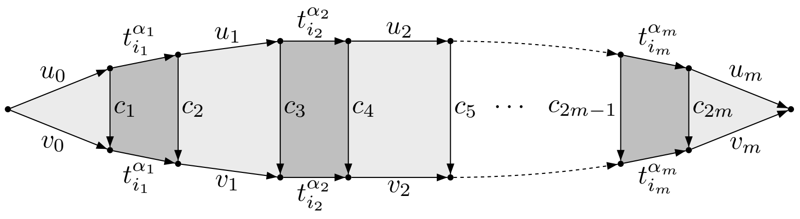
Claim. Let $u, v \in \mathrm{Red}(H, \varphi_1, \dots, \varphi_n)$ be reduced. Then also $T(u)$ and $T(v)$ are reduced. Moreover, the following are equivalent:

(a) $u = v$ in $\langle H, t_1, \dots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \le i \le n, a \in A)\rangle$
(b) $T(u) = T(v)$ in the HNN-extension G' and $\pi_t(u) = \pi_t(v)$.

Proof of the claim. Let $u = u_0 t_{i_1}^{\alpha_1} u_1 \cdots t_{i_\ell}^{\alpha_\ell} u_\ell$ and $v = v_0 t_{j_1}^{\beta_1} v_1 \cdots t_{j_m}^{\beta_m} v_m$. The first statement is obvious due to the fact that $T(u)$ does not contain a subword of the form $t_{i,k}^{\alpha} w t_{j,k}^{\beta}$ for $k \in \{0,1\}$, and similarly for $T(v)$.

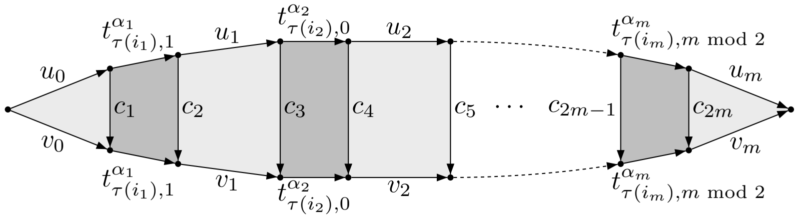
For $(a) \Rightarrow (b)$ note that by Lemma 2.1, $u = v$ in $\langle H, t_1, \dots, t_n \mid a^{t_i} = \varphi_i(a)\ (1 \le i \le n, a \in A)\rangle$ implies $\pi_t(u) = \pi_t(v)$ (i.e. $\ell = m$, $\alpha_1 = \beta_1, \dots, \alpha_m =$

β_m, $i_1 = j_1, \ldots, i_m = j_m$), and that there exists a Van Kampen diagram of the following form:



(†)

The defining equations of G' imply that the following is a valid Van Kampen diagram in G':


$t_{\tau(i_1),1}^{\alpha_1}$
u_1
$t_{\tau(i_2),0}^{\alpha_2}$
u_2
$t_{\tau(i_m),m \bmod 2}^{\alpha_m}$
u_0
u_m
c_1
c_2
c_3
c_4
c_5
$\cdots$
c_{2m-1}
c_{2m}
v_0
v_m
$t_{\tau(i_1),1}^{\alpha_1}$
v_1
$t_{\tau(i_2),0}^{\alpha_2}$
v_2
$t_{\tau(i_m),m \bmod 2}^{\alpha_m}$
(‡)

Hence, $T(u) = T(v)$ in G'.

For $(b) \Rightarrow (a)$ note that we have already seen that $T(u)$ and $T(v)$ are reduced. Hence, $T(u) = T(v)$ in G' together with $\pi_t(u) = \pi_t(v)$ implies that there exists a Van Kampen diagram of the form (‡). Again, we can replace the dark-shaded t-faces by the corresponding t-faces of G in order to obtain a diagram of the form (†). This proofs the claim.

By the previous claim, $T(\mathrm{val}(\mathbb{A}))$ and $T(\mathrm{val}(\mathbb{B}))$ are reduced. Moreover, by [1], SLPs $\mathbb{A}'$ and $\mathbb{B}'$ for these strings can be computed in polynomial time from the SLPS $\mathbb{A}$ and $\mathbb{B}$, respectively. In case $\pi_t(\mathrm{val}(\mathbb{A})) \neq \pi_t(\mathrm{val}(\mathbb{B}))$ we choose these SLPs such that e.g. $\mathrm{val}(\mathbb{A}') = t_1$ and $\mathrm{val}(\mathbb{B}') = t_1^{-1}$. Hence, $\mathrm{val}(\mathbb{A}') = \mathrm{val}(\mathbb{B}')$ in G' if and only if $\mathrm{val}(\mathbb{A}) = \mathrm{val}(\mathbb{B})$ in $\langle H, t_1, \ldots, t_n \mid a^{t_i} = \varphi_i(a)(1 \leq i \leq n, a \in A)\rangle$. This proves the lemma. □

Due to Lemma 3.4 it suffices to concentrate our effort on problems of the form RCWP$(H, A, B, \varphi_1, \ldots, \varphi_k)$, where $k \leq \delta$. We have to check whether for two given SLP-compressed reduced strings u and v conditions (a) and (b) from Lemma 2.1 are satisfied. Condition (a) can be easily checked by computing SLPs for $\pi_t(u)$ and $\pi_t(v)$ and then checking for equality using Plandowski's algorithm [13]. The whole difficulty lies in checking condition (b) from Lemma 2.1. Let

$$G_0 = \langle H, t_1, \ldots, t_k \mid a^{t_i} = \varphi_i(a) \ (1 \leq i \leq k, a \in A)\rangle \tag{4}$$

and let us choose $i \in \{1, \ldots, k\}$ such that $|\mathrm{dom}(\varphi_i)|$ is maximal. W.l.o.g. assume that $i = 1$. Let $\mathrm{dom}(\varphi_1) = A_1 \leq A$ and $\mathrm{ran}(\varphi_1) = B_1 \leq B$. We write t for t_1 in the following and define $\Gamma = \Sigma \cup \{t_2, \ldots, t_k\}$. We can write our HNN-extension G_0 from (4) as

$$G_0 = \langle K, t \mid a^t = \varphi_1(a) \ (a \in A_1)\rangle, \text{ where} \tag{5}$$
$$K = \langle H, t_2, \ldots, t_k \mid a^{t_i} = \varphi_i(a) \ (2 \le i \le k, a \in A)\rangle. \tag{6}$$

The latter group K is generated by Γ. The main reduction step in our algorithm is expressed in the following lemma:

Lemma 3.5. $\mathrm{RCWP}(H, A, B, \varphi_1, \ldots, \varphi_k)$ *is polynomial time Turing-reducible to the problems* $\mathrm{RCWP}(H, A, B, \varphi_2, \ldots, \varphi_k)$ *and* $\mathrm{RUCWP}(A_1, A_1, A_1)$.

Let us briefly sketch the proof of Lemma 3.5: Let $(\mathbb{A}, \mathbb{B})$ be an input for the problem $\mathrm{RCWP}(H, A, B, \varphi_1, \ldots, \varphi_k)$ with $k \le \delta$. Thus, $\mathbb{A}$ and $\mathbb{B}$ are SLPs over the alphabet $\Sigma^{\pm 1} \cup \{t_1, t_1^{-1}, \ldots, t_k, t_k^{-1}\} = \Gamma^{\pm 1} \cup \{t, t^{-1}\}$ with $\mathrm{val}(\mathbb{A}), \mathrm{val}(\mathbb{B}) \in \mathrm{Red}(H, \varphi_1, \ldots, \varphi_k)$. Hence, we also have $\mathrm{val}(\mathbb{A}), \mathrm{val}(\mathbb{B}) \in \mathrm{Red}(K, \varphi_1)$. W.l.o.g. we may assume that $\pi_t(\mathrm{val}(\mathbb{A})) = \pi_t(\mathrm{val}(\mathbb{B}))$. This property can be checked in polynomial time using Plandowski's algorithm [13], and if it is not satisfied then we have $\mathrm{val}(\mathbb{A}) \ne \mathrm{val}(\mathbb{B})$ in G_0.

In a first step, we modify the SLPs $\mathbb{A}$ and $\mathbb{B}$ in such a way that in a first step they generate strings of the form $X_0 t^{\alpha_1} X_1 \cdots t^{\alpha_m} X_m$ and $Y_0 t^{\alpha_1} Y_1 \cdots t^{\alpha_m} Y_m$, respectively. Here the X_i and Y_j are nonterminals that generate in a second phase strings over the alphabet $\Gamma^{\pm 1}$. This is possible in polynomial time but requires some work. Then, we transform our $\mathrm{RCWP}(H, A, B, \varphi_1, \ldots, \varphi_k)$-instance $(\mathbb{A}, \mathbb{B})$ into a compressed word problem for a new group G_1 that is generated by the stable letter t and the symbols $X_1, \ldots, X_m, Y_1, \ldots, Y_m$. Here, the idea is to abstract as far as possible from the concrete structure of the original base group K. In some sense, we only keep those K-relations that are necessary to prove (or disprove) that $\mathrm{val}(\mathbb{A}) = \mathrm{val}(\mathbb{B})$ in the group G_0. These K-relations are translated into relations on the "generic" symbols $X_1, \ldots, X_m, Y_1, \ldots, Y_m$. In order to compute these relations, we need oracle access to $\mathrm{CWP}(K)$ or alternatively (by Lemma 3.3) to $\mathrm{RCWP}(H, A, B, \varphi_2, \ldots, \varphi_k)$. Using Tietze transformations [11], our new group G_1 is finally transformed into an HNN-extension with base group A_1 — this gives us the $\mathrm{RUCWP}(A_1, A_1, A_1)$-instance in Lemma 3.5.

We now apply Lemma 3.4 to the problem $\mathrm{RUCWP}(A_1, A_1, A_1)$ (one of the two target problems in Lemma 3.5). An input for this problem can be reduced in polynomial time to an instance of a problem $\mathrm{RCWP}(A_1, A_1, A_1, \psi_1, \ldots, \psi_k)$, where $\psi_1, \ldots, \psi_k : A_1 \to A_1$ are partial automorphisms and $k \le \delta$ (we have $k \le 2|A_1|! \cdot 2^{|A_1|} \le 2|A|! \cdot 2^{|A|} = \delta$). Hence, we are faced with an HNN-extension of the form $G_2 = \langle A_1, t_1, \ldots, t_k \mid a^{t_i} = \psi_i(a) \, (1 \le i \le k, a \in \mathrm{dom}(\psi_i))\rangle$. Next, we separate the (constantly many) stable letters $t_1, \ldots, t_k$ that occur in the $\mathrm{RCWP}(A_1, A_1, A_1, \psi_1, \ldots, \psi_k)$-instance into two sets: $\{t_1, \ldots, t_k\} = S_1 \cup S_2$ where $S_1 = \{t_i \mid \mathrm{dom}(\psi_i) = A_1\}$ and $S_2 = \{t_1, \ldots, t_k\} \setminus S_1$. W.l.o.g. assume that $S_2 = \{t_1, \ldots, t_\ell\}$. Then we can write our HNN-extension G_2 as

$$G_2 = \langle H', t_1, \ldots, t_\ell \mid a^{t_i} = \psi_i(a) \ (1 \le i \le \ell, a \in \mathrm{dom}(\psi_i))\rangle, \tag{7}$$

where $H' = \langle A_1, t_{\ell+1}, \ldots, t_k \mid a^{t_i} = \psi_i(a) \ (\ell + 1 \le i \le k, a \in A_1)\rangle$. Note that $|\mathrm{dom}(\psi_i)| < |A_1|$ for every $1 \le i \le \ell$ and that $A_1 = \mathrm{dom}(\psi_i)$ for every $\ell + 1 \le i \le k$. By Lemma 2.3, $\mathrm{CWP}(H')$ can be solved in polynomial time; H' is in fact the semidirect product $A_1 \rtimes_\varphi F(t_{\ell+1},\dots,t_k)$, where $\varphi : F(t_{\ell+1},\dots,t_k) \to \mathrm{Aut}(A_1)$ is defined by $\varphi(t_i) = \psi_i$. Recall also that A_1 was chosen to be of maximal cardinality among the domains of all partial isomorphisms $\varphi_1,\dots,\varphi_k$. The following proposition summarizes what we have shown so far:

Proposition 3.6. *Let $\varphi_1,\dots,\varphi_k : A \to B$ be partial isomorphisms, where $k \leq \delta$, $A_1 = \mathrm{dom}(\varphi_1)$, and w.l.o.g $|A_1| \geq |\mathrm{dom}(\varphi_i)|$ for $1 \leq i \leq k$. From an instance $(\mathbb{A},\mathbb{B})$ of the problem $\mathrm{RCWP}(H,A,B,\varphi_1,\dots,\varphi_k)$ we can compute in polynomial time with oracle access to the problem $\mathrm{RCWP}(H,A,B,\varphi_2,\dots,\varphi_k)$*

(1) a semidirect product $A_1 \rtimes_\varphi F$, where F is a free group of rank at most δ,
(2) partial automorphisms $\psi_1,\dots,\psi_\ell : A_1 \to A_1$ with $\ell \leq \delta$ and $|\mathrm{dom}(\psi_i)| < |A_1|$ for all $1 \leq i \leq \ell$, and
(3) an $\mathrm{RCWP}(A_1 \rtimes_\varphi F, A_1, A_1, \psi_1,\dots,\psi_\ell)$-instance, which is positive if and only if the initial $\mathrm{RCWP}(H,A,B,\varphi_1,\dots,\varphi_k)$-instance $(\mathbb{A},\mathbb{B})$ is positive.

Note that in (1) there are only constantly many semidirect products of the form $A_1 \rtimes_\varphi F$ and that $\mathrm{CWP}(A_1 \rtimes_\varphi F)$ can be solved in polynomial time by Lemma 2.3. We are now ready to prove the main theorem of this paper.

Proof of Theorem 3.1. By Lemma 3.2 and Lemma 3.4 it suffices to solve a problem $\mathrm{RCWP}(H,A,B,\varphi_1,\dots,\varphi_\delta)$ in polynomial time. For this we apply Proposition 3.6 repeatedly. We obtain a computation tree, where the root is labeled with an $\mathrm{RCWP}(H,A,B,\varphi_1,\dots,\varphi_\delta)$-instance and every other node is labeled with an instance of a problem $\mathrm{RCWP}(C \rtimes_\varphi F, C, C, \theta_1,\dots,\theta_p)$, where F is a free group of rank at most δ, C is a subgroup of our finite group A, and $p \leq \delta$. The number of these problems is bounded by some fixed constant. Since along each edge in the tree, either the number of stable letters reduces by one, or the maximal size of an associated subgroup becomes strictly smaller, the height of the tree is bounded by a constant (it is at most $|A| \cdot \delta = 2 \cdot |A| \cdot |A|! \cdot 2^{|A|}$). Moreover, along each tree edge, the size of a problem instance can grow only polynomially. Hence, each problem instance that appears in the computation tree has polynomial size w.r.t. the input size. Hence, the total running time is bounded polynomially. □

4 Amalgamated Free Products

Let H_1 and H_2 be two finitely generated groups. Let $A_1 \leq H_1$ and $A_2 \leq H_2$ be finite and $\varphi : A_1 \mapsto A_2$ an isomorphism. The *amalgamated free product of H_1 and H_2, amalgamating the subgroups A_1 and A_2 by the isomorphism φ*, is the group $G = \langle H_1 * H_2 \mid a = \varphi(a)\ (a \in A_1)\rangle$.

Theorem 4.1. *Let $G = \langle H_1 * H_2 \mid a = \varphi(a)\ (a \in A_1)\rangle$ be an amalgamated free product with A_1 finite. Then $\mathrm{CWP}(G) \leq_T^P \{\mathrm{CWP}(H_1), \mathrm{CWP}(H_2)\}$.*

Proof. It is well known [11, Theorem 2.6, p. 187] that G can be embedded into the HNN-extension $G' = \langle H_1 * H_2, t \mid a^t = \varphi(a)\ (a \in A_1)\rangle$ by the homomorphism Φ with $\Phi(x) = t^{-1}xt$ for $x \in H_1$ and $\Phi(x) = x$ for $x \in H_2$. Given an SLP $\mathbb{A}$ we can easily compute an SLP $\mathbb{B}$ with $\mathrm{val}(\mathbb{B}) = \Phi(\mathrm{val}(\mathbb{A}))$. We obtain: $\mathrm{val}(\mathbb{A}) = 1$ in $G \iff \Phi(\mathrm{val}(\mathbb{A})) = 1$ in $\Phi(G) \iff \mathrm{val}(\mathbb{B}) = 1$ in G'. By Theorem 3.1 and Theorem 2.2, $\mathrm{CWP}(G')$ can be solved in polynomial time with oracle access to $\mathrm{CWP}(H_1)$ and $\mathrm{CWP}(H_2)$. $\square$

5 Open Problems

We have shown that the compressed word problem for an HNN-extension with finite associated subgroups is polynomial time Turing-reducible to the compressed word problem for the base group. Here, the base group and the associated subgroups are fixed, i.e. are not part of the input. One might also consider the *uniform* compressed word problem for HNN-extensions of the form $\langle H, t \mid a^t = \varphi(a)\ (a \in A)\rangle$, where H is a finite group that is part of the input. It is not clear, whether this problem can be solved in polynomial time. Finally, one might also consider the compressed word problem for HNN-extensions of semigroups [5].

References

1. Bertoni, A., Choffrut, C., Radicioni, R.: Literal shuffle of compressed words. In: Proceeding of the 5th IFIP International Conference on Theoretical Computer Science (IFIP TCS 2008), Milano (Italy), pp. 87–100. Springer, Heidelberg (2008)
2. Dicks, W., Dunwoody, M.J.: Groups Acting on Graphs. Cambridge University Press, Cambridge (1989)
3. Higman, G., Neumann, B.H., Neumann, H.: Embedding theorems for groups. Journal of the London Mathematical Society. Second Series 24, 247–254 (1949)
4. Haubold, N., Lohrey, M.: Compressed word problems in HNN-extensions and amalgamated products. arXiv.org (2008), `http://arxiv.org/abs/0811.3303`
5. Howie, J.M.: Embedding theorems for semigroups. Quart. J. Math. Oxford Ser. (2) 14, 254–258 (1963)
6. Kapovich, I., Myasnikov, A., Schupp, P., Shpilrain, V.: Generic-case complexity, decision problems in group theory, and random walks. J. Algebra 264(2), 665–694 (2003)
7. Lohrey, M.: Word problems and membership problems on compressed words. SIAM J. Comput. 35(5), 1210–1240 (2006)
8. Lohrey, M., Schleimer, S.: Efficient computation in groups via compression. In: Diekert, V., Volkov, M.V., Voronkov, A. (eds.) CSR 2007. LNCS, vol. 4649, pp. 249–258. Springer, Heidelberg (2007)
9. Lohrey, M., Sénizergues, G.: Theories of HNN-extensions and amalgamated products. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 504–515. Springer, Heidelberg (2006)
10. Lohrey, M., Sénizergues, G.: Rational subsets in HNN-extensions and amalgamated products. Internat. J. Algebra Comput. 18(1), 111–163 (2008)
11. Lyndon, R.C., Schupp, P.E.: Combinatorial Group Theory. Springer, Heidelberg (1977)
12. Myasnikov, A., Shpilrain, V., Ushakov, A.: Group-based Cryptography. Birkhäuser, Basel (2008)

13. Plandowski, W.: Testing equivalence of morphisms on context-free languages. In: van Leeuwen, J. (ed.) ESA 1994. LNCS, vol. 855, pp. 460–470. Springer, Heidelberg (1994)
14. Plandowski, W., Rytter, W.: Complexity of language recognition problems for compressed words. In: Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa, pp. 262–272. Springer, Heidelberg (1999)
15. Schleimer, S.: Polynomial-time word problems. Comment. Math. Helv. 83(4), 741–765 (2008)
16. Stallings, J.R.: Group Theory and Three-Dimensional Manifolds. Yale Mathematical Monographs, No. 4. Yale University Press (1971)

Variations on Muchnik's Conditional Complexity Theorem*

Daniil Musatov[1], Andrei Romashchenko[2], and Alexander Shen[3]

[1] Lomonosov Moscow State University
musatych@gmail.com
[2] LIF Marseille, CNRS & Univ. Aix–Marseille, on leave from IITP RAS
andrei.romashchenko@gmail.com
[3] LIF Marseille, CNRS & Univ. Aix–Marseille, on leave from IITP RAS
alexander.shen@lif.univ-mrs.fr

Abstract. Muchnik's theorem about simple conditional descriptions states that for all strings a and b there exists a short program p transforming a to b that has the least possible length and is simple conditional on b. In this paper we present two new proofs of this theorem. The first one is based on the on-line matching algorithm for bipartite graphs. The second one, based on extractors, can be generalized to prove a version of Muchnik's theorem for space-bounded Kolmogorov complexity.

1 Muchnik's Theorem

An. Muchnik [8] has proven the following theorem:

Theorem 1. *Let a and b be two binary strings, $C(a) < n$ and $C(a|b) < k$. Then there exists a string p such that*

- $C(a|p,b) = O(\log n)$;
- $C(p) \leq k + O(\log n)$;
- $C(p|a) = O(\log n)$.

Here $C(u)$ stands for Kolmogorov complexity of string u (the length of a shortest program generating u); conditional complexity of u given v (the length of a shortest program that translates v to u) is denoted by $C(u|v)$, see [5]. The constants hidden in $O(\log n)$ do not depend on n, k, a, b, p.

Informally, this theorem says that there exists a program p that transforms b to a, has the minimal possible complexity $C(a|b)$ (up to a logarithmic term) and, moreover, can be easily obtained from a. (The last requirement is crucial, otherwise the statement becomes a trivial reformulation of the definition of conditional Kolmogorov complexity.)

This theorem is an algorithmic counterpart of Slepian–Wolf theorem [12] in multisource information theory. Assume that somebody (**S**) knows b and wants

* Supported by ANR Sycomore, NAFIT ANR-08-EMER-008-01 and RFBR 09-01-00709-a grants.

to know a. We know a and want to send some message p to $\mathbf{S}$ that will allow $\mathbf{S}$ to reconstruct a. How long should be this message? Do we need to know b to be able to find such a message? Muchnik's theorem provides a negative answer to the last question (in a sense: we still need a logarithmic advice). Indeed, the absolute minimum for a complexity of a piece of information p that together with b allows $\mathbf{S}$ to reconstruct a, is $C(a|b)$. It is easy to see that this minimum can be achieved (with logarithmic precision) by a string p that has logarithmic complexity conditional to a and b. But it turns out that in fact b is not needed and we can provide p that is simple conditional to a and still does the job.

In many cases statements about Kolmogorov complexity have combinatorial counterparts (and sometimes it is easy to show the equivalence between complexity and combinatorial statements). In the present paper we investigate two different combinatorial objects closely related to Muchnik's theorem and its proof.

First (Sect. 2), we define the *on-line matching* problem for bipartite graphs. We formulate some combinatorial statement about on-line matchings. This statement (1) easily implies Muchnik's theorem and (2) can be proven using the same ideas (slightly modified) that were used by Muchnik in his original proof.

Second (Sect. 3), following [4], we use expanders and their combinatorial properties. Based on this technique, we give a new proof of Muchnik's theorem. With this method we prove a version of this theorem for polynomial space Kolmogorov complexity and also for some very special version of polynomial time Kolmogorov complexity.

Due to the lack of space some technical details are omitted. Complete proofs may be found in the full version of this paper [1].

2 Muchnik's Theorem and On-Line Matchings

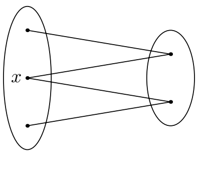
2.1 On-Line Matchings

Consider a bipartite graph with the left part L, the right part R and a set of edges $E \subset L \times R$. Let s be some integer. We are interested in the following property of the graph:

> *for any subset L' of L of size at most s there exists a subset $E' \subset E$ that performs a bijection between L' and some $R' \subset R$.*

A necessary and sufficient condition for this property is provided by the well known Hall theorem: *For each set $L' \subset L$ of size $t \leq s$ the set of all neighbors of elements of L' contains at least t elements.*

This condition is not sufficient for the following on-line version of matching. We assume that an adversary gives us elements of the left part L (up to s elements) one by one. At each step we should provide a counterpart for each given element x, i.e., to choose some neighbor $y \in R$ not used before. (This choice is final and cannot be changed later.)



Providing a matching on-line (when next steps of the adversary are not known in advance) is a more subtle problem than the usual off-line matching. Now the Hall criterion cannot be used as a sufficient condition. For example, for the graph shown on the picture, one can find a matching for each subset of size at most 2 of the left part, but this cannot be done on-line (we are blocked if the adversary starts with x).

Now we formulate a combinatorial statement about on-line matching and then show that this property implies Muchnik's theorem (Sect. 2.2) and prove this property (Sect. 2.3).

Combinatorial statement about on-line matchings (**OM**). *There exists a constant c such that for every integers n and $k \leq n$ there exists a bipartite graph $E \subset L \times R$ whose left part L has size 2^n, right part R has size $2^k n^c$, each vertex in L has at most n^c neighbors in R, and on-line matching is possible up to size 2^k.*

(So the size of the on-line matching is close to the size of R up to a polynomial factor, and the degrees of all L-elements are polynomially bounded.)

2.2 Proof of Muchnik's Theorem

First we show how (OM) implies Muchnik's theorem. We may assume without loss of generality that the length of the string a (instead of its complexity) is less than n. Indeed, if we replace a by a shortest program that generates a, all complexities involving a change by only $O(\log n)$ term: knowing the shortest program for a, we can get a without any additional information, and to get a shortest program for a given a we need only to know $C(a)$ (try all programs of length $C(a)$ until one of them produces a). There may exist several different shortest programs for the word a; we take that one which appears first in exhaustive search over all programs of length $C(a)$.

Consider the graph E provided by (OM) with parameters n and k. Its left part L is interpreted as the set of all strings of length less than n; therefore, a is an element of L. Let us enumerate all strings x of length less than n such that $C(x|b) < k$. There exist at most 2^k such strings, and a is one of them. The property (OM) implies that it is possible to find an on-line matching for all these strings (in the order they appear during the enumeration). Let p be an element of R that corresponds to a in this matching.

Let us check that p satisfies all the conditions of Muchnik's theorem. First of all, note that the graph E can be chosen in such a way that its complexity is $O(\log n)$. Indeed, (OM) guarantees that a graph with the required properties exist. Given n and k, we can perform an exhaustive search until the first graph with these properties is found. This graph is a computable function of n and k, so its complexity does not exceed $C(n, k) = O(\log n)$.

If a is given (as well as n and k), then p can be specified by its ordinal number in the list of a-neighbors. This list contains at most n^c elements, so the ordinal number contains $O(\log n)$ bits.

To specify p without knowing a, we give the ordinal number of p in R, which is $k + O(\log n)$ bits long. (Again n and k are used, but this is another $O(\log n)$ bits.)

To reconstruct a from b and p, we run the enumerating of all strings of lengths less than n that have conditional complexity (relative to b, which is known) less than k, and find R-counterparts for them (using (OM)) until p appears. Then a is the L-counterpart of p in this matching.

Formally speaking, for given n and k we should fix not only a graph G but also some on-line matching procedure, and use the same procedure both for constructing p and for reconstructing a from b and p. □

2.3 On-Line Matchings Exist

It remains to prove the statement (OM). Our proof follows the original Muchnik's argument adapted for the combinatorial setting.

First, let us prove a weaker statement when on-line matchings are replaced by off-line matchings (in this case the statement can be reformulated using Hall's criterion).

Off-line version of (OM). *There exists a constant c such that for any integers n and $k \le n$ there exists a bipartite graph $E \subset L \times R$ whose left part L is of size 2^n, the right part R is of size $2^k n^c$, each vertex in L has at most n^c neighbors in R and for any subset $X \subset L$ of size $t \le 2^k$ the set $N(X)$ of all neighbors of all elements of X contains at least t elements.*

We prove this statement by probabilistic arguments. We choose at random (uniformly and independently) n^c neighbors for each vertex $l \in L$. In this way we obtain a (random) graph where all vertices in L have degree at most n^c (it can be less, as two independent choices for some vertex may coincide).

We claim that this random graph has the required property with positive probability. If it does not, there exists a set $X \subset L$ of some size $t \le 2^n$ and a set Y of size less than t such that all neighbors of all elements of X belong to Y. For fixed X and Y the probability of this event is bounded by $\left(\frac{1}{n^c}\right)^{tn^c}$ since we made tn^c independent choices (n^c times for each of t elements) and for each choice the probability to get into Y is at most $1/n^c$ (the set Y covers at most $1/n^c$ fraction of points in R).

To bound the probability of violating the required property of the graph, we multiply the bound above by the number of pairs X, Y. The set X can be chosen in at most $(2^n)^t$ different ways (for each of t elements we have at most 2^n choices; actually the number is smaller since the order of elements does not matter), and for Y we have at most $(2^k n^c)^t$ choices. Further we sum up these bounds for all $t \le 2^k$. Therefore the total bound is

$$\sum_{t=1}^{2^k} \left(\frac{1}{n^c}\right)^{tn^c} (2^n)^t \left(2^k n^c\right)^t$$

This is a geometric series; the sum is less than 1 (which is our goal) if the base is small. The base is

$$\left(\frac{1}{n^c}\right)^{n^c}(2^n)\left(2^k n^c\right)=\frac{2^{n+k}}{n^{c(n^c-1)}}$$

and $c = 2$ makes it small enough (it even tends to zero as $n \to \infty$). Off-line version is proven. □

Now we have to prove (OM) in its original (on-line) version. Fix a graph $E \subset L \times R$ that satisfies the conditions for the off-line version (for given n and k). Now we use the same graph in on-line setting with the following straightforward ("greedy") strategy. When a new element $x \in L$ arrives, we check if it has neighbors that are not used yet. If yes, one of these neighbors is chosen to be a counterpart of x. If not, x is "rejected".

Before we explain what to do with the rejected elements, let us prove that at most half of 2^k given elements could be rejected. Assume that more than 2^{k-1} elements are rejected. Then less than 2^{k-1} elements are served and therefore less than 2^{k-1} elements of R are used as counterparts. But all neighbors of all rejected elements are used (since this is the only reason for rejection), and we get the contradiction with the condition $\#N(X) \geq \#X$ if X is the set of rejected elements.

Now we need to deal with rejected elements. They are forwarded to the "next layer" where the new task is to find on-line matching for 2^{k-1} elements. If we can do this, then we combine both graphs using the same L and disjoint right parts R_1 and R_2. And the elements rejected at the first layer are satisfied at the second one. In other terms: (n, k) on-line problem is reduced to (n, k) off-line problem and $(n, k-1)$ on-line problem. The latter can then be reduced to $(n, k-1)$ off-line and $(n, k-2)$ on-line problems etc.

Finally we get k levels. At each level we serve at least half of the requests and forwards the remaining ones to the next layer. After k levels of filtering only one request can be left unserved, so one more layer is enough. (Note that we may use copies of the same graph on all layers.)

More precisely, we have proven the following statement: *Let $E \subset L \times R$ be a graph that satisfies the conditions of the off-line version for given n and k. Replace each element in R by $(k+1)$ copies, all connected to the same elements of L as before. Then the new graph provides on-line matchings up to size 2^k.*

Note that this construction multiplies the size of R and the degree of vertices in L by only $(k + 1)$ (a polynomial in n factor). The statement (OM) is proven. □

3 Muchnik's Theorem and Extractors

In this section we present another proof of Muchnik's theorem based on the notion of extractors. This technique was first used in a similar situation in [4]. With this technique we prove some versions of Muchnik's theorem for resource-bounded Kolmogorov complexity (this result was presented in the Master thesis of one of the authors [6]).

3.1 Extractors

Let G be a bipartite graph with N vertices in the left part and M vertices in the right part. The graph may have multiple edges. Let all vertices of the left part have the same degree D. Let us fix an integer $K > 0$ and a real number $\varepsilon > 0$.

Definition 1. *A bipartite graph G is a (K, ε)-extractor if for all subsets S of its left part such that $|S| \geq K$ and for all subsets Y of the right part the inequality*

$$\left| \frac{E(S,Y)}{D \cdot |S|} - \frac{|Y|}{M} \right| < \varepsilon \tag{1}$$

holds, where $E(S, Y)$ stands for the number of edges between S and Y.

In the sequel we always assume that N, M and D are powers of 2 and use n, m and d to denote their logarithms. In this case the extractor may be seen as a function that maps a pair of binary strings of length $n = \log N$ (an index of a vertex on the left) and $d = \log D$ (an index of an edge incident to this vertex) to a binary string of length $m = \log M$ (an index of the corresponding vertex on the right).

The extractor property may be reformulated as follows: consider a uniform distribution on a set S of left-part vertices. The probability of getting a vertex in Y by taking a random neighbor of a random vertex in S is equal to $E(S, Y)/(D \cdot |S|)$; this probability must be ε-close to $|Y|/M$, i.e. the probability of getting a vertex in Y by taking a random vertex in the right part.

It can be proven that (for an extractor graph) a similar property holds not only for uniform distributions on S, but for all distributions with min-entropy at least $k = \log K$ (this means that no string appears with probability greater than $1/K$). That is, an extractor extracts m almost random bits from n quasi-random bits using d truly random bits. For a good extractor m should be close to $k + d$ and d should be small. Standard probabilistic argument shows that for all n, k and ε extractors with near-optimal parameters m and d do exist:

Theorem 2. *For all $1 < K \leq N$, $M > 0$ and $\varepsilon > 0$, there exists an (K, ε)-extractor with*

$$D = \left\lceil \max\left\{ \frac{M}{K} \cdot \frac{\ln 2}{\varepsilon^2},\ \frac{1}{\varepsilon^2}\left(\ln \frac{N}{K} + 1\right) \right\} \right\rceil .$$

In logarithmic scale:

$$d = \log(n - k) + 2\log(1/\varepsilon) + O(1) \quad \text{and} \quad m = k + d - 2\log(1/\varepsilon) - O(1).$$

The proof may be found in [2], which also proves that these parameters are optimal up to an additive term $O(\log(1/\varepsilon))$.

So far no explicit constructions of optimal extractors have been invented. By saying the extractor is *explicit* we mean that there exists a family of extractors for different parameters n and k, other parameters are computable in time poly(n), and the extractor itself as a function of two arguments is computable in poly(n)

time. All known explicit constructions are not optimal in at least one parameter: they either use too many truly random bits, or not fully extract randomness (i.e., $m \ll k + d$), or work not for all values of k. In the sequel we use the following theorem proven in [3]:

Theorem 3. *For all* $1 < k \leq n$ *and* $\varepsilon > 1/\operatorname{poly}(n)$ *there exists an explicit* $(2^k, \varepsilon)$*-extractor for* $m = k + d$ *and* $d = O((\log n \log\log n)^2)$*.*

(For the sake of brevity we use $O(\log^3 n)$ instead of $O((\log n \log\log n)^2)$ in our theorems.)

3.2 The Proof of Muchnik's Theorem

Now we show how to prove Muchnik's theorem using the extractor technique. Consider an extractor with some N, K, D, M and ε. Let S be a subset of its left part with less than K vertices. We say that a right-part element is *bad for* S if it has more than $2DK/M$ neighbors in S, and we say that a left-part element is *dangerous in* S if all its neighbors are bad for S.

Lemma 1. *The number of dangerous elements in* S *is less than* $2\varepsilon K$*.*

Proof. We may assume without loss of generality that S contains exactly K elements (the sets of bad and dangerous elements can only increase when S increases.)

It is immediately clear that the fraction of bad right-part vertices is at most $1/2$, because the degree of a bad vertex is at least twice as large as the average degree. The extractor property reduces this bound from $1/2$ to ε. Indeed, let δ be the fraction of bad elements in the right part. Then the fraction of edges going to bad elements (among all edges starting at S) is at least 2δ. Due to the extractor property, the difference between these fractions should be less than ε. The inequality $\delta < \varepsilon$ follows.

Now we count dangerous elements in S. If their fraction in S exceeds 2ε, then the fraction of edges going to the bad elements (among all edges leaving S) exceeds 2ε too. But the fraction of bad vertices is less than ε, and the difference between two fractions should be at most ε due to the extractor property. □

Now we present a new proof of Muchnik's theorem. As we have seen before, we may assume without loss of generality that the length of a is less than n. Moreover, we assume that conditional complexity $C(a|b)$ equals $k-1$ (otherwise we decrease k) and that $k < n$ (otherwise the theorem is obvious, take $p = a$).

Consider an extractor with parameters n, k, $d = O(\log n)$, $m = k$ and $\varepsilon = 1/n^3$; it exists due to Theorem 2. (The choice of ε will become clear later).

There exists an extractor (with given parameters) of complexity $2 \log n + O(1)$. Indeed, only n and k are needed to describe such an extractor: other parameters are functions of n and k, and we search through all bipartite graphs with given parameters in some natural order until the first extractor with required parameters is found. (This search requires a very long time, so this extractor is not explicit.)

Now assume that an extractor is fixed. We treat the left part of the extractor as the set of all binary strings of length less than n (including a), and the right part as the set of all binary strings of length $m = k$ (we will choose p among them). Consider the set S_b of all strings in the left part such that their complexity conditional to b is less than k (a belongs to this set).

We want to apply Lemma 1 to the set S_b and prove that a is not dangerous in S_b (otherwise it would have too small complexity). So it has a neighbor p that is not bad for S_b, and this p has the required properties. According to this plan, let us consider two cases.

Case 1. If a is not dangerous in S_b then it has a neighbor p that is not bad for S_b. Let us show that p satisfies the claim of the theorem.

Complexity of p is at most $k + O(1)$ because its length is k.

Conditional complexity $C(p|a)$ is logarithmic because p is a neighbor of a in the extractor and to specify p we need a description of the extractor ($2\log n + O(1)$ bits) and the ordinal number of p among the neighbors of a (i.e., $d = \log D = O(\log n)$ bits).

As p is not bad for S_b, it has less than $2D$ neighbors in S_b. If b is known, the set S_b can be algorithmically enumerated; so neighbors of p in S_b can be enumerated too. Thus, to describe a given p and b, we need only a description of the extractor and the ordinal number of a in the enumeration of the neighbors of p in S_b, i.e., $O(\log n)$ bits in total.

Case 2. Assume that a is dangerous in S_b. Since the set S_b is enumerable, the sets of all bad vertices (for S_b) and all dangerous elements in S_b are also enumerable. Therefore, a can be specified by the string b, the extractor and the ordinal number of a in the enumeration. This ordinal number consists of $k-3\log n+O(1)$ bits due to the choice of ε. So, the full description of a given b consists of $k - \log n + O(1)$ bits. This contradicts the assumption that $C(a|b) = k - 1$. Thus, the second case is impossible and Muchnik's theorem is proven. □

3.3 Several Conditions and Prefix Extractors

In [8] An. Muchnik proved also the following generalization of Theorem 1:

Theorem 4. *Let a, b and c be binary strings, and let n, k and l be numbers such that $C(a) < n$, $C(a|b) < k$ and $C(a|c) < l$. Then there exist strings p and q of length k and l, such that one of them is a prefix of the other and the conditional complexities $C(a|p, b)$, $C(a|q, c)$, $C(p|a)$, $C(q|a)$ are of order $O(\log n)$.*

This theorem is quite non-trivial: indeed, it says that information about a that is missing in b and c can be represented by two strings such that one is a prefix of the other (though b and c could be totally unrelated). It implies also that for every three strings a, b, c of length at most n the minimal complexity of a program that transforms a to c and at the same time transforms b to c is at most $\max(C(c|a), C(c|b)) + O(\log n)$.

In fact a similar statement can be proven not only for two but for many (even for poly(n)) conditions. For the sake of brevity we consider only the statement with two conditions.

This theorem also can be proven using extractors. An extractor can be viewed as a function $E\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$. We say that a $(2^k, \varepsilon)$-extractor is a *prefix extractor* if for every $i \leq k$ its prefix of length $m-i$ (i.e., a function $E_i\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m-i}$ determined as a prefix of the initial function) is a $(2^{k-i}, \varepsilon)$-extractor. By using probabilistic method the following theorem can be proven:

Theorem 5. *For all $1 < k \leq n$ and $\varepsilon > 0$ there exists a prefix $(2^k, \varepsilon)$-extractor with parameters $d = \log n + 2\log(1/\varepsilon) + O(1)$ and $m = k + d - 2\log(1/\varepsilon) - O(1)$.*

The proof is quite similar to the proof of Theorem 3: the probabilistic argument shows that a random graph has the required property with probability close to 1; the restriction of a random graph is also a random graph, and the intersection of several events having probability close to 1 is not empty. □

However, using prefix extractors is not enough; we need to modify the argument, since now we need to find two related neighbors in two graphs. So we modify the notion of a dangerous vertex and use the following analog of Lemma 1:

Lemma 2. *Let us call a left-part element weakly dangerous in S if at least half of its neighbors are bad for S. Then the number of weakly dangerous elements in S is at most $4\varepsilon K$.*

The proof is similar to the proof of Lemma 1: since only half of all neighbors are bad, we need twice more elements. □

Now we give a new proof of Theorem 5 based on prefix extractors. Fix a prefix extractor E with parameters n, k, $d = O(\log n)$, $m = k$ and $\varepsilon = 1/n^3$ (again, we may assume that complexity of this extractor is $2\log n + O(1)$). We also may assume that $C(a|b) = k - 1$, $C(a|c) = l - 1$ and (without loss of generality) $k \geq l$.

Let S_b and S_c be the sets of strings of conditional complexity less than k and l conditional to b and c respectively. Call an element *weakly dangerous in S_b* if it is weakly dangerous (in S_b) in the original extractor and *weakly dangerous in S_c* if it is weakly dangerous (in S_c) in the l-bit prefix of E. Since this prefix E_{k-l} is also an extractor, the statement of Lemma 2 holds for S_c. The string a belongs to the intersection of S_b and S_c and is not weakly dangerous in both. Hence, a random neighbor and its prefix are not bad for S_b [resp. S_c] with probability greater than $1/2$. So we can find a k-bit string p such that p and its l-bit prefix q are not bad for S_b and S_c respectively.

They satisfy the requirements. Indeed, the conditional complexities $C(p|a)$ and $C(q|a)$ are logarithmic because p and q can be specified by their ordinal numbers among neighbors of a in the extractor. The string a may be obtained from p and b because p is not bad for S_b in E; similarly, a can be obtained from q and c because q is not bad for S_c in E_{k-l}. This completes the proof of Muchnik's theorem for two conditions. □

3.4 Muchnik's Theorem about Resource-Bounded Complexity

The arguments from Sect. 3.2 together with constructions of explicit extractors imply some versions of Muchnik's theorem for resource-bounded Kolmogorov complexity. In this section we present such a theorem for the space-bounded complexity.

First of all, the definitions. Let φ be a multi-tape Turing machine that transforms pairs of binary strings to binary strings. Conditional complexity $C^{t,s}_{\varphi}(a|b)$ is the length of the shortest x such that $\varphi(x,b)$ produced a in (at most) t steps using space (at most) s. It is known (see [5]) that there exists an *optimal description method* ψ in the following sense: for every φ there exists a constant c such that

$$C^{ct\log t, cs}_{\psi}(a|b) \leq C^{t,s}_{\varphi}(a|b) + c$$

We fix such a method ψ, and in the sequel use notations $C^{t,s}$ instead of $C^{t,s}_{\psi}$.

Now we present our variant of Muchnik's theorem for space-bounded Kolmogorov complexity:

Theorem 6. *Let a and b be binary strings and n, k and s be numbers such that $C^{\infty,s}(a) < n$ and $C^{\infty,s}(a|b) < k$. Then there exists a binary string p, such that*

- $C^{\infty,O(s)+\mathrm{poly}(n)}(a|p,b) = O(\log^3 n)$;
- $C^{\infty,s}(p) \leq k + O(\log n)$;
- $C^{\infty,\mathrm{poly}(n)}(p|a) = O(\log^3 n)$,

where all constants in O- and poly-*notation depend only on the choice of the optimal description method.*

Proof. The proof of this theorem starts as an effectivization of the argument of Sect. 3. To find p effectively, we use an explicit extractor. This increases the conditional complexity of p (when a is given) from $O(\log n)$ to $O(\log^3 n)$, because known explicit extractors use that many truly random bits. The advantage is that now to obtain p from a we need polynomial space (even polynomial time).

First we prove the theorem assuming that the space bound s is space-constructible (i.e., given n we can compute s using space s). Later we explain how to get rid of this restriction.

Assuming that a is not dangerous and p is a good (=not bad) neighbor of a, we can recover a from b and p using $O(\log^3 n)$ extra bits of information and $O(s) + \mathrm{poly}(n)$ space. Indeed, for any string a' we can test in $O(s) + \mathrm{poly}(n)$ space whether $C^{\infty,s}(a'|b) < k$ (we need a counter to stop the computation when it becomes too long to be terminating, but this is just another $O(s)$ if s is known: testing sequentially all strings of length less than k does not increase the space). Therefore, knowing b and p we can enumerate all the strings a' with this property that are neighbors of p and wait until a string with a given ordinal number appears.

The difficulty arises when we try to prove that a is not dangerous; indeed we can enumerate (or recognize: for space complexity it is the same) all dangerous strings in space $O(s) + \mathrm{poly}(n)$, and the number of dangerous strings is small, but this does not give us a contradiction since the space increased from s to $O(s) + \mathrm{poly}(n)$, and even small increase destroys the argument. So we cannot assume a is not dangerous and need to deal somehow with dangerous elements.

To overcome this difficulty, we use the same argument as in Sect. 2.3. The dangerous elements are treated on the next layer, with reduced k and other extractor graph. We need $O(k)$ layers (in fact even $O(k/\log n)$ layers) since at every next layer the number of dangerous elements that still need to be served is reduced at least by n factor. Note also that the space overhead needed to keep the accounting information is $\mathrm{poly}(n)$ and we never need to run in parallel several computations that require space s.

So we get the theorem for space-constructible s. In the general case some changes are needed (we used s to restrict the computation space). Let us sequentially use space bounds $s' = 1, 2, \ldots$: to enumerate all strings a' such that $C^{\infty,s}(a'|b) < k$, we sequentially enumerate all strings that can be obtained from b and k-bit encoding using space $s' = 1, 2$, etc. The corresponding set increases as s' increases, and at some point we enumerate all strings a' such that $C^{\infty,s}(a'|b) < k$ (though this moment is not known to us). Note that we can avoid multiple copies of the same string: performing the enumeration for s', we check for every string whether it has appeared earlier (using $s' - 1$ instead of s'). This requires a lot of time, but only $O(s)$ space. Knowing the ordinal number of a in the entire enumeration, we stop as soon as it is achieved; hence, the enumeration process requires only space $O(s) + \mathrm{poly}(n)$ (though s is not specified explicitely).

Similarly, the set of dangerous a (that go to second or higher layer) increases as s' increases, and can be enumerated sequentially for $s' = 0, 1, 2\ldots$ without repetitions in $O(s') + \mathrm{poly}(n)$ space. Therefore, at every layer we can use the same argument (enumerating all the elements that reach this layer and at the same time are neighbors of p until we produce as many of them as required). □

3.5 Muchnik's Theorem for *CAM*-Complexity

The arguments from the previous sections cannot be applied for Kolmogorov complexity with polynomial *time* bound. Roughly speaking, the obstacle is the fact that we cannot implement an exhaustive search (over the list of 'bad' strings) in polynomial time (unless $P = NP$). The best result that we can prove for poly-time bounded complexity involves a version of Kolmogorov complexity introduced in [10]:

Definition 2. *Let U_n be a non-deterministic universal Turing machine. Arthur-Merlin complexity $CAM^t(x|y)$ is the length of a shortest string p such that*

1. $\mathrm{Prob}_r[U_n(y,p,r)$ *accepts, and all accepting pass return* $x] > 2/3$
2. $U_n(y,p,r)$ *stops in at most time t (for all branches of non-deterministic computation).*

As always, $CAM^t(x) := CAM^t(x|\lambda)$.

Intuitively, a CAM-description p of a string x (given another string y) is an interactive Arthur–Merlin protocol: Arthur himself can do probabilistic polynomial computations, and can ask questions to all-powerful (but not trustworthy) Merlin; Merlin can do any computations and provide to Arthur any requested certificate. So, Arthur should ask such questions that the certificates returned by Merlin could be effectively used to generate x. With this version of resource-bounded Kolmogorov complexity we have a variant of the Muchnik theorem:

Theorem 7. *For every polynomial t_1 there exists a polynomial t_2 such that the following condition holds. For all strings a, b of length at most n such that $C^{t_1(n),\infty}(a|b) \le k$, there exists a string p of length $k + O(\log^3 n)$ such that*

- $C^{t_2(n),\infty}(p|a) = O(\log^3 n)$ *and*
- $CAM^{t_2(n)}(a|b, p) = O(\log^3 n)$.

In the proof of this theorem we cannot use an arbitrary effective extractor. We employ very essentially the properties of one particular extractor constructed by L. Trevisan [11]. Our arguments mostly repeat the proof of Theorem 3 from [10].

Acknowledgments

This article is based on the discussions and results reported at the Kolmogorov seminar (Moscow). Preliminary versions appeared as [9] and [6]. The authors are grateful to all participants of the seminar for many useful comments.

References

1. Musatov, D., Romashchenko, A., Shen, A.: Variations on Muchnik's Conditional Complexity Theorem. Full version, http://arxiv.org/abs/0904.3116
2. Radhakrishnan, J., Ta-Shma, A.: Bounds for dispersers, extractors, and depth-two superconcentrators. SIAM Journal on Discrete Mathematics 13(1), 2–24 (2000)
3. Reingold, O., Shaltiel, R., Wigderson, A.: Extracting randomness via repeated condensing. SIAM Journal on Computing 35(5), 1185–1209 (2006)
4. Buhrman, H., Fortnow, L., Laplante, S.: Resource bounded Kolmogorov complexity revisited. SIAM Journal on Computing 31(3), 887–905 (2002)
5. Li, M., Vitanyi, P.: An Introduction to Kolmogorov Complexity and Its Applications, 2nd edn. Springer, Heidelberg (1997)
6. Musatov, D.: Extractors and an effective variant of Muchnik's theorem. Diplom (Master thesis). Faculty of Mechanics and Mathematics, MSU (2006) (in Russian), http://arxiv.org/abs/0811.3958
7. Muchnik, A.A.: On basic structures of the descriptive theory of algorithms. Soviet Math. Dokl. 32, 671–674 (1985)
8. Muchnik, A.: Conditional complexity and codes. Theoretical Computer Science 271(1-2), 97–109 (2002)
9. Shen, A.: Combinatorial proof of Muchnik's theorem. In: Hutter, M., Merkle, W., Vitanyi, P. (eds.) Kolmogorov complexity and applications, Dagstuhl Seminar Proceedings 06051, ISSN 1862–4405, http://drops.dagstuhl.de/opus/volltexte/2006/625

10. Buhrman, H., Lee, T., van Melkebeek, D.: Language compression and pseudorandom generators. In: Proc. of the 15th IEEE Conference on Computational Complexity. IEEE, Los Alamitos (2004)
11. Trevisan, L.: Construction of extractors using pseudo-random generators. In: Proc. 45th FOCS, pp. 264–275
12. Slepian, D., Wolf, J.K.: Noiseless coding of correlated information sources. IEEE Transactions on information Theory 19, 471–480 (1973)

An Optimal Bloom Filter Replacement Based on Matrix Solving*

Ely Porat

Bar-Ilan University

Abstract. We suggest a method for holding a dictionary data structure, which maps keys to values, in the spirit of Bloom Filters. The space requirements of the dictionary we suggest are much smaller than those of a hashtable. We allow storing n keys, each mapped to value which is a string of k bits. Our suggested method requires $nk + o(n)$ bits space to store the dictionary, and $O(n)$ time to produce the data structure, and allows answering a membership query in $O(1)$ memory probes. The dictionary size does not depend on the *size of the keys*. However, reducing the space requirements of the data structure comes at a certain cost. Our dictionary has a small probability of a one sided error. When attempting to obtain the value for a key that is stored in the dictionary we always get the correct answer. However, when testing for membership of an element that is not stored in the dictionary, we may get an incorrect answer, and when requesting the value of such an element we may get a certain random value. Our method is based on solving equations in $GF(2^k)$ and using several hash functions.

Another significant advantage of our suggested method is that we do not require using sophisticated hash functions. We only require *pairwise* independent hash functions. We also suggest a data structure that requires only nk bits space, has $O(n^2)$ preprocessing time, and has a $O(\log n)$ query time. However, this data structures requires a *uniform* hash functions.

In order replace a Bloom Filter of n elements with an error proability of 2^{-k}, we require $nk + o(n)$ memory bits, $O(1)$ query time, $O(n)$ preprocessing time, and only pairwise independent hash function. Even the most advanced previously known Bloom Filter would require $nk + O(n)$ space, and a uniform hash functions, so our method is significantly less space consuming especially when k is small.

Our suggested dictionary can replace Bloom Filters, and has many applications. A few application examples are dictionaries for storing bad passwords, differential files in databases, Internet caching and distributed storage systems.

1 Introduction

A *Bloom Filter* is a very basic data structure which, given a set of n elements, allows us to quickly decide whether a given element is in the set or not. The main

* A video presentation can be found in *http://www.youtube.com/watch?v=947gWqwkhu0.*

advantage of Bloom Filters is that they are very memory efficient — a Bloom Filter only requires space linear in the number of elements in the set, while other data structures use memory linear in the size of the represented elements in the set. When the elements stored in the set do not have a succinct representation, this is a very significant advantage. For example, consider strings, with average size of 800 bits. A hashtable for storing 100,000,000 such strings would require at least 800*100,000,000 bits, so a hard disk must be used for the table, and lookups would be rather slow. A basic Bloom Filter based structure would only require 145,000,000 bits, which can easily be stored in the main memory. On the other hand, the Bloom Filter achieves this at a certain cost. A Bloom Filter has a certain probability of returning a wrong answer. The error is one sided: if the key is in the set, the Bloom Filter will always return the correct answer, but if the key is not in the set, it might return a wrong answer. However, for many applications, it is possible to overcome this problem, and still gain from the low space requirements of the Bloom Filter.

The main use of the Bloom Filter is to reduce the memory that the data structure uses. The basic Bloom Filter [1] (invented in 1970) used $n \log e$ memory bits and returned the answer using a single probe to memory, with error probability of $\frac{1}{2}$ (for a false positive). One way to reduce the error probability is to run the basic Bloom Filter k times, therefore it would require $nk \log e$ memory bits and k memory probes in order to answer a query.

2 Related Work

During the past few years, several papers have been published on Bloom Filter [3,5,6,12,15]. Most of which provided methods for reducing the memory and the number of probes required, but only considered the case where k is big enough. One more disadvantage of these newer methods is that they do not allow "insertion" operations, which were possible to perform using the original Bloom Filter technique. Yet another disadvantage of these newer methods is that they require universal hash functions. Such functions are computationally inefficient, or have large memory requirements. Recently two papers [4,7] were published which used the same idea of matrix solving in bloom filter like this paper. These papers were done independent to our results. However we think that this paper still have some more issue with worth publishing.

In this paper we provide a new data structure that can replace Bloom Filters, and has lower space requirements. Our data structure requires $nk + o(n)$ memory bits (which is optimal up to $o(n)$), and each query takes $O(1)$ memory probes. However, like most of the other Bloom Filter replacements, our data structure is static and does not support insertions. Building our data structure requires $O(n)$ preprocessing time and $O(n)$ memory. This data structure is based on solving equations, and uses hash functions. We only require hash functions that are *pairwise* independent.

In addition, we suggest a similar data structure that requires only nk memory bits, $O(\log n)$ query time, and $O(n^2)$ preprocessing time. However, this data structure requires uniform hash functions.

3 Applications of Bloom Filters

Bloom Filters, as well as Bloom Filter replacements such as the one we suggest, have many applications. A good survey of Bloom Filter uses can be found in [2]. A few examples are given below.

Dictionaries: Early versions of UNIX's spell checker used a Bloom Filter of the dictionary instead of the dictionary itself. This Bloom Filter left several words misspelled, but the memory in these days was valuable resource and the memory it save was worth it [11,15].

The Bloom Filter was proposed as a method to succinctly store a dictionary of unsuitable passwords for security purposes by Spafford [16]. Manber and Wu describe a simple way to extend the technique so that passwords that are within edit distance 1 of the dictionary word are also not allowed [10]. In this setting, a false positive could force a user to avoid a password even if it is not really in the set of unsuitable passwords.

Databases: Bloom Filters can also be used for differential files [9,16]. Suppose that all the changes to a database that occur during the day are stored in a differential file and are updated back to the database only at the end of a day. During that day, every read from the database should first be checked in that differential file to be sure that the record read is the most recent. This file might be large, so reading through it can be slow, as opposed to querying a database, but still obligated. A possible solution to this problem is keeping a Bloom Filter of the records that have changed. Here, a false positive forces a read of the differential file even when a record has not been changed.

Internet Cache Protocol: Fan, Cao, Almeida, and Broder describe Summary Cache, which uses Bloom Filters for Web cache sharing [8]. In this setup, proxies cooperate in the following way: on a cache miss, a proxy attempts to determine if another proxy cache holds the desired Web page; if so, a request is made to that proxy rather than trying to obtain that page from the Web. For such a scheme to be effective, proxies must know the contents of other proxy caches. In Summary Cache, to reduce message traffic, proxies do not transfer URL lists corresponding to the exact contents of their caches, but instead periodically broadcast Bloom Filters that represent the contents of their cache. If a proxy wishes to determine if another proxy has a page in its cache, it checks the appropriate Bloom Filter. In the case of a false positive, a proxy may request a page from another proxy, only to find that that proxy does not actually have that page cached. In that case, some additional delay has been incurred. But the load on the proxy servers was reduced therefore making them work faster.

Caching for Google's BigTables: BigTable is a distributed storage system for managing structured data that is designed to scale to a very large size: petabytes of data across thousands of commodity servers. Many projects at Google store data in BigTables, including web indexing, Google Earth, and Google Finance. These applications place very different demands on the BigTable, both in terms of data size (from URLs to web pages to satellite imagery) and latency requirements (from back end bulk processing to real-time data serving). Despite these varied demands, BigTable has successfully provided a flexible, high-performance solution for all of the above Google products. In some of the BigTable applications most of the queries aren't in the table. In BigTables Bloom Filter is used to determine whether a query is in the BigTable in first place, thus reducing disk accesses. A Bloom Filter can be also used in the client side as well to reduce the communication and latency.

4 Outline

The structure of this paper is as follows: In section 5 we define the dictionary data structure and give a high-level view of our method, as well as a basic result. In section 6 we show how to improve the data structure to support queries in $O(1)$ time, and how to do the preprocessing in $O(n)$ time. In section 7 we show several methods to reduce constants hidden in these space complexity, which may be important in practice. In section 8 we explain why and how simple pairwise independents hash function are enough. In section 9 we show how to use the dictionary data structure in order to get a good Bloom Filter replacement.

5 Dictionary Based on Matrix Solving

Dictionaries are data structures that hold key-value pairs. This section describes a method for concise representation of dictionaries with one sided errors, in the spirit of Bloom Filters.

Definition 1. *A one sided error dictionary (U,k,n) is a data structure that holds values for keys. It is a mapping from* $x_1, x_2, \ldots, x_n \in U$ *to* $d_1, d_2, \ldots, d_n \in \{0, 1, \ldots, 2^k - 1\}$*. Given a key* x_i*, a dictionary allows retrieving* d_i*. However, given a key* x *which is not one of the* x_i*'s it may return any value.*

We now show how to build a dictionary which requires a storage space of $nk+o(n)$ bits. The high level concept behind our method is solving equations. Assume we have a fully random hash function h from U to n variable equation in $GF(2^k)$ (we later show how to remove the fully random assumption later), i.e. $h : U \rightarrow GF(2^k)^n$. We go over all the x_i's and we write the equation $h(x_i) \cdot \boldsymbol{b} = d_i$. We get n equations with n variables. If these equations are *independent* we can solve them in $O(n^3)$ time. This can be done in a one time preprocessing, after which we can store the hash function h and the vector $\boldsymbol{b}$ as our data structure. The vector $\boldsymbol{b}$ requires nk bits space. To answer a query x we apply h on x and compute $h(x) \cdot \boldsymbol{b}$ and return the answer. If x is one of the x_i's we get the correct d_i. If x is not one of the x_i's we might return an erroneous answer. The overall query time is $O(n)$.

However, this process only works when we get an *independent* set of equations. We now examine the probability of obtaining such an independent equation set.

Theorem 1. *The probability that our method generates an* independent *set of n equations on $n + c$ variables in the field $GF(2^k)$ is at least* $1 - \frac{1}{2^{kc}(2^k-1)}$

Proof. We order the generated equations according to the order in which they are constructed. The set of the equations is dependent when there exist i such that equations $1, 2, \ldots, i-1$ and equation i are dependent. The probability that equation i and equations $1, 2, \ldots, i-1$ are dependent is at most $\frac{(2^k)^i}{(2^k)^{n+c}}$ (the probability is even lower when there are dependent equations before that index). We apply the union bound and get that the probability that there exists an i such that the equation i and the equations before it are dependent is at most $\sum_{i=0}^{n} \frac{(2^k)^i}{(2^k)^{n+c}} < \frac{1}{2^{kc}(2^k-1)}$

Corollary 1. *Even for $c = 0$ we get an independent set of equations with constant probability. Therefore we need to run the preprocessing algorithm $O(1)$ time, each time with a different hash function, in order to get an independent set of equations.*

The main disadvantage of this data structure is that it requires $O(n)$ time in order to answer a query. One possible improvement can be achieved by using t-sparse equations.

Definition 2. *t-sparse equations are equations of the form*
$\sum_{i=1}^{n} a_i$*, where* $|\{a_i | a_i \neq 0\}| \leq t$.

Using t-sparse equations the query time shrinks to t memory probes, $O(t)$ time. However we need at least $m = n(1 + e^{-t-\epsilon})$ variables in our equations set in order to have a full independent equations set.

Theorem 2. *If we have n t-sparse random equations in less than $m = n(1 + e^{-t-\epsilon})$ variables, the equations will be dependent with high probability.*

Proof. When we have n t-sparse random equations on $m = n(1 + e^{-t-\epsilon})$ there are some variables that we do not use. Because we can look on it as throwing $t \times n$ balls to m cells. The expected number of empty cells is $m(1 - \frac{1}{m})^{tn} \approx me^{-\frac{tn}{m}}$. Therefore the expected number of variables we use in our equations is $m(1 - e^{-\frac{tn}{m}})$. If $m(1 - e^{-\frac{tn}{m}}) < n$, we get n equations on less then n variables and therefore they will not be independent.

Actually if we take $n(1+e^{-t})$ we will have a good probability to get independent set of equations.

Note that the preprocessing of the "sparse" data structure is $O(tn^2)$, using the Wiedemann algorithm [18] for solving sparse linear equations.

6 Improved Dictionary

We now show how to reduce the query time to $O(1)$ memory probes. We also reduce the preprocessing time to $O(n)$. The high level idea behind the method suggested in this section is to divide $x_1, x_2, \ldots, x_n$ randomly to small buckets, and to run the same algorithm on each of the buckets.

We can randomly hash the keys to $\frac{n}{s}$ buckets using hash function $h_1 : U \to \{1, 2, \ldots, \frac{n}{s}\}$. The expected number of keys in each bucket would be s, and if s is big enough, with high probability there will not be a bucket with more then $2s$ keys (if there such a bucket we can choose another hash function h_1 and so on). Querying for x is done by simply applying $h_1(x)$ and going to the $h_1(x)$'th data structure. In that data structure we query for x as done in section 5. The $h_1(x)$ data structure does not contain more then $2s$ keys, so it would take $O(s)$ time to answer the query. The preprocessing is now performed by choosing h_1 and checking if there is no bucket with more than $2s$ keys. If there is such a bucket, we choose another hash function h_1. This is done $O(1)$ times. We then divide the keys $x_1, x_2, \ldots, x_n$ to the buckets and run the same preprocessing method described in section 5 on each bucket.

Overall it would take $O(\frac{n}{s}s^3) = O(ns^2)$. The memory that this data structure consumes is $nk + O(\frac{n}{s}\log n)$ memory bits. The $O(\frac{n}{s}\log n)$ is required in order to maintain pointers to each of the data structures. Naturally, our method works best when s is small. However, if we reduce s too much we we lose the fact that with high probability there is no bucket which is bigger then $2s$, and the $O(\frac{n}{s}\log n)$ becomes significant.

We solve this problem by using a two-level hashing. We first explain the preprocessing and then show how to run a query. Given $x_1, x_2, \ldots, x_n$ we hash them using $h_1 : U \to \{1, 2, \ldots, \frac{n}{\log^2 n}\}$, which we now only require to be pairwise independent, to $\frac{n}{\log^2 n}$ buckets. It might be the case that there are some buckets which more then $2\log^4 n$ keys. We call such big buckets *bad buckets*. We choose another h_1 hash function only if we will get more then $\frac{n}{\log^2 n}$ keys hashed to bad buckets.

Theorem 3. *The probability that there are more then $\frac{n}{\log^2 n}$ keys hashed to bad buckets is at most $\frac{1}{2}$*

Proof. We denote by B_i the number of keys hashed to bucket i. Using Markov's inequality we get:

$$Pr\left[Bucket\ i\ is\ bad\right] = Pr\left[B_i > 2\log^4 n\right] = Pr\left[B_i > 2\log^2 E(B_i)\right] < \frac{1}{2\log^2 n}$$

Denote by X_i the event that x_i is hashed to a bad bucket, and by $X = \sum_{i=1}^{n} X_i$ the number of keys hashed to a bad bucket. $Pr\left[X_i = 1\right] < \frac{1}{2\log^2 n}$ therefore $E(X) < \frac{n}{2\log^2 n}$. $Pr\left[X > \frac{n}{\log^2 n}\right] < Pr\left[X > 2E(X)\right] < \frac{1}{2}$ by markov inequality.

Corollary 2. *It takes $O(n)$ time to find a hash function h_1 that we can use for the rest of the procedure.*

After we find a good hash function h_1, we deal with all the keys that are hashed to a bad bucket using a regular dictionary data structure. It takes at most

$O(\frac{n}{\log n}) = o(n)$ bits (we can easily modify it to take $O(\frac{n}{\log^c n})$ bits for any constant c).

Denote by B_i the number of keys hashed by h_1 to bucket i. Each good bucket i (such that $B_i < 2\log^4 n$) is splitted again to sub-buckets using $h_{2,i} : U \to \{1, 2, \ldots, \frac{B_i}{\frac{1}{2}\sqrt{\frac{\log n}{k}}}\}$ (we now assume that $h_{2,i}$ is fully random, in section 8 we show how to relax this assumption). If we get a sub-bucket which is bigger then $\sqrt{\frac{\log n}{2k}}$ we choose another $h_{2,i}$.

Theorem 4. *When we split a bucket to a sub-buckets the probability that there exist sub-bucket which more then* $\sqrt{\frac{\log n}{2k}}$ *keys hashed to it is at most* $\frac{1}{2}$

Proof. The expected number of keys hashed to a sub-bucket is $\frac{1}{2}\sqrt{\frac{\log n}{k}}$. Using Chernoff's inequality we get that the probability for each sub-bucket to have more then $\sqrt{\frac{\log n}{2k}}$ is much smaller then $\frac{1}{\log^4 n}$. Using the union bound we get that the probability that there exist a sub-bucket with more then $\sqrt{\frac{\log n}{2k}}$ is smaller then $\frac{1}{2}$, since we have less then $\frac{\log^4 n}{2}$ sub-buckets.

Corollary 3. *It takes $O(B_i)$ time to find such an $h_{2,i}$. Overall, finding a hash function $h_{2,i}$ for all i's requires $O(n)$ time.*

We now have many smaller dictionary sub-problems. Each one of them has a size of less then $\sqrt{\frac{\log n}{2k}}$. We solve each one of them using the method mentioned in section 5. For each sub problem we get a random matrix of size bounded by $\sqrt{\frac{\log n}{2k}} \times \sqrt{\frac{\log n}{2k}}$ over $GF(2^k)$. The number of different such matrices is at most $2^{k(\sqrt{\frac{\log n}{2k}})^2} = \sqrt{n}$. Thus we can list all the different matrices and solve them in advance in time $O(\sqrt{n}\log^{1.5} n)$, and the list would require $O(\sqrt{n}\log n)$ memory bits.

Thus the preprocessing takes $O(n)$ time, since we can solve each sub-problem by simply looking in the list.

We store the data structure as follows. We store all the keys which map to bad buckets using a regular dictionary, with $o(n)$ memory bits. We store a big array of less then n words, each consisting of k bits which are the concatenation of all the sub-buckets in all the buckets. We also store a select data structure which gives us the ability to jump in $O(1)$ memory probes to each of the buckets and sub-buckets. It requires $o(n)$ memory bits as well. Finally, we store all the hash functions. In section 8 we show how they can be stored. Overall we use $nk + o(n)$ memory bits.

To answer a query we simply use h_1 in order to see to which of the bucket we need to go. If it is a bad bucket, we look for the query in the regular dictionary data structure. Otherwise we use $h_{2,i}$ in order to find in which sub-bucket the query falls. All the operation up to this point take $O(1)$ time, and we use one probe to the memory to retrieve $h_{2,i}$. We use the dictionary data structure of the sub-bucket in order to answer the query. It takes $O(1)$ probes to the memory (we retrieve $\sqrt{\frac{\log nk}{2}}$ bits in these probes, and in the last probe we take a word), but it takes $O(\sqrt{\frac{\log n}{2k}})$ time to retrieve the answer. In order to reduce that time to $O(1)$ we have two options: we can either use sparse equations or we can construct a table holding all the answers to all of the possible equations on all of the possibles assignments, and answer the query in $O(1)$ time by probing a table for getting the answer[1].

7 Practical Improvements

We now examine a few practical improvements for our method.

Sparse equations: Whenever we use the solution of section 5 (even inside the sub-buckets) we can use $\ln n$ sparse equations set (in the sub-bucket case it is $\ln \log n$). This still works fine even when we use only n variables, therefore it requires $nk + o(n)$ memory bits. Note that this will not work if we take only even number of variables per equation.

Another sparse equations improvement is to create equations which will be more or less local i.e. the $\{i | a_i \neq 0\}$ will be close to each other. This way need less memory probes, because in each memory probe we can get $O(\log n)$ continues bits.

Another counting argument: If we make each sub-bucket bigger, we can gain in the $o(n)$ overhead. Denote by s the maximum number of keys hashed to a sub-bucket. For each such sub-bucket (from section 6). In section 6 we had a certain preprocessing analysis. We now give an alternative one. In each sub-bucket we hash keys to $\{1, 2, \ldots, s^2\}$. With probability of at least $\frac{1}{2}$ there will not be any collision in this hash. If we do have a collision we choose another hash function. On average, 2 bits are required to store which hash function we use in each sub-bucket. We now have a list of at most s keys from the universe $\{1, 2, \ldots, s^2\}$, where each key gets a value in $GF(2^k)$. Note that if we have the same set of keys in two different sub-buckets, we can use the same set of equations even if they do not get the same values — being a full rank equations set does not depand on the values (the free vector).

Thus, the number of different sets of equations we use is $\binom{s^2}{s}$. For $s < \frac{\log n}{2 \log \log n}$ we get $o(\sqrt{n})$ different equations sets. For each of the equations sets we compute the inverse and store it in a hashtable. The naive way to perform the preprocessing using this technique takes $\sum_{i=1}^{\#sub-bucket} O(sub - bucket - size^2) = O(n \frac{\log n}{\log \log n})$ time, because we need to multiply the inverse matrix by the data for each sub-bucket. However we can collect $O(\log n)$ sub-buckets that map to the same matrix (inverse matrix) and multiply the same matrix by $O(\log n)$ different values vectors. We get $O(\log n)$ speed up in time using word operations.

[1] We can play a little more with the size of each sub-bucket in order to do this in $o(n)$ space.

Therefore the preprocessing running time shrinks back to $O(n)$. Making the equations $O(\ln \log n)$ sparse and local we get $O(1)$ query time as well[2].

A real nk solution: We can get rid of the extra $o(n)$, by solving n equations in n variables. Each equation will be $\ln n$ sparse equation. The preprocessing time takes $O(n^2)$ using the block Wiedemann algorithm [17], and the query takes $O(\log n)$ time. Note that we need to use a uniform hash function for this result.

8 Using Simple Hash Functions

We only assume a truly random hash function inside the buckets. Each bucket consist of at most $\log^4 n$ keys. Therefore we can construct hash function by simply using array R of $\log^8 n$ random numbers and a pairwise independent hash function $h : U \rightarrow \{0, 1, \dots, \log^8 n\}$. The result for the new hash function is $R[h(x)]$. Given that we hash at most $\log^4 n$ keys. The probability that there exist two keys that use the same random number is less then $\frac{1}{2}$. Therefore we got a random enough hash function with probability $\frac{1}{2}$. If we store $2 \log$ hash functions like this, with probability bigger then $1 - \frac{1}{n}$ each bucket will have at least one hash function which will satisfied it. The only extra space required is $O(\log^9 n)$ memory bits.

9 Membership Queries

We first define a membership data structure.

Definition 3. *A Membership data structure(n,k) for $x_1, x_, \dots, x_n \in U$ is a data structure that allows answering membership queries. Given a query x where x is one of the x_i's, the data structure always returns* 1*, and given a query x where x is not one of the x_i it returns 0 with probability of at least 2^{-k}.*

We can easily build a membership data structure given a dictionary data structure. We simply choose random pairwise independent hash function $h : U \rightarrow \{0, 1, \dots, 2^k - 1\}$ and we store a dictionary that map x_i to $h(x_i)$.

In order to check if x is in the data structure we simply query x from the dictionary data structure and check if it's value equal to $h(x)$. If x is in the data structure it will always return 1.

Theorem 5. *If x isn't in the data structure we will return* 1 *with probability 2^-k.*

Proof. We choose the hash function independent from the dictionary data structure. Therefore the answer of the query x from the dictionary data structure, if x isn't a member is a k-bit string which is independent to $h(x)$. Then the probability that they are equal is 2^{-k} because $h(x)$ is random.

[2] Using tables as well.

10 Conclusions and Open Problems

We have suggested a new data structure that can replace Bloom Filters. This data structure allows maintaining a dictionary mapping keys to values, and allows retrieving the value for a key with a one sided error. Our method has significant advantages over Bloom Filter and other previously know Bloom Filter replacements. It uses only $nk + o(n)$ memory bits (which is optimal up to $o(n)$), and each query takes $O(1)$ memory probes. Also, we only require pairwise independent hash function.

We have also suggested a similar data structure, that has an even lower space requirement, of only nk memory bits. However, it has a $O(\log n)$ query time and requires $O(n^2)$ preprocessing time. Also, this data structure requires uniform hash functions.

Despite its advantages, the method we suggest, like several other Bloom Filter replacements, does not allow "insertion" operations, which the original Bloom Filter technique does support.

We believe the preprocessing phase of our algorithm can be distributed easily. In fact, we believe it should be distributed in most applications, due to the memory it consumes.

There are several directions open for future research. First, it will be interesting to see if it is possible to design a data structure which only requires one pass on the input elements and with small additional memory. Also, it may be possible to develope a fully *dynamic* data structure, with space requirements lower than those of the traditional Bloom Filter.

References

1. Bloom, B.H.: Space/time trade-offs in hash coding with allowable errors. Commun. ACM 13(7), 422–426 (1970)
2. Broder, A., Mitzenmacher, M.: Network applications of bloom filters: A survey (2002)
3. Brodnik, A., Ian Munro, J.: Membership in constant time and almost-minimum space. SIAM J. Comput. 28(5), 1627–1640 (1999)
4. Charles, D., Chellapilla, K.: Bloomier filters: A second look. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 259–270. Springer, Heidelberg (2008)
5. Chazelle, B., Kilian, J., Rubinfeld, R., Tal, A.: The bloomier filter: an efficient data structure for static support lookup tables. In: SODA 2004: Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 30–39. Society for Industrial and Applied Mathematics (2004)
6. Cohen, S., Matias, Y.: Spectral bloom filters. In: SIGMOD 2003: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pp. 241–252. ACM, New York (2003)
7. Dietzfelbinger, M., Pagh, R.: Succinct data structures for retrieval and approximate membership (extended abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 385–396. Springer, Heidelberg (2008)

8. Fan, L., Cao, P., Almeida, J., Broder, A.Z.: Summary cache: a scalable wide-area web cache sharing protocol. IEEE/ACM Trans. Netw. 8(3), 281–293 (2000)
9. Gremillion, L.L.: Designing a bloom filter for differential file access. Commun. ACM 25(9), 600–604 (1982)
10. Manber, U., Wu, S.: An algorithm for approximate membership checking with application to password security. Inf. Process. Lett. 50(4), 191–197 (1994)
11. McIlroy, M.D.: Development of a spelling list. IEEE Transactions on Communications 30(1), 91–99 (1982)
12. Mitzenmacher, M.: Compressed bloom filters. IEEE/ACM Trans. Netw. 10(5), 604–612 (2002)
13. Mullin, J.K., Margoliash, D.J.: A tale of three spelling checkers. Softw. Pract. Exper. 20(6), 625–630 (1990)
14. Mullin, J.K.: A second look at bloom filters. Commun. ACM 26(8), 570–571 (1983)
15. Pagh, A., Pagh, R., Srinivasa Rao, S.: An optimal bloom filter replacement. In: SODA 2005: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, pp. 823–829. Society for Industrial and Applied Mathematics (2005)
16. Spafford, E.H.: Opus: Preventing weak password choices. Computer and Security 10, 273–278 (1992)
17. Villard, G.: A study of coppersmith's block wiedemann algorithm using matrix polynomials
18. Wiedemann, D.H.: Solving sparse linear equations over finite fields. IEEE Trans. Information Theory IT-32(1), 54–62 (1986)

Aperiodicity Measure for Infinite Sequences

Yuri Pritykin[1,2] and Julya Ulyashkina[1]

[1] Department of Mechanics and Mathematics,
Moscow State University, Russia
[2] Computer Science Department,
Princeton University, USA

Abstract. We introduce the notion of aperiodicity measure for infinite symbolic sequences. Informally speaking, the aperiodicity measure of a sequence is the maximum number (between 0 and 1) such that this sequence differs from each of its non-identical shifts in at least fraction of symbols being this number. We give lower and upper bounds on the aperiodicity measure of a sequence over a fixed alphabet. We compute the aperiodicity measure for the Thue–Morse sequence and its natural generalization the Prouhet sequences, and also prove the aperiodicity measure of the Sturmian sequences to be 0. Finally, we construct an automatic sequence with the aperiodicity measure arbitrarily close to 1.

1 Introduction

Combinatorics on words is a deeply studied field in theoretical computer science and discrete mathematics. In this paper we focus on infinite words, or sequences, over a finite alphabet. Periodic sequences have the simplest structure, and it is natural to try to measure how far a sequence may be from any periodic sequence.

In this paper we introduce the notion of aperiodicity measure AM for infinite symbolic sequences. Our definition is based on the discrete version of Besicovitch distance that was used by Morse and Hedlund [14] when defining sequences that they called almost periodic[1]. The same approach was also used in [7] when defining α-aperiodic two-dimensional sequences. As it is essentially noticed in [7], if $\mathrm{AM}(x) > \alpha$ for a sequence x, then x has Besicovitch distance at least $\alpha/2$ with every eventually periodic sequence. In [14] it is also proved that $\mathrm{AM}(\mathbf{t}) \geqslant 1/4$ where $\mathbf{t}$ is the Thue–Morse sequence.

Informally speaking, the aperiodicity measure of a sequence is the maximum number (between 0 and 1) such that this sequence differs from each of its non-identical shifts in at least fraction of symbols being this number. Our interest to this notion was mostly inspired by the following conjecture from the personal communication with B. Durand, A. Romashchenko, and A. Shen, that we positively prove as Theorem 6.

[1] This term “almost periodic” from [14] should not be mixed with the recent usage of the term “almost periodic sequence” which stands for sequences also known as uniformly recurrent or minimal, e.g., see [16] for a survey.

Conjecture. For every $\alpha < 1$ there exists an automatic sequence x such that $\mathrm{AM}(x) \geqslant \alpha$.

The solution of this conjecture allows to simplify the construction of a strongly aperiodic tiling from [7].

Besides this conjecture, we believe that the notion of aperiodicity measure is interesting and natural itself, and the main goal of the paper is to support this statement.

Other similar notions and results have appeared in the literature: to name a few, see [14] on Besicovitch-almost-periodic sequences, [8] on tilings of the Thue–Morse sequence, [10] on approximate squares in sequences. The closest to ours seems to be the notion of correlation measure introduced in [12] and then studied in a series of papers currently concluding with [5], see also [13] and many others. Their correlation measure of order 2 is essentially the same as our aperiodicity measure for binary sequences, though in general motivation, frameworks, and approaches are somewhat different. After the submission of our paper, we became aware of the recent paper [9] continuing the investigations of the aforementioned correlation measure. In particular, a result from [9] improves the result of our Theorem 2 below.

The paper is organized as follows. In Section 2 we give necessary preliminaries. In Section 3 we define the aperiodicity measure AM of an infinite sequence and then study some basic properties of this notion. We prove that there exist sequences with AM arbitrarily close to 1 (Theorem 1), though there exists an upper bound strictly less than 1 for AM of sequences over a fixed finite alphabet (Theorem 2). Then we calculate AM for the Thue–Morse sequence (Theorem 3) and for the Sturmian sequences (Theorem 4). In Section 4 we construct an automatic sequence with AM arbitrarily close to 1 (Theorem 6), though first we prove that the Prouhet sequences, natural generalization of the Thue–Morse sequence, do not suffice for this purpose (Theorem 5). Due to space constraints, some proofs are only sketched. Section 5 concludes the paper with a number of open problems.

2 Preliminaries

We use all common definitions and notions of combinatorics on words, which can be found, i.e., in [4] or [11]. We recall some of them here for establishing our notations.

The number of elements in a finite set X is denoted $\#X$. Let $\mathbb{N}$ be the set of natural numbers $\{0, 1, 2, \dots\}$. We use $[i, j]$ for denoting the segment of natural numbers $\{i, i+1, i+2, \dots, j\}$, while the segment $[0, j]$ is simply denoted $[j]$. Let A be a finite alphabet. We consider finite *words* as mappings $u\colon [n-1] \to A$ and denote the length of u by $|u|$, that is, $u = u(0)\,u(1)\,u(2)\,\dots\,u(|u|-2)\,u(|u|-1)$. An empty word is denoted Λ. We also deal with *sequences* over this alphabet, i.e., mappings $x\colon \mathbb{N} \to A$, and denote the set of these sequences by $A^{\mathbb{N}}$. A word of the form $x[0, i]$ for some i is called a *prefix* of x, and respectively a sequence of the form $x(i)x(i+1)x(i+2)\dots$ for some i is called a *suffix* of x. A *left* *shift* L maps a sequence to the same sequence with the first symbol cut, that is, $Lx = L(x(0)x(1)x(2)\dots) = x(1)x(2)\dots$.

A sequence x is *periodic* if for some $T > 0$ we have $x(i) = x(i+T)$ for each $i \in \mathbb{N}$. This T is called a period of x. A sequence is *eventually periodic* if some of its suffixes is periodic.

Let A, B be finite alphabets. A mapping $\phi\colon A^* \to B^*$ is called a *morphism* if $\phi(uv) = \phi(u)\phi(v)$ for all $u, v \in A^*$. Obviously, a morphism is determined by its values on single-letter words. A morphism is *k-uniform* if $|\phi(a)| = k$ for each $a \in A$. A 1-uniform morphism is called a *coding*. For $x \in A^{\mathbb{N}}$ denote $\phi(x) = \phi(x(0))\phi(x(1))\phi(x(2))\dots$ Further we consider only morphisms of the form $A^* \to A^*$. Let $\phi(t) = tu$ for some $t \in A$, $u \in A^*$. Then for all natural $m < n$ the word $\phi^n(t)$ begins with the word $\phi^m(t)$, so $\phi^\infty(t) = \lim_{n\to\infty} \phi^n(t) = tu\phi(u)\phi^2(u)\phi^3(u)\dots$ is correctly defined. If $\phi^n(u) \neq \Lambda$ for all n, then $\phi^\infty(t)$ is infinite. In this case ϕ is said to be *prolongable* on t. Sequences of the form $h(\phi^\infty(t))$ for a coding $h\colon A \to B$ are called *morphic*, of the form $\phi^\infty(t)$ are called *pure morphic*.

Unless stated otherwise, usually in this paper we assume $A = \{0, \dots, k-1\}$ for some $k \in \mathbb{N}$ and assume usual operations $+$ and $\cdot$ in A modulo k. We also assume $t = 0 \in A$, that is, we usually iterate a morphism on symbol 0 to obtain a sequence.

The class of morphic sequences of the form $h(\phi^\infty(t))$ with ϕ being k-uniform coincides with the class of so-called *k-automatic* sequences. Sequences that are k-automatic for some k, are called simply *automatic* (this class was introduced in [6] under the name of uniform tag sequences and was widely studied afterwards, see [4]).

The famous *Thue–Morse sequence* $\mathbf{t} = 0110100110010110100101 10\dots$ is the automatic sequence generated by the morphism $0 \to 01$, $1 \to 10$. This sequence can also be defined using conditions $\mathbf{t}(0) = 0$ and $\mathbf{t}(2n) = \mathbf{t}(n)$ and $\mathbf{t}(2n+1) = 1 - \mathbf{t}(n)$ for every n. The Thue–Morse sequence has several other names (due to other researchers who discovered it independently), a lot of interesting properties and appears in a lot of contexts; for a survey on the Thue–Morse sequence see [2] and also [4,11].

Another famous class of sequences is that of *Sturmian sequences*. They were introduced in [15] and have been widely studied since that time, e.g., see [11]. Sturmian sequences have many different equivalent definitions, and we will use the following, via (lower) mechanical sequences. For real numbers α and ρ with $0 < \alpha < 1$, α irrational, and $0 \leqslant \rho < 1$, let $c_{\alpha,\rho}(n) = \lfloor \alpha(n+1) + \rho \rfloor - \lfloor \alpha n + \rho \rfloor$ be the Sturmian sequence with parameters α and ρ. Here $\lfloor y \rfloor$ for a real number y is the maximal integer not greater than y. Let $\mathrm{fr}(y) = y - \lfloor y \rfloor$ for a real y to be its fractional part. Note that $c_{\alpha,\rho}(n) = 0$ if $0 \leqslant \mathrm{fr}(\alpha n + \rho) < 1 - \alpha$ and $c_{\alpha,\rho}(n) = 1$ if $1 - \alpha \leqslant \mathrm{fr}(\alpha n + \rho) < 1$. For example, the well-known *Fibonacci sequence* $\mathbf{f} = 010010100100101001010\dots$ that can be obtained as $\phi^\infty(0)$ for $\phi(0) = 01$, $\phi(1) = 0$, is the Sturmian sequence $\mathbf{f} = c_{1/\gamma^2, 1/\gamma^2}$ where $\gamma = \frac{1+\sqrt{5}}{2}$ is the golden ratio.

3 Aperiodicity Measure

The Besicovitch distance between sequences x and y is defined as $d(x, y) = \liminf_{n\to\infty} \frac{1}{n}\#\{i : i \in [n-1], x(i) \neq y(i)\}$. Then define the *aperiodicity measure* of a sequence x to be $\mathrm{AM}(x) = \inf\{d(x, L^n x) : n \geqslant 1\}$. In other words, $\mathrm{AM}(x)$ is the maximum number between 0 and 1 such that x differs from every non-trivial shift of x in at least $\mathrm{AM}(x)$ fraction of symbols. Let $s_n = \limsup_{m\to\infty} \frac{1}{m}\#\{i : i \in [m-1], x(i) = x(i+n)\}$. Then $\mathrm{AM}(x) = 1 - \sup\{s_n : n \geqslant 1\}$.

Clearly if a sequence x is eventually periodic with a period p then $s_p = 1$ and therefore $\mathrm{AM}(x) = 0$. The opposite is not necessarily true, though if $\mathrm{AM}(x)$ is relatively small then it is reasonable to think of x as close to a periodic sequence. We prove that $\mathrm{AM}(x) = 0$ for the Sturmian sequences x (Theorem 4) that are often considered as close to periodic. Note that if $\mathrm{AM}(x) > \alpha$ for a sequence x, then x has Besicovitch distance at least $\alpha/2$ with every eventually periodic sequence. Indeed, suppose $d(x, y) < \alpha/2$ for an eventually periodic sequence y with a period p. Then $d(L^p x, x) \leqslant d(L^p x, L^p y) + d(L^p y, y) + d(y, x) = d(x, y) + 0 + d(y, x) < \alpha$, since the Besicovitch distance is symmetric and satisfies the triangle inequality.

In Section 4 we prove that there exist automatic sequences with aperiodicity measure arbitrarily close to 1. However, if we do not require a sequence to be automatic, then the sequence with aperiodicity measure arbitrarily close to 1 can easily be proved to exist.

Theorem 1. *For every $\alpha < 1$ there exists a sequence x such that $\mathrm{AM}(x) \geqslant \alpha$.*

Proof. Let x be a sequence over an alphabet with k symbols. It is not difficult to prove that x can be chosen so that for every n the fraction of i's such that $x(i) = x(i+n)$, exists and is equal to $\frac{1}{k}$. Indeed, for every fixed n the Lebesgue measure (should not be mixed with aperiodicity measure!) of sequences not satisfying the above condition, is 0. Therefore, total Lebesgue measure of "bad" sequences is 0.

Let us choose such x. Then $s_n = \frac{1}{k}$ for every n. Therefore $\mathrm{AM}(x) = 1 - \frac{1}{k}$. □

However, for every fixed alphabet the aperiodicity measure can be bounded from above by some number strictly less than 1.

Theorem 2. *If a sequence x has no more than k symbols, then* $\mathrm{AM}(x) \leqslant 1 - \frac{1}{2k}$.

Proof. Suppose the alphabet of the sequence is $A = \{0, \ldots, k-1\}$.

The first observation we can make is that among any $k+1$ consecutive symbols of x there is one that occurs twice, by pigeonhole principle. After proper calculation similar to what we do below, one gets $\mathrm{AM}(x) \leqslant 1 - \frac{1}{k^2}$.

Generalizing this idea, let $u = x[l, l+N]$ be some segment of x, and for $0 \leqslant j \leqslant k-1$ denote by r_j how many times symbol j occurs in u. We have $\sum_{j=0}^{k-1} r_j = N + 1$. For a symbol $j \in A$, there are $\frac{r_j(r_j-1)}{2}$ pairs (p, q) with $p, q \in [l, l+N]$, $p < q$, such that $x(p) = x(q) = j$. Therefore $\#\{(p, q) : p, q \in [l, l+N], p < q, x(p) = x(q)\} = \sum_{j=0}^{k-1} \frac{1}{2} r_j(r_j - 1) = \sum_{j=0}^{k-1} \frac{1}{2} r_j^2 - \frac{N+1}{2} \geqslant \frac{1}{2k}(N+1)^2 - \frac{1}{2}(N+1)$, where we used the Cauchy's inequality $\sum_{j=0}^{k-1} r_j^2 \geqslant \frac{1}{k}\left(\sum_{j=0}^{k-1} r_j\right)^2$.

Now let us approximate $\sum_{n=1}^{N} s_n$:

$$\begin{aligned}
&\sum_{n=1}^{N} \frac{1}{m}\#\{i \in [m-1] : x(i) = x(i+n)\} \\
&\quad = \frac{1}{m}\#\{(i,n) : i \in [m-1], n \in [1,N], x(i) = x(i+n)\} \\
&\quad \geqslant \sum_{t=0}^{\lfloor m/N \rfloor -1} \frac{1}{m}\#\{(i,n) : i \in [Nt, N(t+1)-1], n \in [1,N], x(i) = x(i+n)\} \\
&\quad \geqslant \sum_{t=0}^{\lfloor m/N \rfloor -1} \frac{1}{m}\#\{(p,q) : p,q \in [Nt, N(t+1)],\, p<q, x(p) = x(q)\} \\
&\quad\quad \geqslant \frac{\lfloor m/N \rfloor}{m}\left(\frac{(N+1)^2}{2k} - \frac{N+1}{2}\right) \to \frac{(N+1)^2}{2kN} - \frac{N+1}{2N}
\end{aligned}$$

as $m \to \infty$. Thus $\sum_{n=1}^{N} s_n \geqslant \frac{(N+1)^2}{2kN} - \frac{N+1}{2N}$, and therefore $s_n \geqslant \frac{(N+1)^2}{2kN^2} - \frac{N+1}{2N^2}$ for some n such that $1 \leqslant n \leqslant N$. Tending $N \to \infty$, we can find s_n arbitrarily close to $\frac{1}{2k}$, therefore $\mathrm{AM}(x) \leqslant 1 - \frac{1}{2k}$. □

Note that in [9] the upper bound $1 - \frac{1}{k}$ was obtained for aperiodicity measure of sequences over k-letter alphabet which matches the lower bound from Theorem 1.

Now we compute the aperiodicity measure for some well known sequences.

Theorem 3. $\mathrm{AM}(\mathbf{t}) = \frac{1}{3}$.

Proof. Let $s_n^m = \frac{1}{m}\#\{i \in [m-1] : \mathbf{t}(i) = \mathbf{t}(i+n)\}$.

First of all, clearly $s_0^m = 1$ for every m. Then one can obtain the following equations for s_n^m:

$$\begin{aligned}
s_{2n}^{2m} &= s_n^m, \\
s_{2n}^{2m+1} &= \frac{m+1}{2m+1}s_n^{m+1} + \frac{m}{2m+1}s_n^m, \\
s_{2n+1}^{2m} &= 1 - \frac{1}{2}(s_n^m + s_{n+1}^m), \\
s_{2n+1}^{2m+1} &= \frac{m+1}{2m+1}(1 - s_n^{m+1}) + \frac{m}{2m+1}(1 - s_{n+1}^m)
\end{aligned} \tag{1}$$

for every m and n.

The idea is to consider separately even and odd indices of the sequence. Let us prove for instance the fourth equation. Indeed,

$$\begin{aligned}
s_{2n+1}^{2m+1} &= \frac{1}{2m+1}\#\{i \in [2m] : \mathbf{t}(i) = \mathbf{t}(i+2n+1)\} \\
&= \frac{1}{2m+1}\#\{i \in [m] : \mathbf{t}(2i) = \mathbf{t}(2i+2n+1)\} +
\end{aligned}$$

$$\begin{aligned}
&\frac{1}{2m+1}\#\{i \in [m-1] : \mathbf{t}(2i+1) = \mathbf{t}(2i+1+2n+1)\} \\
&= \frac{1}{2m+1}\#\{i \in [m] : \mathbf{t}(i) \neq \mathbf{t}(i+n)\} + \\
&\quad \frac{1}{2m+1}\#\{i \in [m-1] : \mathbf{t}(i) \neq \mathbf{t}(i+n+1)\} \\
&= \frac{m+1}{2m+1}(1 - s_n^{m+1}) + \frac{m}{2m+1}(1 - s_{n+1}^{m}),
\end{aligned}$$

where we used equations $\mathbf{t}(2i) = \mathbf{t}(i)$ and $\mathbf{t}(2i+1) = 1 - \mathbf{t}(i)$ for the Thue–Morse sequence. Other equations from (1) are proved in a similar way.

From (1) we derive

$$s_1^{2m} = \frac{1}{2} - \frac{1}{2}s_1^m, \qquad s_1^{2m+1} = \frac{m}{2m+1}(1 - s_1^m).$$

Note that $s_1^1 = 0$. Our goal is to prove that $\lim_{m\to\infty} s_1^m = \frac{1}{3}$. Let $s_1^m = \frac{1}{3} + a_m$. Then we have $a_1 = -\frac{1}{3}$, $a_{2m} = -\frac{1}{2}a_m$, and $a_{2m+1} = -\frac{m}{2m+1}a_m - \frac{1}{6m+3}$. Let $b_m = 3ma_m$. Then $b_1 = -1$, $b_{2m} = 6ma_{2m} = -3ma_m = -b_m$ and $b_{2m+1} = 3(2m+1)a_{2m+1} = -3ma_m - 1 = -b_m - 1$, from what it can easily be seen that $|b_m| = O(\log m)$, and therefore $\lim_{m\to\infty} a_m = 0$ and there exists $\lim_{m\to\infty} s_1^m = s_1 = \frac{1}{3}$.

Now from (1) one can prove that $s_n = \lim_{m\to\infty} s_n^m = \lim_{m\to\infty} \frac{1}{m}\#\{i \in [m-1] : \mathbf{t}(i) = \mathbf{t}(i+n)\}$ exists for every $n \geqslant 2$, and moreover one gets

$$s_{2n} = s_n, \qquad s_{2n+1} = 1 - \frac{1}{2}(s_n + s_{n+1})$$

for every n, and $s_0 = 1$, $s_1 = \frac{1}{3}$.

Now it is easy to see by induction that $\frac{1}{3} \leqslant s_n \leqslant \frac{2}{3}$ for every $n \geqslant 1$. And since $s_3 = \frac{2}{3}$, then $\mathrm{AM}(\mathbf{t}) = 1 - \frac{2}{3} = \frac{1}{3}$. □

Note that most part of the proof of Theorem 3 was spent on proving the existence of limits $s_n = \lim_{m\to\infty} s_n^m$. If one is ready to assume that these limits exist, then the proof becomes much simpler and shorter. Though we do not know how to prove the existence of these limits simpler, we will be omitting such proofs later, since they are all similar to each other and rather technical.

Theorem 4. *If x is Sturmian, then* $\mathrm{AM}(x) = 0$.

Proof. Let $x = c_{\alpha,\rho}$ be a Sturmian sequence. Recall that by definition $0 < \alpha < 1$, α irrational, and $0 \leqslant \rho < 1$.

Our goal is to show that s_n can be arbitrarily close to 1. Then from the definition of aperiodicity measure it follows that $\mathrm{AM}(x) = 0$.

Let $\varepsilon > 0$. Since α is irrational, we can find n such that $\mathrm{fr}(n\alpha) < \varepsilon$. We have $\frac{1}{m}\#\{i \in [m-1] : x(i) = x(i+n)\} = 1 - \frac{1}{m}\#\{i \in [m-1] : x(i) \neq x(i+n)\}$. Recall that $x(j) = 0$ if $0 \leqslant \mathrm{fr}(\alpha j + \rho) < 1 - \alpha$ and $x(j) = 1$ if $1 - \alpha \leqslant \mathrm{fr}(\alpha j + \rho) < 1$ for every j. Note also that $\mathrm{fr}(\alpha(i+n) + \rho) = \mathrm{fr}(\mathrm{fr}(\alpha i + \rho) + \mathrm{fr}(\alpha n))$. Therefore

$$\begin{aligned}&\{i \in [m-1] : x(i) \neq x(i+n)\}\\ &\subseteq \{i \in [m-1] : 1-\alpha-\varepsilon \leqslant \mathrm{fr}(\alpha i+\rho) < 1-\alpha\}\\ &\quad \cup \{i \in [m-1] : 1-\varepsilon \leqslant \mathrm{fr}(\alpha i+\rho) < 1\}.\end{aligned}$$

Therefore, $\#\{i \in [m-1] : x(i) \neq x(i+n)\} \leqslant \#\{i \in [m-1] : 1-\alpha-\varepsilon \leqslant \mathrm{fr}(\alpha i+\rho) < 1-\alpha\} + \#\{i \in [m-1] : 1-\varepsilon \leqslant \mathrm{fr}(\alpha i+\rho) < 1\}$ which is asymptotically $< 2\varepsilon m$ as $m \to \infty$. Indeed, it is well known that for every irrational β, every real γ, every real a, b such that $0 \leqslant a < b \leqslant 1$, we have $\lim_{m\to\infty} \frac{1}{m}\#\{i \in [m-1] : a \leqslant \mathrm{fr}(\beta i+\gamma) \leqslant b\} = b-a$, that is, the sequence $(\mathrm{fr}(\beta i+\gamma))_{i=0}^{\infty}$ is uniformly distributed in $[0,1]$ (the Kronecker–Weyl Theorem).

Thus $s_n > 1-2\varepsilon$. Since ε can be chosen arbitrarily small, it follows that $\mathrm{AM}(x) = 0$. □

4 Automatic Sequences with High Aperiodicity Measure

The following generalization of the Thue–Morse sequence was called Prouhet sequences in [1] (see [17]) and has been widely studied (e.g., see [3,19] etc.).

Let $\phi\colon \{0,\ldots,k-1\}^* \to \{0,\ldots,k-1\}^*$ be as follows:

$$\begin{aligned}\phi(0) &= 0\,1\,2\,3\ldots(k-2)\,(k-1)\\ \phi(1) &= 1\,2\,3\ldots(k-2)\,(k-1)\,0\\ \phi(2) &= 2\,3\ldots(k-2)\,(k-1)\,0\,1\\ &\ldots\\ \phi(k-1) &= (k-1)\,0\,1\,2\,3\ldots(k-2),\end{aligned}$$

in other words, $(\phi(i))(j) = i+j$ (where $+$ is modulo k) for $0 \leqslant i,j \leqslant k-1$. Let $\mathbf{t}_k = \phi^{\infty}(0)$. Initially it was conjectured that $\mathbf{t}_k$ may have high aperiodicity measure. However, it turns out to be not the case.

Theorem 5. $\mathrm{AM}(\mathbf{t}_k) \leqslant \frac{2}{k+1} - \frac{2}{k^{k-1}(k+1)}$.

Proof (sketch). Remind that $s_n = \limsup_{m\to\infty} \frac{1}{m}\#\{i \in [m-1] : x(i) = x(i+n)\}$. Let us generalize this and define $s_n(d) = \limsup_{m\to\infty} \frac{1}{m}\#\{i \in [m-1] : x(i+n) - x(i) = d\}$. That is, $s_n(0) = s_n$.

It is clear that $s_0(0) = 1$ and $s_0(d) = 0$ for $1 \leqslant d \leqslant k-1$.

In the same manner as what we did in the proof of Theorem 3, one can prove the existence of limits $s_n(d) = \lim_{m\to\infty} \frac{1}{m}\#\{i \in [m-1] : x(i+n) - x(i) = d\}$ and obtain the following equations:

$$s_{kn+p}(d) = \frac{k-p}{k} s_n(d-p) + \frac{p}{k} s_{n+1}(d-p) \qquad (2)$$

for every n and every d, p such that $0 \leqslant d, p \leqslant k-1$.

In particular, one can derive the following equations $s_1(d) = \frac{k-1}{k} s_0(d-1) + \frac{1}{k} s_1(d-1)$ for $0 \leqslant d \leqslant k-1$ and prove that

$$s_1(0) = \frac{k-1}{k^k-1} \text{ and } s_1(d) = k^{k-d}\frac{k-1}{k^k-1}$$

for $1 \leqslant d \leqslant k-1$.

One can prove for $0 \leqslant i \leqslant k$ by induction on i that

$$s_{k^i-1}(k-i) = \frac{1}{k^i}\left(1 + \frac{k-1}{k+1}\frac{k^{2i}-1}{k^k-1}\right)$$

using $s_0(0) = 1$ and $s_{k^{i+1}-1}(k-(i+1)) = \frac{1}{k}s_{k^i-1}(k-i) + \frac{k-1}{k}s_{k^i}(k-i) = \frac{1}{k}s_{k^i-1}(k-i) + \frac{k-1}{k}k^i\frac{k-1}{k^k-1}$ (follows from equations (2)). In particular,

$$s_{k^k-1}(0) = \frac{1}{k^k}\left(1 + \frac{k-1}{k+1}\frac{k^{2k}-1}{k^k-1}\right) = 1 - \frac{2}{k+1} + \frac{2}{k^{k-1}(k+1)},$$

from what it follows that $\mathrm{AM}(\mathbf{t}_k) \leqslant \frac{2}{k+1} - \frac{2}{k^{k-1}(k+1)}$. □

We believe that $\mathrm{AM}(\mathbf{t}_k) = \frac{2}{k+1} - \frac{2}{k^{k-1}(k+1)}$ though did not manage to show this. To prove this, one has to find the maximum of the above sequence $s_n(0)$, and we believe that this maximum is indeed reached in $n = k^k - 1$. This statement is supported by computer tests we performed.

An additional interest to study sequences $\mathbf{t}_k$ is in the following alternative definition for these sequences. Let $f_k(i)$ be the sum of digits of i written in base k. Then $\mathbf{t}_k(i) \equiv f_k(i) \pmod{k}$. This representation is well known for the Thue–Morse sequence. One may ask the following question: what is the number n such that the fraction of numbers i for which $f_k(i) \equiv f_k(i+n) \pmod{k}$ is maximum possible? We conjecture that this n is $k^k - 1$, that is, the number consisting of $k-1$ digits $k-1$ in base k, and this maximum possible fraction is $1 - \frac{2}{k+1} + \frac{2}{k^{k-1}(k+1)}$, that is, approximately $1 - \frac{2}{k+1}$ for large k.

Other interesting regularities we noticed while performing some computer tests, are the following. Let $s_n^{(k)}(d)$ in this paragraph be the value of $s_n(d)$ for $\mathbf{t}_k$. Let $\operatorname{argmax}_n f(n)$ for $f\colon \mathbb{N} \to \mathbb{R}$ be the smallest value of the argument n on which $f(n)$ reaches its maximum. It seems that $\operatorname{argmax}_n s_n^{(k)}(0) = k^k - 1$ (see above), $\operatorname{argmax}_n s_n^{(k)}(-1) = 1$, $\operatorname{argmax}_n s_n^{(k)}(-2) = k^{k-1} + 1$. It seems also for instance that $\operatorname{argmax}_n s_n^{(4)}(1) = 3332333_4$ (here lower index k means base k representation), and $\operatorname{argmax}_n s_n^{(5)}(1) = 444434444_5$. It also seems that $\operatorname{argmax}_n s_n^{(5)}(2) = 100010001_5$. Sequences $\mathbf{t}_k$ and the aforementioned regularities should definitely be studied more properly, especially keeping in mind the alternative definition from the previous paragraph.

Now we construct automatic sequences with the aperiodicity measure arbitrarily close to 1.

Theorem 6. *For every $\alpha < 1$ there exists an automatic sequence x such that $\mathrm{AM}(x) \geqslant \alpha$.*

Proof (sketch). Let $k \geqslant 3$ and let $\phi\colon \{0,\dots,k-1\}^* \to \{0,\dots,k-1\}^*$ be such that $(\phi(i))(j) = i + 1 + 2 + \cdots + (j-1) + j$ (where $+$ is always modulo k) for $0 \leqslant i, j \leqslant k-1$. Let $x_k = \phi^\infty(0)$. For instance, if $k = 5$, then ϕ is as follows:

$$\begin{aligned}\phi(0) &= 01310\\ \phi(1) &= 12421\\ \phi(2) &= 23032\\ \phi(3) &= 34143\\ \phi(4) &= 40204,\end{aligned}$$

and $x_5 = 0131012421341431242101310124 21\ldots$

Claim. If $k \geqslant 3$ is prime, then $\mathrm{AM}(x_k) = 1 - \frac{2}{k}$.

Let us define $s_n^m(d) = \frac{1}{m}\#\{i \in [m-1] : x_k(i+n) - x_k(i) = d\}$.

It follows from the definition that for every m we have $s_0^m(0) = 1$ and $s_0^m(d) = 0$ for $1 \leqslant d \leqslant k-1$. Analogously to the proof of Theorem 3, for every n, every $m \geqslant 1$, and every d, p, t such that $0 \leqslant d, p, t \leqslant k-1$, one can obtain the following equations (compare with equations (1)):

$$\begin{aligned} &s_{kn+p}^{km+t}(d)\\ &= \frac{1}{km+t}\sum_{j=0}^{k-p-1} m_j s_n^{m_j}\left(d - \frac{p(p+1)}{2} - jp\right)\\ &+ \frac{1}{km+t}\sum_{j=k-p}^{k-1} m_j s_{n+1}^{m_j}\left(d - \frac{p(p+1)}{2} - jp\right), \end{aligned} \tag{3}$$

where $m_j = m+1$ for $j < t$ and $m_j = m$ for $j \geqslant t$. The idea again is to consider separately sets of indices $\{ik + j : i \in \mathbb{N}\}$ for different j such that $0 \leqslant j \leqslant k-1$.

In particular, one can derive from (3) that $s_1^{km+t}(0) = \frac{m}{km+t} s_1^m(0)$ and $s_1^{km+t}(d) = \frac{1}{km+t}\left(m_{d-1} + m s_1^m(d)\right)$ for $1 \leqslant d \leqslant k-1$. Our goal is to prove that there exist $\lim_{m\to\infty} s_1^m(0) = 0$ and $\lim_{m\to\infty} s_1^m(d) = \frac{1}{k-1}$ for $1 \leqslant d \leqslant k-1$.

The former equation is clear, since $s_1^t(0) = 0$ for $1 \leqslant t \leqslant k-1$ can be checked easily. For the latter, fix some d such that $1 \leqslant d \leqslant k-1$ and let b_m be such that $s_1^m(d) = \frac{1}{k-1} + \frac{b_m}{m}$. Then one gets $b_{km+t} = m_{d-1} - m - \frac{t}{k-1} + b_m$, from what it is easy to see that $|b_m| = O(\log m)$. Therefore there exists $\lim_{m\to\infty} s_1^m(d) = s_1(d) = \frac{1}{k-1}$.

Using (3), now one can prove by induction on n the existence of limits $s_n(d) = \lim_{m\to\infty} \frac{1}{m}\#\{i \in [m-1] : x(i+n) - x(i) = d\}$ and to obtain the following equations:

$$s_{kn+p}(d) = \frac{1}{k}\left(\sum_{j=0}^{k-p-1} s_n\left(d - \frac{p(p+1)}{2} - jp\right) + \sum_{j=k-p}^{k-1} s_{n+1}\left(d - \frac{p(p+1)}{2} - jp\right)\right) \tag{4}$$

for every n and every d, p such that $0 \leqslant d, p \leqslant k-1$.

For instance, for $k = 5$ we get

$$s_{5m}(d) = s_m(d)$$
$$s_{5m+1}(d) = \frac{1}{5}(s_m(d-1) + s_m(d-2) + s_m(d-3) + s_m(d-4) + s_{m+1}(d-5))$$
$$s_{5m+2}(d) = \frac{1}{5}(s_m(d-3) + s_m(d-5) + s_m(d-7) + s_{m+1}(d-9) + s_{m+1}(d-11))$$
$$\dots$$

Clearly, $s_0(0) = 1$ and $s_0(d) = 0$ for $1 \leqslant d \leqslant k-1$. We already proved that $s_1(0) = 0$ and $s_1(d) = \frac{1}{k-1}$ for $1 \leqslant d \leqslant k-1$.

Using (4), it is easy to see that $s_{k-1}(0) = \frac{1}{k}s_0(0) + \frac{1}{k}\sum_{j=1}^{k-1} s_1(j) = \frac{2}{k}$.

Now it is easy to prove using equations (4) that $s_n(d) \leqslant \frac{2}{k}$ for every d and $n \geqslant 1$. Indeed, note that k is prime (this is the first time we use it), and therefore $\{d - \frac{p(p+1)}{2} - p, d - \frac{p(p+1)}{2} - 2p, \dots, d - \frac{p(p+1)}{2} - kp\} = \{0, \dots, k-1\}$. Thus for $n \geqslant 0$ and $1 \leqslant p \leqslant k-1$ we have

$$\begin{aligned} &s_{kn+p}(d) \\ &= \frac{1}{k}\left(\sum_{j=0}^{k-p-1} s_n\left(d - \frac{p(p+1)}{2} - jp\right) + \sum_{j=k-p}^{k-1} s_{n+1}\left(d - \frac{p(p+1)}{2} - jp\right)\right) \\ &\leqslant \frac{1}{k}\sum_{j=0}^{k-1} s_n(j) + \frac{1}{k}\sum_{j=0}^{k-1} s_{n+1}(j) = \frac{1}{k} + \frac{1}{k} = \frac{2}{k}, \end{aligned}$$

and we also need $s_{kn}(d) = s_n(d)$ for $n \geqslant 1$.

Therefore $\mathrm{AM}(x_k) = 1 - \sup\{s_n(0) : n \geqslant 1\} = 1 - s_{k-1}(0) = 1 - \frac{2}{k}$. □

5 Conclusion and Open Problems

In this paper we introduced the notion of aperiodicity measure for infinite symbolic sequences. It seems that this notion was not studied before, though looks very natural at least from a combinatorial point of view. However, the results of our paper are far from sufficient before we could say that the notion of aperiodicity measure is properly studied. Here we formulate some open questions, in addition to those listed throughout the paper, that we think may be interesting for future research.

1. As we already discussed, $\mathrm{AM}(x)$ of a sequence x over the alphabet with k symbols ranges from 0 to $1 - \frac{1}{k}$. What values in this range may the aperiodicity measure have?
2. For each $k \geqslant 2$, what is the maximum possible $\mathrm{AM}(x)$ for an automatic sequence x over the alphabet of k symbols? For a morphic sequence? What values can the aperiodicity measure of a morphic sequence have?
3. For each $k \geqslant 2$, what is the maximum possible $\mathrm{AM}(x)$ for a k-automatic x? What values can the aperiodicity measure of a k-automatic sequence have?

4. For which sequences x can one take lim instead of lim sup in the definition of $s_n(x) = \limsup \frac{1}{m}\#\{i \in [m-1] : x(i) = x(i+n)\}$, that is, when does this limit exist? In particular, does it exist for every automatic sequence x? Is it true that if this limit exists for $n = 1$, then it exists for all n?
5. Study the behavior of the sequence s_n, and more generally, of the sequence $s_n(i)$ for different i, more properly. In particular, describe its set of accumulation points.
6. Calculate the aperiodicity measure for some other sequences and classes of sequences, for instance, for the Toeplitz sequences, some morphic sequences, some generalizations of the Sturmian sequences, etc.
7. We characterized the aperiodicity measure of some sequences and suggested that this work should be continued. However, one can also ask the inverse question: to characterize the set of sequences with some fixed aperiodicity measure α. This is specifically interesting for the extremal values $\alpha = 0$ and $\alpha = 1 - \frac{1}{k}$ for k-letter sequences.
8. There is a generic way (an algorithm) to calculate the aperiodicity measure for morphic sequences. For example, this can be seen in the following way. Let the Cartesian product of sequences x and y be the sequence $x \times y$ such that $(x \times y)(n) = (x(n), y(n))$. Then for a morphic x, the sequence $x \times L^p x$ is morhic, since it can be obtained from x by a finite transduction (e.g., see [4]). Then it only remains to calculate the frequency of letters in $x \times L^p x$ (e.g., see [18]). However, this method is clearly non-practical. Is there a sufficiently simpler method to calculate the aperiodicity measure for morphic sequences? Or at least for automatic sequences? In particular, can one generalize the method used in the proofs of Theorems 3, 5, 6?

Acknowledgements

The authors are grateful to A. Semenov and N. Vereshchagin for their permanent support, and also to S. Avgustinovich, B. Durand, A. Frid, A. Romashchenko, A. Shen for fruitful discussions, as well as to anonymous referees for many useful comments.

References

1. Adler, A., Li, S.-Y.R.: Magic cubes and Prouhet sequences. American Mathematical Monthly 84, 618–627 (1977)
2. Allouche, J.-P., Shallit, J.: The ubiquitous Prouhet–Thue–Morse sequence. In: Proceedings of Sequences and their applications, SETA 1998, pp. 1–16. Springer, Heidelberg (1999)
3. Allouche, J.-P., Shallit, J.: Sums of digits, overlaps, and palindromes. Discrete Mathematics and Theoretical Computer Science 4, 1–10 (2000)
4. Allouche, J.-P., Shallit, J.: Automatic Sequences: Theory, Applications, Generalizations. Cambridge University Press, Cambridge (2003)
5. Cassaigne, J., Mauduit, C., Sárközy, A.: On finite pseudorandom binary sequences. VII. The measures of pseudorandomness. Acta Arithmetica 103(2), 97–118 (2002)

6. Cobham, A.: Uniform tag sequences. Mathematical Systems Theory 6, 164–192 (1972)
7. Durand, B., Romashchenko, A., Shen, A.: Fixed point and aperiodic tilings. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 276–288. Springer, Heidelberg (2008), enhanced version arXiv:0802.2432, `http://arxiv.org/abs/0802.2432`
8. Ferenczi, S.: Tiling and local rank properties of the Morse sequence. Theoretical Computer Science 129, 369–383 (1994)
9. Grant, E., Shallit, J., Stoll, T.: Bounds for the discrete correlation of infinite sequences on k symbols and generalized Rudin–Shapiro sequences (2008); preprint arXiv:0812.3186
10. Krieger, D., Ochem, P., Rampersad, N., Shallit, J.: Avoiding approximate squares. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) DLT 2007. LNCS, vol. 4588, pp. 278–289. Springer, Heidelberg (2007)
11. Lothaire, M.: Algebraic Combinatorics on Words. Cambridge University Press, Cambridge (2002)
12. Mauduit, C., Sárközy, A.: On finite pseudorandom binary sequences. I. Measure of pseudorandomness, the Legendre symbol. Acta Arithmetica 82(4), 365–377 (1997)
13. Mauduit, C., Sárközy, A.: On the measures of pseudorandomness of binary sequences. Discrete Mathematics 271, 195–207 (2003)
14. Morse, M., Hedlund, G.A.: Symbolic dynamics. American Journal of Mathematics 60(4), 815–866 (1938)
15. Morse, M., Hedlund, G.A.: Symbolic dynamics ii: Sturmian trajectories. American Journal of Mathematics 62, 1–42 (1940)
16. Muchnik, A.A., Pritykin, Y.L., Semenov, A.L.: Sequences close to periodic (2009); preprint arXiv:0903.5316 (in Russian, to appear in English)
17. Prouhet, M.E.: Mémoire sur quelques relations entre les puissances des nombres. Comptes Rendus des Séances de l'Académie des Sciences 33, 225 (1851), `http://gallica.bnf.fr/ark:/12148/bpt6k29901.image.f227.langFR`
18. Saari, K.: On the frequency of letters in morphic sequences. In: Grigoriev, D., Harrison, J., Hirsch, E.A. (eds.) CSR 2006. LNCS, vol. 3967, pp. 334–345. Springer, Heidelberg (2006)
19. Séébold, P.: On some generalizations of the Thue–Morse morphism. Theoretical Computer Science 292, 283–298 (2003)

On the Complexity of Matroid Isomorphism Problems

Raghavendra Rao B.V.[1] and Jayalal Sarma M.N.[2,⋆]

[1] The Institute of Mathematical Sciences, C.I.T. Campus, Chennai 600 113, India
bvrr@imsc.res.in

[2] Institute for Theoretical Computer Science, Tsinghua University, Beijing 100 084, China
jayalal@tsinghua.edu.cn

Abstract. We study the complexity of testing if two given matroids are isomorphic. The problem is easily seen to be in Σ_2^p. In the case of linear matroids, which are represented over polynomially growing fields, we note that the problem is unlikely to be Σ_2^p-complete and is coNP-hard. We show that when the rank of the matroid is bounded by a constant, linear matroid isomorphism and matroid isomorphism are both polynomial time many-one equivalent to graph isomorphism.

We give a polynomial time Turing reduction from graphic matroid isomorphism problem to the graph isomorphism problem. We then give a polynomial time many-one reduction from bounded rank matroid isomorphism problem to graphic matroid isomorphism, thus showing that all the above problems are polynomial time equivalent.

Further, for linear and graphic matroids, we prove that the automorphism problem is polynomial time equivalent to the corresponding isomorphism problems. In addition, we give a polynomial time membership test algorithm for the automorphism group of a graphic matroid.

1 Introduction

Isomorphism problems over various mathematical structures have been a source of intriguing problems in complexity theory (see [1]). The most important problem of this domain is the well-known graph isomorphism problem. Though the complexity characterisation of the general version of this problem is still unknown, there have been various interesting special cases of the problem which are known to have polynomial time algorithms [10,12] and many structural results are known [9,11,17]. In this paper we talk about isomorphism problem associated with matroids.

A matroid M is a combinatorial object defined over a finite set S (of size m) called the *ground set*, equipped with a non-empty family $\mathcal{I}$ of subsets of S (containing the empty subset) which is closed under taking of subsets and

⋆ This work was supported in part by the National Natural Science Foundation of China Grant 60553001, and the National Basic Research Program of China Grant 2007CB807900,2007CB807901.

satisfies the *exchange axiom* : for any $I_1, I_2 \in \mathcal{I}$ such that $|I_1| > |I_2|$, $\exists x \in I_1 \setminus I_2$, $I_2 \cup \{x\} \in \mathcal{I}$. The sets in $\mathcal{I}$ are called *independent sets.* The rank of the matroid is the size of the maximal independent set. This provides useful abstractions of many concepts in combinatorics and linear algebra and is well studied [15]. We study the problem of testing isomorphism between two given matroids.

Two matroids M_1 and M_2 are said to be isomorphic if there is a bijection between the elements of the ground set which maps independent sets to independent sets (or equivalently circuits to circuits, or bases to bases, see section 2). Quite naturally, the representation of the input matroids is important in deciding the complexity of the algorithmic problem.

There are several equivalent representations of a matroid. For example, enumerating the maximal independent sets (called bases) or the minimal dependent sets (called circuits) also defines the matroid. These representations, although can be exponential in the size of the ground set, indeed exist for every matroid, by definition. With this enumerative representation, Mayhew [13] studied the matroid isomorphism problem, and shows that the problem is equivalent to graph isomorphism problem. However, a natural question is whether the problem is difficult when the representation of the matroid is more implicit? In a black-box setting, one can also consider the input representation in the form of an oracle or a black-box, where the oracle answers whether a given set is independent or not.

More implicit (and efficient) representation of matroids have been studied. One natural way is to identify the given matroid with matroids defined over combinatorial or algebraic objects which have implicit descriptions. A general framework in this direction is the representation of a matroid over a field. A matroid $M = (S, \mathcal{I})$ of rank r is said to be *representable* over a field $\mathbb{F}$ if there is a map, $\phi : S \to \mathbb{F}^r$ such that, $\forall A \subseteq S$, $A \in \mathcal{I} \iff \phi(A)$ is linearly independent over $\mathbb{F}^r$ as a vector space. However, there are matroids which do not admit linear representations over any field. (For example, the Vamós Matroid, See Proposition 6.1.10, [15].). In contrast, there are matroids (called regular matroids) which admit linear representations over all fields.

Another natural representation for a matroid is over graphs. For any graph X, we can associate a matroid $M(X)$ as follows: the set of edges of X is the ground set, and the acyclic subgraphs of the given graph form the independent sets. A matroid M is called a *graphic matroid* (also called polygon matroid or cyclic matroid) if it is isomorphic to $M(X)$ for some graph X. It is known that graphic matroids are linear. Indeed, the incidence matrix of the graph will give a representation over $\mathbb{F}_2$. There are linear matroids which are not graphic. (See [15] for more details.)

The above definitions themselves highlight the importance of testing isomorphism between two given matroids. We study the isomorphism problem for the case of linear matroids (Linear Matroid Isomorphism problem (LMI) and graphic matroids (Graphic Matroid Isomorphism problem (GMI)) where the inputs are in the implicit representation (matrices and graphs resp.).

From a complexity perspective, the general case of the problem (where the matroid is given as an independent set oracle) is in Σ_2^p. However, it is not even clear a priori if the problem is in NP even in the above restricted cases where there are implicit representations. But we note that for the case of graphic matroids the problem admits an NP algorithm. Hence an intriguing question is about the comparison of this problem to the well studied graph isomorphism problem.

An important result in this direction, due to Whitney (see [19]), says that in the case of 3-connected graphs, the graphs are isomorphic if and only if the corresponding matroids are isomorphic (see section 5). Thus the problems of testing isomorphism of graphs and of the corresponding graphic matroids are equivalent for the case of 3-connected graphs. Despite this similarity between the problems, to the best of our knowledge, there has not been a systematic study of GMI and its relationships to graph isomorphism problem (GI). This immediately gives a motivation to study the isomorphism problem for 3-connected graphs. In particular, from the recent results on graph isomorphism problem for these classes of graphs [4], it follows that graphic matroid isomorphism problem for 3-connected planar graphs is L-complete.

In this context we study the general, linear and graphic matroid isomorphism problems. Our main contributions in the paper are as follows:

– We prove that when the rank of the matroid is bounded, linear matroid isomorphism and matroid isomorphism are both equivalent to GI (Theorem 2)[1]
– We develop tools to handle colouring of ground set elements in the context of the isomorphism problem. We show that coloured versions of linear matroid isomorphism and graphic matroid isomorphism are as hard as the general version (Lemma 2, 1). As an immediate application of this, we show that the automorphism problems for graphic matroids and linear matroids are polynomial time Turing equivalent to the corresponding isomorphism problems. In this context, we also give a polynomial time membership test algorithm for the automorphism group of a graphic matroid (Theorem 8).
– We give a polynomial time Turing reduction (Theorem 3) from graphic matroid isomorphism problem to the graph isomorphism problem by developing an edge colouring scheme which algorithmically uses a decomposition given by [7] (and [3]). Our reduction, in particular implies that the graphic matroid isomorphism testing for planar graphs can be done in deterministic polynomial time (Corollary 2).
– Finally, we give a reduction from bounded rank matroid isomorphism problem to graphic matroid isomorphism (Theorem 5), thus showing that all the above problems are poly-time equivalent.

Due to space limitations we have omitted many proofs. The omitted proofs can be found in the full version [16].

[1] We note that, although not explicitly stated, the equivalence of bounded rank matroid isomorphism and and graph isomorphism also follows from the results of Mayhew [13]. However, it is not immediately clear if the GI-hard instances of [13] are linearly representable. Our proof is different and extend this to linear matroids.

2 Notations and Preliminaries

All the complexity classes used here are standard and we refer the reader to any standard text book (for e.g. see [5]).

An isomorphism between two matroids M_1 and M_2 is a bijection $\phi : S_1 \to S_2$ such that $\forall C \subseteq S_1 : C \in \mathcal{C}_1 \iff \phi(C) \in \mathcal{C}_2$, where $\mathcal{C}_1$ and $\mathcal{C}_2$ are the family of circuits of the matroids M_1 and M_2 respectively. It is clear that for two matroids to be isomorphic the ground set has to be of the same size (say m) and they have to be of the same rank (say r). Now we state the computational problems more precisely.

*Problem 1 (*MATROID ISOMORPHISM(MI)*).* Given two matroids M_1 and M_2 as their ground sets and the independent set oracles, test if $M_1 \cong M_2$.

Given a matrix $A_{n\times m}$ over a field $\mathbb{F}$, we can define a matroid $M[A]$ with columns of A as the ground set (of m elements) and linearly independent columns as the independent sets of $M[A]$. A matroid $\mathcal{M} = (E, \mathcal{I})$ of rank r ($\leq n$) is said to be representable over $\mathbb{F}$, if there exists a matrix $A \in \mathbb{F}^{r\times m}$ such that $\mathcal{M}$ is isomorphic to the matroid $M[A]$. *Linear matroids* are matroids representable over fields. We assume that the field on which the matroid is represented is also a part of the input as the table for both operations, and that the field has at least m elements and at most $poly(m)$ elements.

*Problem 2 (*LINEAR MATROID ISOMORPHISM(LMI)*).* Given two matrices A and B over a given field $\mathbb{F}$, test if $M[A] \cong M[B]$.

As mentioned in the introduction, given a graph $X = (V, E)$ ($|V| = n, |E| = m$), a classical way to associate a matroid $M(X)$ with X is to treat E as ground set elements, the bases of $M(X)$ are spanning forests of X. Equivalently circuits of $M(X)$ are simple cycles in X. A matroid $\mathcal{M}$ is called *graphic* iff $\exists X$ such that $\mathcal{M}$ is isomorphic to $M(X)$.

*Problem 3 (*GRAPHIC MATROID ISOMORPHISM(GMI)*).* Given two graphs X_1 and X_2, test if $M(X_1) \cong M(X_2)$?.

We denote by PMI, the version of GMI where the input graphs are planar. Another associated terminology in the literature is about 2-isomorphism. Two graphs X_1 and X_2 are said to be 2-*isomorphic* (denoted by $X_1 \cong_2 X_2$) if their corresponding graphic matroids are isomorphic. Thus the above problem asks to test if two given graphs are 2-isomorphic. In a rather surprising result, Whitney [20] came up with a combinatorial characterisation of 2-isomorphic graphs. See [15] for more details.

3 Linear Matroid Isomorphism

In this section we present some observations and results on LMI. Some of these follow easily from the techniques in the literature. We make them explicit in a form that is relevant to the problem that we are considering.

As a basic complexity bound, it is easy to see that MI $\in \Sigma_2^p$. Indeed, the algorithm will existentially guess a bijection $\sigma : S_1 \to S_2$ and universally verify if for every subset $C \subseteq S_1$, $C \in \mathcal{C}_1 \iff \sigma(C) \in \mathcal{C}_2$ using the independent set oracle. We first observe that using the arguments similar to that of [11] one can show,

Theorem 1. LMI $\in \Sigma_2^p$. *In addition,* LMI is Σ_2^{P}-hard $\implies \mathsf{PH} = \Sigma_3^{\mathsf{P}}$.

Using the results of [14] and noting that uniform matroids are representable, we have the following,

Proposition 1. LMI *is* coNP*-hard.*

The above proposition also holds when the representation is over infinite fields. In this case, the proposition also more directly follows from a result of Hlinený [6], where it is shown that the problem of testing if a spike (a special kind of matroids) represented by a matrix over $\mathbb{Q}$ is the free spike is coNP complete. He also derives a linear representation for spikes.

Now we look at bounded rank variants of the problem. We denote by LMI$_b$ (MI$_b$), the restriction of LMI (MI) for which the input matrices have rank bounded by b. In the following, we use the following construction due to Babai [2] to prove LMI$_b \equiv_m^p$ GI.

Given a graph $X = (V, E)$ and a $k \in [3, d]$, where d is the minimum vertex degree of X, define a matroid $M = St_k(X)$ of rank k with the ground set as E as follows: every subset of $k-1$ edges is independent in M and every subset of E with k edges is independent if and only if they do not share a common vertex. Babai proved that $Aut(X) \cong Aut(St_k(X))$ and also gave a linear representation for $St_k(X)$ (Lemma 2.1 in [2]) for all k in the above range. By tightening Babai's result, we obtain the following theorem, (See [16] for more details.)

Theorem 2. *For any constant* $b \geq 3$, LMI$_b \equiv_m^p$ GI.

The above reduction (LMI$_b \leq_m^p$ GI) works even if the matroids are not linear, provided they are given via an independent set oracle. This gives the following corollary.

Corollary 1. LMI$_b \equiv_m^p$ MI$_b \equiv_m^p$ GI.

4 Isomorphism Problem of Coloured Matroids

Vertex or edge colouring is a classical tool used extensively in proving various results in graph isomorphism problem. We develop similar techniques for matroid isomorphism problems too.

An edge-k-colouring of a graph $X = (V, E)$ is a function $f : E \to \{1, \ldots, k\}$. Given two graphs $X_1 = (V_1, E_1, f_1)$ and $X_2 = (V_2, E_2, f_2)$ with edge colourings, the Coloured-GMI asks for an isomorphism which preserves the colours of the edges. Not surprisingly, we can prove the following.

Lemma 1. Coloured-GMI *is* AC^0 *many-one reducible to* GMI.

Using linear algebraic constructions, which we defer to the full version due to shortage of space, we generalise the above construction to the case of linear matroid isomorphism. Coloured-LMI denotes the variant of LMI where the inputs are the linear matroids M_1 and M_2 along with colour functions $c_i : \{1, \ldots, m\} \to \mathbb{N}, i \in \{1, 2\}$. The problem is to test if there is an isomorphism between M_1 and M_2 which preserves the colours of the column indices. We have,

Lemma 2. Coloured-LMI *is* AC^0 *many-one reducible to* LMI.

5 Graphic Matroid Isomorphism

In this section we study GMI. Unlike in the case of the graph isomorphism problem, an NP upper bound is not so obvious for GMI. We start with the discussion of an NP upper bound for GMI.

Whitney gave an exact characterisation of when two graphs are 2-isomorphic, in terms of three operations; twisting, cleaving and identification. (see [15].) Note that it is sufficient to find 2-isomorphisms between 2-connected components of X_1 and X_2. In fact, any *matching* between the sets of 2-connected components whose edges connect 2-isomorphic components will serve the purpose. This is because, any 2-isomorphism preserves simple cycles, and any simple cycle of a graph is always within a 2-connected component. Hence we can assume that both the input graphs are 2-connected and in the case of 2-connected graphs, twist is the only possible operation.

The set of separating pairs does not change under a twist operation. Moreover, despite the fact that the twist operations need not commute, Truemper [18] proved : for any two 2-connected 2-isomorphic graphs X and Y (on n vertices), X can be transformed to graph X' isomorphic to Y through a sequence at most $n - 2$ twists.

Using this lemma we get an NP upper bound for GMI. Given two graphs, X_1 and X_2, the NP machine just guesses the sequence of $n - 2$ separating pairs which corresponding to the 2-isomorphism. For each pair, guess the cut w.r.t which the twist operation is to be done, and apply each of them in sequence to the graph X_1 to obtain a graph X_1'. Now ask if $X_1' \cong X_2'$. This gives an upper bound of $\exists.\mathrm{GI} \subseteq \mathsf{NP}$. Thus we have,

Proposition 2. GMI *is in* NP.

This can also be seen as an NP-reduction from GMI to GI. Now we will give a deterministic reduction from GMI to GI. Although, this does not improve the NP upper bound, it implies that GMI cannot be NP-hard unless PH collapses. This deterministic reduction, stated in the theorem 3 below, is the main result of the paper.

Theorem 3. GMI $\leq_T^p$ GI.

Let us first look into the case of 3-connected graphs. A *separating pair* is a pair of vertices whose deletion leaves the graph disconnected. A 3-connected graph is a connected graph which does not have any separating pairs. Whitney ([19]) proved the following equivalence,

Theorem 4 ([19]). *X_1 and X_2 be 3-connected graphs, $X_1 \cong_2 X_2 \Longleftrightarrow X_1 \cong X_2$.*

Before giving a formal proof of Theorem 3, we describe the idea roughly here:

Basic Idea. Let X_1 and X_2 be the given graphs. From the above discussion, we can assume that the given graph is 2-connected.

In [7], Hopcroft and Tarjan proved that every 2-connected graph can be decomposed uniquely into a tree of 3-connected components, bonds or polygons.[2]

Moreover, [7] showed that this decomposition can be computed in polynomial time. The idea is to then find the isomorphism classes of these 3-connected components using queries to GI (see theorem 4), and then colour the tree nodes with the corresponding isomorphism class, and then compute a coloured tree isomorphism between the two trees produced from the two graphs.

A first mind block is that these isomorphisms between the 3-connected components need not map separating pairs to separating pairs. We overcome this by colouring the separating pairs (in fact the edge between them), with a canonical label of the two sub trees which the corresponding edge connects. To support this, we observe the following. There may be many isomorphisms between two 3-connected components which preserves the colours of the separating pairs. However, the order in which the vertices are mapped within a separating pair is irrelevant, since any order will be canonical up to a twist operation with respect to the separating pair.

So with the new colouring, the isomorphism between 3-connected components maps a separating pair to a separating pair, if and only if the two pairs of sub trees are isomorphic. However, even if this is the case, the coloured sub trees need not be isomorphic. This creates a simultaneity problem of colouring of the 3-connected components and the tree nodes and thus a second mind block.

We overcome this by colouring again using the code for coloured sub trees, and then finding the new isomorphism classes between the 3-connected components. This process is iterated till the colours stabilise on the tree as well as on the individual separating pairs (since there are only linear number of 3-connected components). Once this is ensured, we can recover the 2-isomorphism of the original graph by weaving the isomorphism of the 3-connected components guided by the tree adjacency relationship. In addition, if two 3-connected components are indeed isomorphic in the correctly aligned way, the above colouring scheme, at any point, does not distinguish between them.

[2] Cunningham et al. [3] shows that any graphic matroid $M(X)$ is isomorphic to $M(X_1) \oplus M(X_2) \ldots \oplus M(X_k)/\{e_1, e_2, \ldots, e_k\}$, where $M(X_1), \ldots, M(X_k)$ are 3-connected components, bonds or polygons of $M(X)$ and $e_1, \ldots, e_k$ are the virtual edges. However, it is unclear if this can be turned into a reduction from GMI to GI using edge/vertex colouring.

Breaking into Tree of 3-connected components. We use the algorithm of Hopcroft and Tarjan [7] to compute the set of 3-connected components of a 2-connected graph in polynomial time. We will now describe some details of the algorithm which we will exploit.

Let $X(V, E)$ be a 2-connected graph. Let Y be a connected component of $X \setminus \{a, b\}$, where $\{a, b\}$ is a separating pair. X is an *excisable* component w.r.t $\{a, b\}$ if $X \setminus Y$ has at least 2 edges and is 2-connected. The operation of excising Y from X results in two graphs: $C_1 = X \setminus Y$ plus a *virtual edge* joining (a, b), and $C_2 =$ the induced subgraph on $X \cup \{a, b\}$ plus a *virtual edge* joining (a, b). This operation may introduce multiple edges.

The decomposition of X into its 3-connected components is achieved by the repeated application of the excising operation (we call the corresponding separating pairs as *excised pairs*) until all the resulting graphs are free of excisable components. This decomposition is represented by a graph G_X with the 3-connected components of X as its vertices and two components are adjacent in G_X if and only if they share a virtual edge. In the above explanation, the graph G_X need not be a tree as the components which share a separating pair will form a clique.

To make it a tree, [7] introduces another component corresponding to the virtual edges thus identifying all the virtual edges created in the same excising operation with each other.

Instead, we do a surgery on the original graph X and the graph G_X. We add an edge between all the *excised pairs* (excised while obtaining G_X) to get the graph X'. Notice that, following the same series of decomposition gives a new graph T_X which is the same as G_X except that the cliques are replaced by star centred at a newly introduced vertex (component) corresponding to the newly introduced excised edges in X'. The newly introduced edges form a 3-connected component themselves with one virtual edge corresponding to each edge of the clique they replace.

We list down the properties of the tree T_X for further reference. (1) For every node in $t \in T_X$, there is exactly one 3-connected component in X'. We denote this by c_t. (2) For every edge $e = (u, v) \in T_X$, there are exactly two virtual edges, one each in the 3-connected components c_u and c_v. We call these virtual edges as the *twin edges* of each other. (3) For any given graph X, T_X is unique up to isomorphism (since G_X is unique [7]). In addition, T_X can be obtained from G_X in polynomial time.

The following claim states that (we omit the proof) this surgery in the graphs does not affect the existence of 2-isomorphisms.

Claim. $X_1 \cong_2 X_2 \iff X_1' \cong_2 X_2'$.

Thus it is sufficient to give an algorithm to test if $X_1' \cong_2 X_2'$, which we describe as follows.

INPUT: 2-connected graphs X_1' and X_2' and tree of 3-connected components T_1 and T_2.

OUTPUT: YES if $X_1' \cong_2 X_2'$, and NO otherwise.

```
ALGORITHM:

Notation: CODE($T$) denotes the canonical label[3] for a tree $T$.

1. Initialise $T_1' = T_1$, $T_2' = T_2$.
2. REPEAT
   (a) Set $T_1 = T_1'$, $T_2 = T_2'$.
   (b) For each edge $e = (u, v) \in T_i$, $i \in \{1, 2\}$:
       Let $T_i(e, u)$ and $T_i(e, v)$ be subtrees of $T_i$ obtained by deleting the edge $e$,
       containing $u$ and $v$ respectively.
       Colour virtual edges corresponding to the separating pairs in the components
       $c_u$ and $c_v$ with the set $\{\text{CODE}(T_i(e, u)), \text{CODE}(T_i(e, v))\}$. From now on, $c_t$ de-
       notes the coloured 3-connected component corresponding to node $t \in T_1 \cup T_2$.
   (c) Let $S_1$ and $S_2$ be the set of coloured 3-connected components of $X_1'$ and $X_2'$
       and let $S = S_1 \cup S_2$. Using queries to GI (see Proposition 3) find out the
       isomorphism classes in $S$. Let $C_1, \ldots, C_q$ denote the isomorphism classes.
   (d) Colour each node $t \in T_i$, $i \in \{1, 2\}$, with colour $\ell$ if $c_t \in C_\ell$. (This gives two
       coloured trees $T_1'$ and $T_2'$.)
   UNTIL (CODE($T_i$) $\neq$ CODE($T_i'$), $\forall i \in \{1, 2\}$)
3. Check if $T_1' \cong T_2'$ preserving the colours. Answer YES if $T_1' \cong T_2'$, and NO otherwise.
```

First we prove that the algorithm terminates in linear number of iterations of the repeat-until loop. Let q_i denote the number of isomorphism classes of the set of the coloured 3-connected components after the i^{th} iteration. We claim that, if the termination condition is not satisfied, then $|q_i| > |q_{i-1}|$. To see this, suppose the termination is not satisfied. This means that the coloured tree T_1' is different from T_1. This can happen only when the colour of a 3-connected component c_v, $v \in T_1 \cup T_2$ changes. In addition, this can only increase the isomorphism classes. Thus $|q_i| > |q_{i-1}|$. Since q can be at most $2n$, this shows that the algorithm exits the loop after at most $2n$ steps.

Now we prove the correctness of the algorithm. We follow the notation described in the algorithm.

Lemma 3. $X_1' \cong_2 X_2'. \iff T_1' \cong T_2'.$

Proof. We give a proof sketch here.
($\Rightarrow$) This dirction is easy and we omit the proof.
($\Leftarrow$) First, we recall some definitions needed in the proof. A *centre* of a tree T is defined as a vertex v such that $\max_{u \in T} d(u, v)$ is minimised at v, where $d(u, v)$ is the number of edges in the unique path from u to v. It is known that every tree T has a centre consisting of a single vertex or a pair of adjacent vertices. The minimum achieved at the centre is called the *height* of the tree, denoted by $ht(T)$.

[3] When T is coloured, CODE(T) is the code of the tree obtained after attaching the necessary gadgets to the coloured nodes. Notice that even after colouring, the graph is still a tree. In addition, for any T, CODE(T) can be computed in P.

Claim. Let ψ be a colour preserving isomorphism between T_1' and T_2', and χ_t is an isomorphism between the 3-connected components c_t and $c_{\psi(t)}$. Then, $X_1' \cong_2 X_2'$ via a map σ such that $\forall t \in T_1'$, $\forall e \in c_t \cap E_1 : \sigma(e) = \chi_t(e)$ where E_1 is the set of edges in X_1'.

Proof. The proof is by induction on height of the trees $h = ht(T_1') = ht(T_2')$, where the height (and centre) is computed with respect to the underlying tree ignoring colours on the vertices. Base case is when $h = 0$; that is, T_1' and T_2' have just one node (3-connected component) without any virtual edges. Simply define $\sigma = \chi$. By Theorem 4, this gives the required 2-isomorphism. Suppose that if $h = ht(T_1') = ht(T_2') < k$, the above claim is true. For the induction step, suppose further that $T_1' \cong T_2'$ via ψ, and $ht(T_1') = ht(T_2') = k$. Notice that ψ should map the centre(s) of T_1 to that of T_2. We consider two cases.

In the first case, T_1' and T_2' have unique centres α and β. It is clear that $\psi(\alpha) = \beta$. Let c_1 and c_2 be the corresponding coloured (as in step 2b) 3-connected components. Therefore, there is a colour preserving isomorphism $\chi = \chi_\alpha$ between c_α and c_β. Let $f_1, \ldots f_k$ be the virtual edges in c_α corresponding to the tree edges $e_1 = (\alpha, v_1), \ldots, e_k = (\alpha, v_k)$ where $v_1, \ldots, v_k$ are neighbours of α in T_1'. Denote $\psi(e_i)$ by e_i', and $\psi(v_i)$ by v_i'.

Observe that only virtual edges are coloured in the 3-connected components in step 2b while determining their isomorphism classes. Therefore, for each i, $\chi(f_i)$ will be a virtual edge in c_β, and in addition, with the same colour as f_i. That is, $\{\textsc{code}(T_1(e_i, \alpha)), \textsc{code}(T_1(e_i, v_i))\} = \{\textsc{code}(T_2(e_i', \beta)), \textsc{code}(T_2(e_i', v_i')))\}$. Since α and β are the centres of T_1' and T_2', it must be the case that in the above set equality, $\textsc{code}(T_1(e_i, v_i)) = \textsc{code}(T_2(e_i', v_i'))$. From the termination condition of the algorithm, this implies that $\textsc{code}(T_1'(e_i, v_i)) = \textsc{code}(T_2'(e_i', v_i'))$. Hence, $T_1'(e_i, v_i) \cong T_2'(e_i', v_i')$. In addition, $ht(v_i) = ht(v_i') < k$. Let X_{f_i}' and $X_{\chi(f_i)}'$ denote the subgraphs of X_1' and X_2' corresponding to $T_1'(e_i, v_i)$ and $T_2'(e_i', v_i')$ respectively. By induction hypothesis, the graphs X_{f_i}' and $X_{\chi(f_i)}'$ are 2-isomorphic via σ_i which agrees with the corresponding χ_t for $t \in T_1'(e_i, v_i)$. Define π_i as a map between the set of all edges, such that it agrees with σ_i on all edges of $X_{f(i)}'$ and with χ_t (for $t \in T_1'(e_i, v_i)$) on the coloured virtual edges.

We claim that π_i must map the twin-edge of f_i to twin-edge of $\tau(f_i)$. Suppose not. By the property of the colouring, this implies that there is a subtree of $T_1'(e_i, v_i)$ isomorphic to $T_1' \setminus T_1'(e_i, v_i)$. This contradicts the assumption that c_α is the centre of T_1'.

For each edge $e \in E_1$, define $\sigma(e)$ to be $\chi(e)$ when $e \in c_\alpha$ and to be $\pi_i(e)$ when $e \in E_{f_i}$ (edges of X_{f_i}).

From the above argument, $\chi = \chi_\alpha$ and σ_i indeed agrees on where it maps f_i to. This ensures that every cycle passing through the separating pairs of c_α gets preserved. Thus σ is a 2-isomorphism between X_1' and X_2'.

For case 2, let T_1' and T_2' have two centres (α_1, α_2) and (β_1, β_2) respectively. An essentially similar argument works in this case too. ■

This completes the proof of correctness of the algorithm (Lemma 3). ■

To complete the proof of Theorem 3, we need the following proposition:

Proposition 3. Coloured-GMI *for 3-connected graphs reduces to* GI.

Observe that the above construction does not use non-planar gadgets. It is known that isomorphism testing for planar 3-connected graphs can be done in linear time [7] (in fact in L [4]) we get the following.

Corollary 2. PMI $\in \mathsf{P}$.

Now we give a polynomial time many-one reduction from MI_b to GMI.

Theorem 5. $\mathrm{MI}_b \leq_m^p$ GMI.

Combining Corollary 1, Theorem 3 and Theorem 5 we have,

Theorem 6. GI $\equiv_T^p$ GMI $\equiv_T^p$ $\mathrm{MI}_b \equiv_T^p \mathrm{LMI}_b$.

6 Matroid Automorphism Problem

With any isomorphism problem, there is an associated automorphism problem i.e, to find a generating set for the automorphism group of the underlying object. Relating the isomorphism problem to the corresponding automorphism problem gives access to algebraic tools associated with the automorphism groups. In the case of graphs, studying automorphism problem has been fruitful.(e.g. see [12].) In this section we turn our attention to Matroid automorphism problem.

An automorphism of a matroid $M = (S, \mathcal{C})$ (where S is the ground set and $\mathcal{C}$ is the set of circuits) is a permutation ϕ of elements of S such that $\forall C \subseteq S$, $C \in \mathcal{C} \iff \phi(C) \in \mathcal{C}$. $Aut(M)$ denotes the group of automorphisms of the matroid M. When the matroid is graphic we denote by $Aut(X)$ and $Aut(M_X)$ the automorphism group of the graph and the graphic matroid respectively.

To begin with, we note that given a graph X, and a permutation $\pi \in S_m$, it is not clear a priori how to check if $\pi \in Aut(M_X)$ efficiently. This is because we need to ensure that π preserves all the simple cycles, and there could be exponentially many of them. Note that such a membership test (given a $\pi \in S_n$) for $Aut(X)$ can be done easily by testing whether π preserves all the edges. We provide an efficient test for this problem, i.e.,

Theorem 7. *Given any $\pi \in S_m$, testing if $\pi \in Aut(M_X)$ can be done in* P.

To prove the above theorem, we use the notion of a cycle bases of X. A *cycle basis* of a graph X is a minimal set of cycles $\mathcal{B}$ of X such that every cycle in X can be written as a linear combination (viewing every cycle as a vector in $\mathbb{F}_2^m$) of the cycles in $\mathcal{B}$. Let $\mathscr{B}$ denote the set of all cycle basis of the graph X.

Lemma 4. *Let $\pi \in S_n$, $\exists \mathcal{B} \in \mathscr{B} : \pi(\mathcal{B}) \in \mathscr{B} \implies \forall \mathcal{B} \in \mathscr{B} : \pi(\mathcal{B}) \in \mathscr{B}$*

Lemma 5. *Let $\pi \in S_m$, and let $\mathcal{B} \in \mathscr{B}$, then $\pi \in Aut(M_X) \iff \pi(\mathcal{B}) \in \mathscr{B}$.*

Using Lemmas 4 and 5 it follows that, given a permutation π, to test if $\pi \in Aut(M_X)$ it suffices to check if for a cycle basis $\mathcal{B}$ of X, $\pi(\mathcal{B})$ is also a cycle basis. Given a graph X a cycle basis $\mathcal{B}$ can be computed in polynomial time (see e.g, [8]). Now it suffices to show:

Lemma 6. *Given a permutation $\pi \in S_m$, and a cycle basis $\mathcal{B} \in \mathscr{B}$, testing whether $\pi(\mathcal{B})$ is a cycle basis, can be done in polynomial time.*

Notice that similar arguments can also give another proof of Proposition 2. As in the case of graphs, we can define automorphism problems for matroids.

MATROID AUTOMORPHISM(MA)*: Given a matroid M as independent set oracle, compute a generating set for $Aut(M)$.*

We define GMA and LMA as the corresponding automorphism problems for graphic and linear matroids, when the input is a graph and matrix respectively. Using the colouring techniques from Section 4, we prove the following.

Theorem 8. LMI $\equiv_T^p$ LMA*, and* GMI $\equiv_T^p$ GMA*.*

References

1. Arvind, V., Torán, J.: Isomorphism testing: Perspective and open problems. Bulletin of the EATCS 86, 66–84 (2005)
2. Babai, L.: Vector Representable Matroids of Given Rank with Given Automorphism Group. Discrete Math. 24, 119–125 (1978)
3. Cunningham, W.H., Edmonds, J.: A Combinatorial Decomposition Theory. Canad. Jl. of Math. 17, 734–765 (1980)
4. Datta, S., Limaye, N., Nimbhorkar, P.: 3-connected planar graph isomorphism is in log-space. In: FSTTCS (2008)
5. Goldreich, O.: Computational Complexity: A Conceptual Perspective. Cambridge Univ. Press, Cambridge (2008)
6. Hlinený, P.: Some hard problems on matroid spikes. Theory of Computing Sys. 41(3), 551–562 (2007)
7. Hopcroft, J., Tarjan, R.: Dividing a graph into triconnected components. SIAM Jl. of Comp. 2(3), 135–158 (1973)
8. Horton, J.D.: A Poly-time Algorithm to Find the Shortest Cycle Basis of a graph. SIAM Jl. of Comp. 16(2), 358–366 (1987)
9. Jenner, B., Köbler, J., McKenzie, P., Torán, J.: Completeness results for graph isomorphism. J. Comput. Syst. Sci. 66(3), 549–566 (2003)
10. Köbler, J.: On graph isomorphism for restricted graph classes. In: Beckmann, A., Berger, U., Löwe, B., Tucker, J.V. (eds.) CiE 2006. LNCS, vol. 3988, pp. 241–256. Springer, Heidelberg (2006)
11. Köbler, J., Schöning, U., Torán, J.: The Graph Isomorphism Problem: its Structural Complexity. Birkhauser, Basel (1993)
12. Luks, E.M.: Isomorphism of graphs of bounded valence can be tested in polynomial time. In: FOCS, pp. 42–49 (1980)
13. Mayhew, D.: Matroid Complexity and Nonsuccinct Descriptions. SIAM Jl. D. Math. 22(2), 455 (2008)
14. Oxley, J., Welsh, D.: Chromatic, flow and reliability polynomials: The complexity of their coefficients. Comb. Prob. and Comp. 11, 403–426 (2002)
15. Oxley, J.G.: Matroid theory. Oxford University Press, New York (1992)
16. Raghavendra Rao, B.V., Sarma, J.M.N.: On the complexity of matroid isomorphism problem, November 24 (2008), http://arxiv.org/abs/cs.CC/0811.3859
17. Toran, J.: On the Hardness of Graph Isomorphism. SIAM Jl. of Comp. 33(5), 1093–1108 (2004)

18. Truemper, K.: On Whitney's 2-isomorphism theorem for graphs. Jl. of Graph Th., 43–49 (1980)
19. Whitney, H.: Congruent graphs and connectivity of graphs. American Journal of Mathematics 54(1), 150–168 (1932)
20. Whitney, H.: 2-isomorphic graphs. American Journal of Mathematics 55, 245–254 (1933)

Breaking Anonymity by Learning a Unique Minimum Hitting Set

Dogan Kesdogan[1], Daniel Mölle[2,⋆], Stefan Richter[2], and Peter Rossmanith[2]

[1] University of Siegen, Germany
kesdogan@fb5.uni-siegen.de
[2] RWTH Aachen University, Germany
{moelle,richter,rossmani}@cs.rwth-aachen.de

Abstract. Anonymity protocols are not secure unless the communication structure is not learnable even in the case that the entire network traffic can be monitored by an adversary. When the true communication structure is cloaked under anonymity sets, the adversary may disclose the peers of a certain user by waiting for the observations to contain a unique minimum hitting set. This approach has been called the hitting set attack in the literature. We give the first mathematical analysis on the number of observations required to learn a unique minimum hitting set. Because this attack involves solving an NP-hard problem in each round, we propose two new learning algorithms, both of which are very efficient computationally. The first one breaks anonymity by combining the most suspicious elements into a hitting set. Because this algorithm is not capable of verifying its hypothesis, it is imperative to estimate the required number of observations. On the other hand, the second one is able to prove its hypothesis correct, but needs more observations.

1 Introduction

Analysing learning algorithms helps to understand the complexity of hard problems. When solutions to a certain problem turn out to be learnable in little time, the resulting algorithm can be employed to solve the problem in every-day applications. In so far, learnability yields a positive answer to the question of tractability.

Conversely, a system that relies on the intractability of a computational problem can be proven useless, or at least endangered, by adequate results on the learnability of that problem. In this paper, we show how such results can be used to analyze an anonymity system that relies on the hardness of the unique minimum hitting set problem. In fact, this so called *hitting set attack* [6] sets a limit to the possible anonymity of persistent communication in any anonymity system if we follow the well-accepted definition of anonymity as "the state of being not identifiable within a set of subjects, the anonymity set." [5]. Thus, we begin with a short introduction to the general field of anonymity techniques in

⋆ This author was supported by the DFG project TAPI, grant RO 927/6-1.

order to justify our model. George Danezis and Claudia Diaz have recently published a comprehensive "Survey on Anonymous Communication Channels" [3], to which we refer the interested reader for further details.

1.1 Anonymity Techniques

To protect traffic information, a number of anonymity techniques have been suggested [1,2], An anonymity technique —a so called MIX server— collects n data packets from n distinct users, changing the appearance and order of the packets so that no outsider can link an incoming packet to an outgoing packet. Thus, this mechanism conceals the specific communication relationships of Alice amidst the additional traffic of the other users. Following this general description we model the attacker and the anonymity system so that the attacker observes and records all sets of incoming and outgoing packets to and from a MIX if Alice has contributed with a message. Hence, the following abstract model can be used:

Assume [6] that the set of all peers is $A = \{1, 2, \ldots, N\}$ and that Alice frequently communicates with a subset B of A, say $B = \{1, 2, \ldots, m\}$. Then the aim of the adversary is to learn B. Since Alice uses a strong anonymity technique, the elements of B are only observable in terms of anonymity sets (more precisely, multisets), i.e., the adversary makes observations $\mathcal{O}_i$ consisting of one element from B and $n-1$ additional elements from A, with multiple occurrences possible. For the adversary, given only one observation and uniform distribution, each element of $\mathcal{O}_i$ is equally likely to be the real peer partner. To learn B, the adversary collects a number of observations $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_t$ and tries to determine the set of real peer partners. The resulting question is: what is the minimum number of t required to learn B?

It is commonly believed in the security community[1], as well as intuitive from an information theoretic point of view, that non-uniform distributions only ease the task of revealing that piece of information, as they eliminate maximum uncertainty. Thus, the uniform distribution model is not only mathematically smooth but can also be justified as a kind of worst-case scenario for the adversary, provided that the anonymizer cannot adapt to the model or algorithm used by the adversary.

In earlier scholarship [6], a learning algorithm based on the computation of hitting sets[2] has been suggested and evaluated using simulations, and it has been shown that B is exactly learnable if the learning algorithm can identify a unique minimum hitting set[3]. Informally, B can be identified as the unique minimum hitting set of $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_t$, because B has to intersect (hit) all observations by

[1] "All other things being equal, anonymity is the stronger, the larger the respective anonymity set is and the more evenly distributed the sending or receiving, respectively, of the subjects within that set is." [8]

[2] Given a family of multisets $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_t$, a set B is called a *hitting set* for this family iff, for every $1 \leq i \leq t$, $|B \cap O_i| > 0$.

[3] We call a hitting set B *unique minimum* for a family $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_t$ iff $|B| < |C|$ for every other hitting set $C \neq B$.

construction. For the sake of simplicity, the number of peer partners $|B|$ is often assumed to be known. In this work, we give a theoretical analysis by assuming the same parameters $N := |A|$, $m := |B|$, and $n := |\mathcal{O}_i|$. Surprisingly, our results do not depend on any knowledge of N, m or n.

1.2 Contributions

The paper at hand consists of three main results, the first of which is a bound on the number t of observations required for B to be a unique minimum hitting set, rather than just a hitting set of possibly minimum size, for $\mathcal{O}_1, \ldots, \mathcal{O}_t$. This is followed by two learning algorithms with distinct objectives. The first, simple one shows that the unique minimum hitting set can be learnt efficiently and within a small number of observations in many settings. The second, more involved one relies on more preconditions and is highly likely to take more time, but it is able to prove its hypothesis correct with high probability.

It is easy to see that B can be identified in the limit even if m is not known: A text that contains all observations contains a multisets $\{i, \ldots, i\}$ for every $i \in B$ as well. A simple learning strategy is to present the set of all i's seen in such pure observations. If m is known, this strategy is even self-monitoring. As such pure observations will only occur extremely rarely in practice, however, such a learning algorithm seems to be rather useless. Knowledge about the underlying probability distribution leads to much faster algorithms [10].

When using more sophisticated learning algorithms, the correctness of a hypothesis H can be shown by proving that H is a unique minimum hitting set [6]. Whereas it is easy to verify that H is a hitting set, determining uniqueness is known to be NP-complete. Until now, this has been an obstacle for the hitting set attack [6]. Exploiting a special structure of the hitting set instances, however, the second algorithm can prove uniqueness in linear time.

The following table gives some examples for the numbers of observations required to reach an error probability of at most 1/100 in each of the three aforementioned contexts: uniqueness (U), simple algorithm (A1), and advanced algorithm (A2). The parameters have been chosen to exemplify practically plausible configurations. On the other hand, the results show nontrivial dependency upon parameter variation for the derived worst-case bounds.

m	n	N	U	A1	A2
10	20	20000	19	14898	783048
10	50	50000	22	14899	759100
20	50	20000	53	66468	580094
20	50	50000	46	66259	1449190

Note that all three bounds are upper limits on the security of an anonymity system under varying thread models.

2 The Uniqueness of Hitting Sets

In this section, we show bounds on the probability for B to become a unique minimum hitting set after a certain number of observations. There are many experimental results on the learning speed of the hitting set attack (see, e.g., [6]) and corresponding conjectures, but as of today, there has been no accompanying mathematical investigation. Let us first recall and formalize the objects in question.

Definition 1. *Let $\mathcal{O}_1, \ldots, \mathcal{O}_t$ be a sequence of observations, where each $\mathcal{O}_i$ is a length-n multiset taken from A that contains at least one element from B. For any element $i \in A$, Z_i denotes the number of observations containing i. Let Z_B and $Z_{A \setminus B}$ be random variables that have the same distribution as Z_i for $i \in B$ and $i \in A \setminus B$, respectively. Let finally*

$$p_{A\setminus B} := 1 - \left(1 - \frac{1}{N}\right)^{n-1}.$$

Note that $p_{A\setminus B}$ describes the probability that a fixed element from $A \setminus B$ occurs in a random observation. We are now able to establish the first result.

Theorem 1. *The probability that B is not a unique minimum hitting set for $\mathcal{O}_1, \ldots, \mathcal{O}_t$ is at most $\binom{N}{m} \exp(-\frac{1}{m}(1 - m/N)^{n-1}t)$.*

Proof. By construction, B is a hitting set for $\mathcal{O}_1, \ldots, \mathcal{O}_t$. Let

$$\mathcal{H} := \left\{ H \subseteq A \;\middle|\; |H| = m, H \neq B \right\} = \binom{A}{m} \setminus \{B\}.$$

In order for B to be unique, the family $\mathcal{H}$ must not contain any hitting set for $\mathcal{O}_1, \ldots, \mathcal{O}_t$. That is, there has to exist an $i \in \{1, \ldots, t\}$ for every $H \in \mathcal{H}$ such that $H \cap \mathcal{O}_i = \emptyset$.

Each $\mathcal{O}_i$ can be written as $\{b_i\} \cup A_i$, where $b_i \in B$ and A_i is a multiset consisting of $n-1$ elements from A. Since $H \neq B$, there exists an $x \in B \setminus H$. The probability for the event $[b_i = x]$ is $1/m$, and the probability of choosing an element from $A \setminus H$ when choosing from A is $(N-m)/N$. Hence,

$$Pr[H \cap \mathcal{O}_i = \emptyset] \geq \frac{1}{m}\left(\frac{N-m}{N}\right)^{n-1}.$$

This implies

$$Pr[\forall i \; H \cap \mathcal{O}_i \neq \emptyset] \leq \left(1 - \frac{1}{m}\left(\frac{N-m}{N}\right)^{n-1}\right)^t \leq e^{-\frac{1}{m}(1-m/N)^{n-1}t}$$

and, finally,

$$Pr[\exists H \in \mathcal{H} \; \forall i \; H \cap \mathcal{O}_i \neq \emptyset] \leq \binom{N}{m} e^{-\frac{1}{m}(1-m/N)^{n-1}t}.$$

□

3 A Simple Learning Algorithm

The unostentatious algorithm investigated in this section tries to learn B by combining the elements $i \in A$ of the largest Z_i into a hitting set for the present sequence $\mathcal{O}_1, \ldots, \mathcal{O}_t$ of observations. We prove that with a certain probability, this method suffices to learn B within a bounded number of observations. The following lemma establishes two expectations required for future calculations.

Lemma 1

$$EZ_B = \frac{t}{m} + t(1 - \frac{1}{m})p_{A \setminus B}$$
$$EZ_{A \setminus B} = tp_{A \setminus B}$$

Proof. Each $\mathcal{O}_i$ can be written as $\mathcal{O}_i = \{b_i\} \cup A_i$ for some $b_i \in B$ and a multiset A_i consisting of $n - 1$ elements from A. For a single observation $\mathcal{O}_i$, the probability for some fixed $j \in B$ to be b_i is $1/m$. The probability for j to be chosen into A_i is $p_{A \setminus B}$. By the principle of inclusion/exclusion, we get $\Pr[j \in \mathcal{O}_i] = \Pr[b_i = j] + \Pr[b_i \neq j] \Pr[j \in A_i] = 1/m + (1 - 1/m)p_{A \setminus B}$. By additivity of expectation, $EZ_B = EZ_j = t \Pr[j \in \mathcal{O}_i]$. A similar but even simpler argument shows the second claim. □

We also define the order $\succ$ on A to ease the descriptions of our algorithms. Notice that $\succ$ is obviously total.

Definition 2. *For $i, j \in A$ and occurrence counters Z_i, Z_j, let*

$$i \succ j \equiv ((Z_i > Z_j) \vee ((Z_i = Z_j) \wedge (i > j))).$$

Moreover, let $i_1, \ldots, i_N$ be the elements from A sorted according to $\succ$, or, in a more formal notation, $i_1 \succ i_2 \succ \ldots \succ i_N$.

Surprisingly, our first algorithm does not even need to know m. Instead, it just uses a kind of greedy majority vote to compute its hypothesis. There is, however, no guarantee that it will learn B entirely, unless no subset of B forms a hitting set for the present sequence of observations.

1. Initially, set $t := 0$ and $Z_i := 0$ for all $i \in A$.
2. Increase t and read the t-th observation $\mathcal{O}_t$. For every element $i \in A$ that occurs at least once in $\mathcal{O}_t$, increase Z_i.
3. Set $H_t := \emptyset$. Add $i_1, i_2, \ldots$ to H_t until H_t is a hitting set for $\mathcal{O}_1, \ldots, \mathcal{O}_t$. Go to step (2).

We want to show that the hypothesis H_t is likely to be (partly) correct after a certain number of observations, at least in the case that the (first two of the) following three conditions hold for an appropriate number c:

$P_1 \equiv Z_i \geq EZ_B - c\sqrt{EZ_B}$ for all $i \in B$,
$P_2 \equiv Z_i \leq EZ_{A \setminus B} + c\sqrt{EZ_{A \setminus B}}$ for all $i \in A \setminus B$, and
$P_3 \equiv$ for all $i \in B$ there is a $1 \leq j \leq t$ such that $B \cap \mathcal{O}_j = \{i\}$.

Lemma 2

$$\Pr[\bar{P}_1] \leq e^{-c^2/2},$$
$$\Pr[\bar{P}_2] \leq e^{-c^2/3},$$
$$\Pr[\bar{P}_3] \leq m \cdot \exp\Big(-\frac{1}{m}\Big(1-\frac{m-1}{N}\Big)^{n-1} t\Big).$$

Proof. The first two claims follow from Chernoff bounds [7, 4.6] because Z_i is binomially distributed. To see the third claim, check that for an arbitrary $i \in B$,

$$\alpha := \Pr[\forall 1 \leq j \leq t.\ B \cap O_j \neq \{i\}] = \Big(1-\frac{1}{m}\Big(1-\frac{m-1}{N}\Big)^{n-1}\Big)^t$$

and estimate according to $\Pr[\exists i \in B.\ \forall 1 \leq j \leq t.\ B \cap O_j \neq \{i\}] \leq m\alpha$ as well as $(1-\beta)^t \leq e^{-\beta t}$. □

Corollary 1. $\Pr[P_1 \wedge P_2] \geq 1 - e^{-c^2/2} - e^{-c^2/3} \geq 1 - 2e^{-c^2/3}$.

We first establish a bound for the partial correctness, i.e., that the hypothesis contains only elements from B.

Lemma 3. $\Pr[H_t \not\subseteq B] \leq 2\exp(-t(1-p_{A\setminus B})^2/12m^2)$.

Proof. Let $c = \sqrt{t}(1-p_{A\setminus B})/2m$ and apply Corollary 1 to see that $P_1 \wedge P_2$ holds with probability at least $1 - 2\exp(-t(1-p_{A\setminus B})^2/12m^2)$.

It is obvious that $H_t \subseteq B$ as soon as $Z_i > Z_j$ for all $i \in B$ and $j \in A \setminus B$, since in that case, the algorithm chooses only elements from B for addition to H_t. Under the assumption that P_1 and P_2 hold, it hence suffices to have that

$$EZ_B - c\sqrt{EZ_B} > EZ_{A\setminus B} + c\sqrt{EZ_{A\setminus B}}.$$

Using the definitions of EZ_B and $EZ_{A\setminus B}$ as well as simple transformations, it is easy to see that this is the case whenever

$$t > \left(\frac{cm}{1-p_{A\setminus B}}\left(\sqrt{\frac{1}{m}+(1-1/m)p_{A\setminus B}}+\sqrt{p_{A\setminus B}}\right)\right)^2. \qquad (1)$$

According to the fact that $p_{A\setminus B} < 1$, the right hand side of (1) is strictly less than $(2cm/(1-p_{A\setminus B}))^2 = t$. Hence, (1) holds for all t. □

In addition, we want to estimate the probability for the algorithm to find B itself.

Lemma 4

$$\Pr[H_t \neq B] \leq 2\exp(-t(1-p_{A\setminus B})^2/12m^2) + m\exp\Big(-\frac{1}{m}\Big(1-\frac{m-1}{N}\Big)^{n-1} t\Big).$$

Proof. For a sequence $\mathcal{O}_1, \ldots, \mathcal{O}_t$ of observations, let us call an $i \in B$ *dispensable* if $B \setminus \{i\}$ is a hitting set, that is, if there is no $1 \le j \le t$ such that $B \cap \mathcal{O}_j = \{i\}$. Moreover, we call B *oversaturated* if it contains at least one dispensable element. For some $i \in B$ and $1 \le j \le t$, we have that

$$\Pr[B \cap \mathcal{O}_j = \{i\}] = \frac{1}{m}\Big(1 - \frac{m-1}{N}\Big)^{n-1}.$$

It is hence easy to see that, for an arbitrary $i \in B$,

$$\Pr[i \text{ is dispensable}] = \Big(1 - \frac{1}{m}\Big(1 - \frac{m-1}{N}\Big)^{n-1}\Big)^t \le e^{-\frac{1}{m}(1-\frac{m-1}{N})^{n-1}t}.$$

There are only two ways for H_t not to equal B: if H_t is not even a subset of B, or if B contains a dispensable element. Thus,

$$\Pr[H_t \ne B] \le \Pr[H_t \not\subseteq B] + \Pr[B \text{ is oversaturated}].$$

Using the bounds calculated above and in Lemma 3, the claim follows. □

In what follows, S denotes the sampling complexity, that is, the number of observations required to learn B. Formally, $S = \min\{\, t \mid H_\tau = B \text{ for all } \tau \ge t\,\}$. Our algorithm is likely to find B, or a subset thereof, rather efficiently in many settings. However, there is no guarantee that it will not abandon a correct hypothesis in the future. In particular, the algorithm is not conservative. To establish a bound on S, we have to use the bound from Lemma 4 on infinitely many values.

Theorem 2. $\Pr[S > t] \le 2e^{-d_1 t}/(1 - e^{-d_1}) + me^{-d_2 t}/(1 - e^{-d_2})$, *where* $d_1 = (1 - p_{A\setminus B})^2/12m^2$ *and* $d_2 = \frac{1}{m}(1 - (m-1)/N)^{n-1}$.

Proof. Lemma 4 implies that $\Pr[H_\tau \ne B] \le 2e^{-d_1 t} + me^{-d_2 t}$. Summing up for all $\tau \ge t$ yields the claim. □

4 An Advanced Learning Algorithm

The aforementioned algorithm may compute a correct hypothesis within only a small number of observations in many settings, but it is not capable of proving the hypothesis correct. In opposition to that, our advanced algorithm is able to check and prove that its hypothesis is correct, but does not work for ill-conditioned combinations of N, m, and n. These, however, can be avoided by demanding that, e.g., $p_{A\setminus B} < 1/m^2$. Note that this condition will be satisfied whenever N is much larger than n and m. For instance, it holds for the values depicted in the table that concludes Section 1.

In what follows, we formalize three conditions that will be crucial in the development. For the moment, let c be an arbitrary number,

$$Q_1 \equiv Z_i \le EZ_B + c\sqrt{EZ_{A\setminus B}} \text{ for all } i \in B,$$

$Q_2 \equiv Z_i \geq EZ_B - c\sqrt{EZ_{A\setminus B}}$ for all $i \in B$, and
$Q_3 \equiv Z_i \leq EZ_{A\setminus B} + c\sqrt{EZ_{A\setminus B}}$ for all $i \in A \setminus B$.

It will turn out that these three properties ensure that B is a unique minimum hitting set for $\mathcal{O}_1, \ldots, \mathcal{O}_t$, given appropriate values for $p_{A\setminus B}$ and t.

Lemma 5

$$\Pr[\bar{Q}_1] \leq \exp\Big(-\frac{c^2}{3}\,\frac{p_{A\setminus B}}{1/m + (1-1/m)p_{A\setminus B}}\Big),$$
$$\Pr[\bar{Q}_2] \leq \exp\Big(-\frac{c^2}{2}\,\frac{p_{A\setminus B}}{1/m + (1-1/m)p_{A\setminus B}}\Big),$$
$$\Pr[\bar{Q}_3] \leq e^{-c^2/3}.$$

Proof. The claims follow from Chernoff bounds [7, (4.6)]. □

For the rest of the paper, let us fix $c = \sqrt{6\ln 5/mp_{A\setminus B}}$. This allows for the following result, which is necessary for the applicability of our second algorithm.

Lemma 6. $\Pr[Q_1 \wedge Q_2 \wedge Q_3] > \frac{1}{2}$.

Proof. Using the above lemma, we get that $\Pr[\bar{Q}_1], \Pr[\bar{Q}_2] \leq \frac{1}{5}$. Observe that $c > 3$ in any case, implying that $\Pr[\bar{Q}_3] < \frac{1}{10}$. Altogether, we get the result that $\Pr[\bar{Q}_1 \vee \bar{Q}_2 \vee \bar{Q}_3] < \frac{1}{5} + \frac{1}{5} + \frac{1}{10} = \frac{1}{2}$. □

The second algorithm is described below. Whereas the first two steps are exactly the same as in the first method, the Z_i are then used for entirely different computations. Since the Z_i are in descending order, h represents the maximum number of observations that can be explained (hit) when we replace at least one element of H_t. Thus, if $h < t$, H_t is the only hitting set of its size that is able to explain our observations, the unique minimum hitting set.

1. Initially, set $t := 0$ and $Z_i := 0$ for all $i \in A$.
2. Increase t and read the t-th observation $\mathcal{O}_t$. For every element $i \in A$ that occurs at least once in $\mathcal{O}_t$, increase Z_i.
3. Set $H_t := \{i_1, \ldots, i_m\}$ and $h := Z_{i_1} + \ldots + Z_{i_{m-1}} + Z_{i_{m+1}}$.
4. Output H_t. If $h < t$, then claim that $H_t = B$ and stop. Otherwise go to step (2).

Observe that, instead of assuming m to be known, the algorithm can simply choose the smallest m such that $\{i_1, \ldots, i_m\}$ is a hitting set, just as in the simple Algorithm. The following theorem and the trailing corollary show some conditions under which B becomes a unique minimum hitting set, where uniqueness follows from a simple counting argument that is easy to check.

Theorem 3. *Let* $p_{A\setminus B} < 1/m^2$ *and assume*

$$t > \frac{6\ln(5)m}{(1/m - p_{A\setminus B}(m-1+1/m))^2}.$$

In the event that Q_1, Q_2 and Q_3 hold, we have that B is a unique minimum hitting set for $\mathcal{O}_1, \ldots, \mathcal{O}_t$. Moreover, the elements in B occur in more observations than any element from $A \setminus B$, and every other m-element set $H \subseteq A$ has $\sum_{i \in H} Z_i < t$.

Proof. By definition, B is a hitting set for $\mathcal{O}_1, \ldots, \mathcal{O}_t$. Notice that in particular this implies $\sum_{i \in B} Z_i \geq t$. We will show that replacing even a single element from B by an element from $A \setminus B$ results in missing the required lower bound of t occurrences. This will imply that B can be obtained by majority voting and that no proper subset suffices to cover all observations. Thus the claim follows.

To see the statement in question, let us look at an upper bound on the number of observations hit by $m-1$ elements from B and one element from $A \setminus B$. Such an upper bound is given by Q_1 and Q_3:

$$\begin{aligned}
&(m-1)(EZ_B + c\sqrt{EZ_{A\setminus B}}) + EZ_{A\setminus B} + c\sqrt{EZ_{A\setminus B}} \\
&= (m-1)EZ_B + EZ_{A\setminus B} + mc\sqrt{EZ_{A\setminus B}} \\
&= (m-1)\frac{t}{m} + (m-1)t\Big(1 - \frac{1}{m}\Big)p_{A\setminus B} + tp_{A\setminus B} + mc\sqrt{tp_{A\setminus B}}
\end{aligned}$$

We want to show that the upper bound lies below t. Equivalently, it suffices to prove that

$$mc\sqrt{\frac{p_{A\setminus B}}{t}} < 1 - \left(\frac{m-1}{mp_{A\setminus B}} + \frac{(m-1)^2}{m} + 1\right)p_{A\setminus B}.$$

Using $p_{A\setminus B} < 1/m^2$, a simple calculation shows that the right hand side of the inequality is positive. Thus we have to ensure that

$$\sqrt{t} > \frac{mc\sqrt{p_{A\setminus B}}}{1 - \big((m-1)/mp_{A\setminus B} + (m-1)^2/m + 1\big)p_{A\setminus B}}.$$

This is equivalent to

$$t > \frac{m^2c^2p_{A\setminus B}}{(1/m - p_{A\setminus B}(m-1+1/m))^2} = \frac{6\ln(5)m}{(1/m - p_{A\setminus B}(m-1+1/m))^2}.$$

□

Corollary 2. *Let $p_{A\setminus B} < 1/m^2$ and assume $t > c^2m^2(1+1/m)$. In the event that Q_1, Q_2 and Q_3 hold, we have that B is a unique minimum hitting set for $\mathcal{O}_1, \ldots, \mathcal{O}_t$. Moreover, the elements in B occur in more observations than any element from $A \setminus B$, and every other m-element set $H \subseteq A$ has $\sum_{i \in H} Z_i < t$.*

Proof. Observe that $t > c^2m^2(1+1/m)$ and $p_{A\setminus B} < 1/m^2$ imply

$$t > \frac{6\ln(5)m}{(1/m - p_{A\setminus B}(m-1+1/m))^2}.$$

Thus, the claim follows according to Theorem 3. □

Let S' denote the minimum number t of observations our advanced algorithm takes before it claims that $H_t = B$. Using the above as well as older results, we can easily derive bounds on S' as follows.

Theorem 4. *If* $p_{A \setminus B} < 1/m^2$, *then* $\Pr[S' \geq c^2 m^2 (1 + 1/m)] < \frac{1}{2}$.

Proof. Combine Lemma 6 and Corollary 2. $\square$

Corollary 3. *If* $p_{A \setminus B} < 1/m^2$, *then* $\Pr[S' \geq k c^2 m^2 (1 + 1/m)] \leq 2^{-k}$.

Proof. The claim follows from [9, Theorem 6] if the algorithm is both conservative and rearrangement-independent. This can be established by modifying the algorithm as to use $\emptyset$ as its hypothesis unless $h < t$, that is, unless the algorithm claims that $H_t = B$. $\square$

5 Conclusions

We have seen that learning algorithms can be used to break anonymity protocols. Whereas even a small number of observations may give enough information to match communication partners, the required computation involves solving hard problems. Given the chance to make many observations, however, the instances tend to have a certain structure that can be exploited in order to find the secret in question efficiently.

Once again, this shows how hard problems can be seen to be tractable — at least with high probability — in every-day applications, meaning that most instances that occur in practice are not so computationally hard at all. Policywise this implies that it does not suffice to base anonymity or security in general on the intractability of a computational problem. Instead, we need to look closer at the hardness of the problem instances arising in the protocol at hand. This is an approach taken by an active community of researchers in the field of exact and parameterized algorithms for hard problems (See, e.g., [4] for a survey). Our results argue for a close collaboration with this community when designing security protocols, in order to forestall unexpected attacks.

It remains to test the algorithms on data collected in real networks. Efforts in this direction are under way. In addition, it would be interesting to see lower bounds on the number of observations our algorithms require.

References

1. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), 84–88 (1981)
2. Chaum, D.: The dining cryptographers problem: Unconditional sender and recipient untraceability. Journal of Cryptology 1(1), 65–75 (1988)
3. Danezis, G., Diaz, C.: A survey of anonymous communication channels. Technical Report MSR-TR-2008-35, Microsoft Research (January 2008)
4. Fomin, F.V., Grandoni, F., Kratsch, D.: Some new techniques in design and analysis of exact (exponential) algorithms. EATCS Bulletin 87, 47–77 (2005)

5. Hansen, M., Pfitzmann, A.: Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. Draft (July 2000)
6. Kesdogan, D., Pimenidis, L.: The hitting set attack on anonymity protocols. In: Fridrich, J. (ed.) IH 2004. LNCS, vol. 3200, pp. 326–339. Springer, Heidelberg (2004)
7. Motwani, R., Raghavan, P.: Randomized Algorithms. Cambridge University Press, Cambridge (1995)
8. Pfitzmann, A., Köhntopp, M.: Anonymity, unobservability, and pseudonymity: A consolidated proposal for terminology. Draft, version 0.23 (July 2000); For date of the latest version, see the document itself
9. Rossmanith, P., Zeugmann, T.: Stochastic finite learning of the pattern languages. Machine Learning 44(1/2), 67–91 (2001)
10. Zeugmann, T.: Stochastic finite learning. In: Steinhöfel, K. (ed.) SAGA 2001. LNCS, vol. 2264, pp. 155–171. Springer, Heidelberg (2001)

The Budgeted Unique Coverage Problem and Color-Coding
(Extended Abstract)

Neeldhara Misra[1], Venkatesh Raman[1], Saket Saurabh[2], and Somnath Sikdar[1]

[1] The Institute of Mathematical Sciences, Chennai, India
{neeldhara,vraman,somnath}@imsc.res.in
[2] University of Bergen, Bergen, Norway
saket.saurabh@ii.uib.no

Abstract. We show, by a non-trivial application of the color-coding method of Alon et al. [2], that BUDGETED UNIQUE COVERAGE (a variant of SET COVER) is fixed-parameter tractable, answering an open problem posed in [13]. We also give improved fixed-parameter tractable algorithms for two special cases of BUDGETED UNIQUE COVERAGE: UNIQUE COVERAGE (the unweighted version) and BUDGETED MAX CUT.

To derandomize our algorithms we use an interesting variation of k-perfect hash families known as (k, s)-hash families which were studied by Alon et al. [1] in the context of a class of codes called parent identifying codes [3]. In this setting, for every s-element subset S of the universe, and every k-element subset X of S, there exists a function that maps X injectively and maps the remaining elements of S into a different range.

We give several bounds on the size of (k, s)-hash families. We believe that our application of color-coding may be used for other problems and that this is the first application of (k, s)-hash families to a problem outside the domain of coding theory.

1 Introduction

The UNIQUE COVERAGE problem is a variant of SET COVER where, given a family of subsets of a finite universe, one is interested in finding a subfamily that maximizes the number of elements *uniquely* covered. This problem is motivated by a real-world application arising in wireless networks and has connections to several problems including MAX CUT and MAXIMUM COVERAGE [10].

Demaine et al. [5] introduced this problem and gave efficient approximation algorithms and inapproximability results. Moser et al. [13] studied the parameterized complexity of UNIQUE COVERAGE. They show that the problem is fixed-parameter tractable when parameterized by the number of elements to be uniquely covered. In particular, they left open the parameterized complexity of the more general version where elements have integral profits and sets have integral costs and one is interested in maximizing the total profit of elements uniquely covered by a minimum cost subfamily. In this paper, we show that (the standard parameterized version of) BUDGETED UNIQUE COVERAGE is fixed-parameter tractable. We also give improved algorithms for two special cases of BUDGETED UNIQUE COVERAGE: UNIQUE COVERAGE, the unweighted version of the problem and BUDGETED MAX CUT, a weighted variant of the well-known MAX CUT problem. See [7] and [9] for other related work on the UNIQUE COVERAGE problem.

In the BUDGETED UNIQUE COVERAGE problem, we are given a universe, where each element has a positive integral profit and a family of subsets of the universe, where each set has a positive integral cost. The question is whether there is a subfamily with total cost at most B that uniquely covers elements with total profit at least k. We show that this problem is fixed-parameter tractable with parameters k and B using the color-coding technique introduced by Alon et al. [2]. It is possible to derandomize the algorithm using standard s-perfect hash families where s is the maximum number of elements in a solution subfamily. However, we can use a variation of s-perfect families called (k, s)-hash families which were introduced in the context of a class of codes called parent identifying codes [3,1]. To the best of our knowledge, we provide the first application of this class of hash families outside the domain of coding theory.

The rest of this paper is organized as follows. In Section 2, we apply color-coding to BUDGETED UNIQUE COVERAGE and show that it is fixed-parameter tractable. This section also contains the description of the hash families we use for derandomization. In Section 3 we consider two special cases of BUDGETED UNIQUE COVERAGE: UNIQUE COVERAGE and BUDGETED MAX CUT, and give better deterministic algorithms for these problems than the ones presented in [13]. We conclude with some open problems in Section 4. Complete proofs appear in a full version of this paper [12].

A parameterized problem is *fixed-parameter tractable (FPT)* if there is an algorithm that takes as input an instance (x, k) of the problem and correctly decides whether it is a YES or NO-instance in time $O(f(k) \cdot |x|^{O(1)})$, where f is some arbitrary function of the parameter k. When there is more than one parameter, k would represent an appropriate function (the sum or the maximum of them, for example) of the parameters. For further details and an introduction to parameterized complexity, we refer to [6,8,14]. For an integer n, by $[n]$ we denote the set $\{1, 2, \ldots n\}$. We let e denote the base of the natural logarithm (denoted by $\ln$) and $\log$ denote logarithms to base 2. We let $\mathbb{Q}$ denote the set of rationals, $\mathbb{Z}$ the set of integers, and for a number a, we use $\mathbb{Q}^{\geq a}$ to denote the set $\{x \in \mathbb{Q} : x \geq a\}$.

2 Budgeted Unique Coverage

An instance of UNIQUE COVERAGE consists of a family $\mathcal{F}$ of m subsets of a finite universe $\mathcal{U}$ of size n and a nonnegative integer k. The question is whether there exists a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ that covers k elements uniquely. An element is said to be covered uniquely by $\mathcal{F}'$ if it appears in exactly one set of $\mathcal{F}'$. An instance of BUDGETED UNIQUE COVERAGE contains, in addition to $\mathcal{U}$ and $\mathcal{F}$, a cost function $c : \mathcal{F} \to \mathbb{Z}^+$, a profit function $p : \mathcal{U} \to \mathbb{Z}^+$ and nonnegative integers k and B. The question, in this case, is whether there exists a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of total cost at most B such that the total profit of elements uniquely covered by $\mathcal{F}'$ is at least k.

In [13] it was shown that BUDGETED UNIQUE COVERAGE with arbitrarily small positive (rational) costs and profits is not fixed-parameter tractable with parameters B

and k, unless P = NP. Further, BUDGETED UNIQUE COVERAGE is $W[1]$-hard when parameterized by the budget B alone, even when cost and profit functions are integral. In this paper, we assume that both costs and profits assume positive integral values and that both B and k are parameters. Let $(\mathcal{U}, \mathcal{F}, c, p, B, k)$ be an instance of BUDGETED UNIQUE COVERAGE. We may assume that for all $S_i, S_j \in \mathcal{F}$, $i \neq j$, we have

- $S_i \neq S_j$;
- $c(S_i) \leq B$;
- $|S_i| \leq k - 1$.

For if $c(S_i) > B$ then S_i cannot be part of any solution and may be discarded; if $|S_i| \geq k$ then the given instance is trivially a YES-instance. We make these assumptions implicitly in the rest of the paper.

Demaine et al. [5] show that there exists an $\Omega(1/\log n)$-approximation algorithm for BUDGETED UNIQUE COVERAGE (Theorem 4.1). We use the same proof technique to show the following.

Lemma 1. *Let* $(\mathcal{U}, \mathcal{F}, c, p, B, k)$ *be an instance of* BUDGETED UNIQUE COVERAGE *and let* $c : \mathcal{F} \to \mathbb{Q}^{\geq 1}$ *and* $p : \mathcal{U} \to \mathbb{Q}^{\geq 1}$. *Then either*

1. *we can find in polynomial time a subfamily* $\mathcal{F}' \subseteq \mathcal{F}$ *with total cost at most* B *such that the total profit of elements uniquely covered by* $\mathcal{F}'$ *is at least* k*; or*
2. *for every subfamily* $\mathcal{H}$ *with total cost at most* B*, we have* $|\bigcup_{S \in \mathcal{H}} S| \leq 18k \log B$.

Proof. Appears in the full version [12]. □

The first step of our algorithm is to apply Step 1 of Lemma 1. From now on we assume that every subfamily of total cost at most B covers at most $18k \log B$ elements of the universe.

We now proceed to show that BUDGETED UNIQUE COVERAGE is FPT. We first show this for the case when the costs and profits are all one and then handle the more general case of integral costs and profits. Therefore let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of BUDGETED UNIQUE COVERAGE with unit costs and profits. For this version of the problem, we have to decide whether there exists a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ of size at most B that uniquely covers at least k elements. A subfamily $\mathcal{F}'$ of size at most B that uniquely covers at least k elements is called a *solution subfamily*.

To develop our color-coding algorithm, we use two sets of colors $\mathcal{C}_g$ and $\mathcal{C}_b$ with the understanding that the (good) colors from $\mathcal{C}_g$ are used for the elements that are uniquely covered and the (bad) colors from $\mathcal{C}_b$ are used for the remaining elements. In the present setting, $\mathcal{C}_g = \{1, \ldots, k\}$ and $\mathcal{C}_b = \{k + 1\}$.

Remark. For our algorithms, any subset of k colors can play the role of good colors. For ease of presentation, we fix a set of good and bad colors while describing our randomized algorithms. Our derandomized algorithms assume that any set of k colors may be good colors.

We now describe the notion of a *good configuration*. Given $h : \mathcal{U} \to \mathcal{C}_g \uplus \mathcal{C}_b$ and $\mathcal{F}' \subseteq \mathcal{F}$, define $h(\mathcal{F}') := \bigcup_{\{i \in S, S \in \mathcal{F}'\}} \{h(i)\}$ and $\mathcal{U}(\mathcal{F}') := \bigcup_{S \in \mathcal{F}'} S$.

Definition 1. *Given $h : \mathcal{U} \to \mathcal{C}_g \uplus \mathcal{C}_b$ and $\mathcal{C}'_g \subseteq \mathcal{C}_g$, we say that*

a. *$\mathcal{F}' \subseteq \mathcal{F}$ has a good configuration with respect to (wrt) h and $\mathcal{C}'_g$ if*
  1. *$h(\mathcal{F}') \cap \mathcal{C}_g = \mathcal{C}'_g$, and*
  2. *the elements of $\mathcal{U}(\mathcal{F}')$ that are assigned colors from $\mathcal{C}'_g$ have distinct colors and are uniquely covered by $\mathcal{F}'$.*

b. *$\mathcal{F}$ has a good configuration wrt h and $\mathcal{C}'_g$ if there exists a subfamily $\mathcal{F}'$ with a good configuration wrt h and $\mathcal{C}'_g$. Call $\mathcal{F}'$ a witness subfamily.*

A solution subfamily (for the unit costs and profits version) is a subfamily $\mathcal{F}' \subseteq \mathcal{F}$ with at most B sets and which uniquely covers at least k elements.

The next lemma shows that if h is chosen uniformly at random from the space of all functions $f : \mathcal{U} \to [k+1]$ and $(\mathcal{U}, \mathcal{F}, B, k)$ is a YES-instance of BUDGETED UNIQUE COVERAGE with unit costs and profits, then with high probability a solution subfamily $\mathcal{F}'$ has a good configuration wrt h and $\mathcal{C}_g$. Note that such a uniformly chosen h maps every element from $\mathcal{U}$ uniformly at random to an element in $[k+1]$.

Lemma 2. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be a* YES*-instance of* BUDGETED UNIQUE COVERAGE *with unit costs and profits and let $h : \mathcal{U} \to [k+1]$ be a function chosen uniformly at random. Then a solution subfamily $\mathcal{F}'$ has a good configuration wrt h and $\mathcal{C}_g$ with probability at least $2^{-k(18 \log B \log(k+1) - \log k + \log e)}$.*

Proof. Let $\mathcal{F}'$ be a solution subfamily with at most B sets that covers the elements $Q = \{i_1, \ldots, i_k\}$ uniquely. Then $p := |\mathcal{U}(\mathcal{F}')| \leq 18k \log B$, by Lemma 1. To complete the proof, we show that $\mathcal{F}'$ has a good configuration with respect to h and $\mathcal{C}_g$ with probability at least $2^{-k(18 \log B \log(k+1) - \log k + \log e)}$. For $\mathcal{F}'$ to have a good configuration, we must have $h(i) = k+1$ for all $i \in \mathcal{U}(\mathcal{F}') \setminus Q$ and $h(i_1), \ldots, h(i_k)$ a permutation of $1, \ldots, k$. The probability **Pr** that this happens is:

$$\begin{aligned}\mathbf{Pr} = \tfrac{1}{(k+1)^{|\mathcal{U}(\mathcal{F}')\setminus Q|}} \times \tfrac{k!}{(k+1)^k} &\geq \left(\tfrac{k}{e}\right)^k \tfrac{1}{(k+1)^p} = e^{k\ln(k/e) - p\ln(k+1)} \\ &\geq e^{k\ln(k/e) - 18k\log B \ln(k+1)} \geq 2^{-k(18\log B \log(k+1) - \log k + \log e)}\end{aligned}$$ □

Given a coloring h, how do we find out whether $\mathcal{F}$ has a good configuration wrt h and $\mathcal{C}_g$? We answer this next.

Finding a good configuration. Observe that if $\mathcal{F}$ has a good configuration wrt h and $\mathcal{C}_g$, then any witness subfamily $\mathcal{F}'$ covers at least k elements uniquely. To locate such a family of size at most B we use dynamic programming over subsets of $\mathcal{C}_g$. To this end, let W be a $2^k \times B$ array where we identify the rows of W with subsets of $\mathcal{C}_g$ and the columns with the size of a subfamily. For a fixed coloring function h, a subset $\mathcal{C}'_g \subseteq \mathcal{C}_g$ and $1 \leq i \leq B$, define $W[\mathcal{C}'_g][i]$ as follows:

$$W[\mathcal{C}'_g][i] = \begin{cases} 1, & \text{if there exists } \mathcal{F}' \subseteq \mathcal{F}, \text{ with } |\mathcal{F}'| \leq i, \text{ with a good con-} \\ & \text{figuration wrt } \mathcal{C}'_g \text{ and } h. \\ 0, & \text{otherwise.} \end{cases}$$

The entry corresponding to $W[\emptyset][i]$ is set to 1 for all $1 \leq i \leq B$, as a convention. We fill this array in increasing order of the sizes of subsets of $\mathcal{C}_g$. Let $\mathcal{T}$ be the family of all sets $S \in \mathcal{F}$ such that $h(S) \cap \mathcal{C}_g \subseteq \mathcal{C}'_g$. Let $g(S)$ denote the set of good colors used in S. Then $W[\mathcal{C}'_g][i] = \bigvee_{S \in \mathcal{T}} W[\mathcal{C}'_g \setminus g(S)][i-1]$.

The correctness of the algorithm is immediate. Clearly if $W[\mathcal{C}_g][B] = 1$, then a subfamily with at most B sets that uniquely covers at least k elements exists, and can be found out by simply storing the witness families $\mathcal{F}'$ for every entry in the table and backtracking. The time taken by the algorithm is $O(2^k Bmk)$, since the size of the array is $2^k B$ and each entry of the array can be filled in time $O(mk)$, where $m = |\mathcal{F}|$.

Lemma 3. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of* Budgeted Unique Coverage *with unit costs and profits and $h : \mathcal{U} \to \mathcal{C}$ a coloring function. Then we can find a subfamily $\mathcal{F}'$ of size at most B which has a good configuration wrt h and $\mathcal{C}_g$, if there exists one, in time $O(2^k Bmk)$.*

A randomized algorithm for Budgeted Unique Coverage with unit costs and profits is as follows.

1. Randomly choose a coloring function $h : \mathcal{U} \to \{1, \ldots, k+1\}$.
2. Apply Lemma 3 and check whether there exists a family $\mathcal{F}'$ of size at most B that is witness to a good configuration wrt h and $\mathcal{C}_g$. If such a family exists, return YES, else go to Step 1.

By Lemma 2, if the given instance is a YES-instance, the probability that a solution subfamily $\mathcal{F}'$ has a good configuration wrt a randomly chosen function $h : \mathcal{U} \to \mathcal{C}$ and $\mathcal{C}_g$ is at least $2^{-k(18 \log B \log(k+1) - \log k + \log e)}$. By Lemma 3, we can find such a subfamily in time $O(2^k Bmk)$.

Theorem 1. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of* Budgeted Unique Coverage *with unit costs and profits. There exists a randomized algorithm that finds a subfamily $\mathcal{F}'$ of size at most B covering at least k elements uniquely, if there exists one, in expected $O(2^{18k \log B \log(k+1)} \cdot Bmk)$ time.*

2.1 Improving the Run-Time

It is clear that if a solution subfamily $\mathcal{F}'$ is to have a good configuration wrt a randomly chosen coloring function h and $\mathcal{C}_g$, then h must assign all the non-uniquely covered elements of $\mathcal{F}'$ the color in $\mathcal{C}_b$. Intuitively, if we increase the number of colors in $\mathcal{C}_b$, we increase the probability that a specific target subfamily has a good configuration wrt a randomly chosen coloring function. We formalize this intuition below. We need the following inequality whose proof we omit.

Lemma 4. *For all $t \geq 2k$, $\left(\frac{t-k}{t}\right)^t \geq (2e)^{-k}$.*

Lemma 5. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be a* YES*-instance of* Budgeted Unique Coverage *with unit costs and profits; let $\mathcal{C}_g = [k]$, $\mathcal{C}_b = \{k+1, \ldots, q\}$ and $\mathcal{C} = [q]$ so that $q \geq 2k$. If $h : \mathcal{U} \to \mathcal{C}$ is chosen uniformly at random then every solution subfamily $\mathcal{F}'$ with p elements of the universe has a good configuration wrt h and $\mathcal{C}_g$ with probability at least $e^{-k}\left(\frac{k}{q-k}\right)^k (2e)^{-\frac{kp}{q}}$.*

Proof. Let the set of elements uniquely covered by $\mathcal{F}'$ be $Q = \{i_1, \ldots, i_k\}$. For $\mathcal{F}'$ to have a good configuration, the function h must map every element of $\mathcal{U}(\mathcal{F}') \setminus Q$ to $\mathcal{C}_b$ and map Q to $\mathcal{C}_g$ injectively. Therefore the probability **Pr** that $\mathcal{F}'$ has a good configuration wrt $\mathcal{C}_g$ and a randomly chosen h is:

$$\mathbf{Pr} = \tfrac{(q-k)^{p-k}}{q^{p-k}} \times \tfrac{k!}{q^k} \geq \left(\tfrac{q-k}{q}\right)^p \left(\tfrac{1}{q-k}\right)^k k^k e^{-k}$$
$$\geq e^{-k}\left(\tfrac{k}{q-k}\right)^k \left(1 - \tfrac{k}{q}\right)^p \geq e^{-k}\left(\tfrac{k}{q-k}\right)^k (2e)^{-\frac{kp}{q}} \text{ (by Lemma 4).} \qquad \square$$

If $(\mathcal{U}, \mathcal{F}, B, k)$ is a YES-instance of BUDGETED UNIQUE COVERAGE with unit costs and profits then $p \leq 18k \log B$. Also observe that $B \geq 2$, for otherwise the given instance is a NO-instance. Setting $p = 18k \log B$ and $q = k + p$ in Lemma 5 we can show that a solution subfamily $\mathcal{F}'$ has a good configuration wrt a randomly chosen coloring function h and $\mathcal{C}_g$ with probability at least $2^{-8.2k - k \log\log B}$. Combining this with Lemma 3, we obtain:

Theorem 2. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of* BUDGETED UNIQUE COVERAGE *with unit costs and profits. Then we can find a subfamily $\mathcal{F}'$ of size at most B covering at least k elements uniquely, if there exists one, in $O(2^{9.2k + k \log\log B} \cdot Bmk)$ expected time.*

2.2 Derandomization

We now discuss how to derandomize the algorithms described in the last subsection. In general, randomized algorithms based on the color-coding method are derandomized using a suitable family of hash functions or "universal sets". We need a family of functions from $\mathcal{U}$ to $[t]$, where $t \geq k+1$, such that for all $S \subseteq \mathcal{U}$ of size $s = \lceil 18k \log B \rceil$ and all $X \subseteq S$ of size k, there exists a function h in the family which maps X injectively and the colors it assigns to the elements in $S \setminus X$ are different from the ones it assigns to those in X.

Such hash families are called (k, s)-hash families (with domain $[n]$ and range $[t]$) and they were introduced by Barg et al. [3] in the context of particular class of codes called parent identifying codes. At this point, we recall the definition of an (n, t, s)-*perfect hash family*. A family $\mathcal{H}$ of functions from $[n]$ to $[t]$ is called an (n, t, s)-perfect hash family if for every subset $X \subseteq [n]$ of size s, there is a function $h \in \mathcal{H}$ that maps X injectively. Note that an (n, t, s)-perfect hash family is a (k, s)-hash family with domain $[n]$ and range $[t]$, and a (k, s)-hash family with domain $[n]$ and range $[t]$ is an (n, t, k)-perfect hash family. Therefore (k, s)-hash families may be thought of as standing in between k-perfect and s-perfect hash families.

Our deterministic algorithm simply uses functions from these families $\mathcal{H}$ for coloring and is described below. Given an instance $(\mathcal{U}, \mathcal{F}, B, k)$ of BUDGETED UNIQUE COVERAGE with unit costs and profits, we let $n = |\mathcal{U}|$, $\mathcal{C} = [t]$, and s to be the closest integer to our estimate in Lemma 1, which is $O(k \log B)$.

```
Deterministic Algorithm
for each $h \in \mathcal{H}$ do
    for each subset $X \subseteq \mathcal{C}$ of size $k$ do
        1. Define $\mathcal{C}_g = X$ and $\mathcal{C}_b = \mathcal{C} \setminus X$;
        2. Apply Lemma 3 and check whether there exists a subfamily $\mathcal{F}'$ of
           size at most $B$ which has a good configuration wrt $\mathcal{C}_g$ and $h$;
        3. if yes, then return the corresponding $\mathcal{F}'$;
return NO;
```

The correctness of the algorithm follows from the description—if a witness subfamily for the given $\mathcal{F}$ exists, at least one $h \in \mathcal{H}$ will color all the uniquely covered elements of the witness subfamily distinctly, thereby resulting in a good configuration. The running time of the algorithm is $O\left(|\mathcal{H}| \cdot \binom{t}{k} \cdot 2^k Bmk\right)$.

Alon et al. [1] provide explicit constructions of (k, s)-hash families when the range is $k+1$ and ks, respectively.

Theorem 3 (Alon et al. [1]). *There exists an absolute constant $c > 0$ such that for all $2 \leq k < s$ there is an explicit construction of a (k, s)-hash family $\mathcal{H}$ with domain $[n]$ and range $[k+1]$ of size at most $2^{ck \log s} \cdot \log_{k+1} n$. When the range is $[ks]$, there exists an explicit construction of a (k, s)-hash family of size $O(k^2 s^2 \log n)$.*

If $t = k+1$, then by the above theorem, the running time of our deterministic algorithm is $O(2^{O(k \log k + k \log\log B)} \cdot Bmk \cdot \log n)$; when $t = ks$, the running time works out to be $O(2^{O(k \log k + k \log\log B)} \cdot mk^5 \cdot B \log^2 B \cdot \log n)$.

Theorem 4. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of* BUDGETED UNIQUE COVERAGE *with unit costs and profits. Then we can find a subfamily $\mathcal{F}'$ of size at most B covering at least k elements uniquely, if there exists one, in time $O(2^{O(k \log k + k \log\log B)} \cdot Bmk \cdot \log n)$.*

We next give alternate running time bounds using standard s-perfect hash families for derandomizing our algorithm.

Theorem 5. ([2,15,4]) *There exist explicit constructions of (n, t, s)-perfect hash families of size $2^{O(s)} \log n$ when $t = s$, and of size $s^{O(1)} \log n$ when $t = s^2$. In fact, for the case $t = s$, an explicit construction of an s-perfect hash family of size $6.4^s \log^2 n$ in time $6.4^s n \log^2 n$ is known.*

For $t = s$, using the construction of s-perfect hash families by Chen et al. [4], we obtain a running time of $O(6.4^s \log^2 n \cdot \binom{s}{k} \cdot 2^k \cdot Bkm)$. Since $s = O(k \log B)$, this expression simplifies to of $O(2^{O(k \log B)} \cdot \log^2 n \cdot Bmk)$. For $t = s^2$, we can use a hash family of size $s^{O(1)} \log n$ [2], and the expression for the running time then works out to be $O(2^{O(k \log k + k \log\log B)} \cdot \log n \cdot Bmk)$. We thus have

Theorem 6. *Let $(\mathcal{U}, \mathcal{F}, B, k)$ be an instance of* BUDGETED UNIQUE COVERAGE *with unit costs and profits. Then we can find a subfamily $\mathcal{F}'$ of size at most B covering at least k elements uniquely, if there exists one, in time $O(f(k, B) \cdot \log^2 n \cdot Bmk)$, where $f(k, B) = \min\{2^{O(k \log B)}, 2^{O(k \log k + k \log\log B)}\}$.*

If we ignore constants, Theorem 6 gives a run-time which is at least as good as that in Theorem 4.

We now consider existential results concerning hash families. The following is known about (n, t, s)-hash families.

Theorem 7 **([11]).** *For all positive integers $n \geq t \geq s \geq 2$, there exists an (n, t, s)-perfect hash family $\Delta(n, t, s)$ of size $e^{s^2/t} s \ln n$.*

Alon et al. [1] provided existential bounds for (k, s)-hash functions for the case when $t = k + 1$. If we assume that $s \geq 2k$, then their existential bound works out to $(2e)^s \cdot e^{k \ln s} \cdot s \ln n$. In the lemmas that follow, we provide existential bounds for an arbitrary range.

Lemma 6. *Let $k \leq s \leq n$ be positive integers and let $t \geq 2k$ be an integer. There exists a (k, s)-hash family $\mathcal{H}$ with domain $[n]$ and range $[t]$ of size $(2e)^{sk/t} \cdot s \log n$.*

Proof. Let $\mathcal{A} = \{h : [n] \to [t]\}$ be the set of all functions from $[n]$ to $[t]$. For $h \in \mathcal{A}$, $S \subseteq [n]$ of size s and $X \subseteq S$ of size k, define h to be (X, S)-*hashing* if h maps X injectively such that $h(X) \cap h(S \setminus X) = \emptyset$ and *not* (X, S)-*hashing* otherwise.

Fix $S \subseteq [n]$ of size s and $X \subseteq S$ of size k. The probability **Pr** that a function h picked uniformly at random from $\mathcal{A}$ is (X, S)-hashing, is given by:

$$\begin{aligned}\mathbf{Pr} &= \frac{\binom{t}{k} k! (t-k)^{s-k}}{t^s} > \left(\frac{t}{k}\right)^k \cdot \left(\frac{k}{e}\right)^k \cdot \frac{1}{t^k} \cdot \left(\frac{t-k}{t}\right)^{s-k} \\ &= \frac{1}{e^k} \left(\frac{t-k}{t}\right)^{s-k} \geq \frac{1}{e^k} \cdot \left(\frac{1}{2e}\right)^{k(s-k)/t} \qquad \text{(By Lemma 4.)} \\ &\geq \left(\frac{1}{2e}\right)^{ks/t}.\end{aligned}$$

The probability that the function h is not (X, S)-hashing is less than $1 - (2e)^{-ks/t}$. If we pick N functions uniformly at random from $\mathcal{A}$ then the probability that none of these functions is (X, S)-hashing is less than $(1 - (2e)^{-ks/t)})^N$. The probability that none of these N functions is (X, S)-hashing for some (S, X) pair is less than $\binom{n}{s}\binom{s}{k}(1 - (2e)^{-ks/t})^N$, which in turn is less than $n^s(1 - (2e)^{-ks/t})^N$. For this family of N functions to be (X, S)-hashing for every (S, X) pair, we would want $n^s(1 - (2e)^{-ks/t})^N$ to be at most one. A simple calculation yields that this will hold when $N \geq (2e)^{ks/t} s \log n$. □

Lemma 7. *Let $k \leq s \leq n$ be positive integers and let $t \geq k + 1$. There exists a (k, s)-hash family $\mathcal{H}$ with domain $[n]$ and range $[t]$ of size $2^{O(k \log(s/k))} \cdot s \log n$.*

Proof. Let $F = \Delta(n, m, s)$, the (n, m, s)-perfect hash family obtained from Theorem 7, where we set $m = \lceil s^2/(k \log(s/k)) \rceil$. Let G be a family of functions g_X from $[m]$ to $[t]$, indexed by k-element subsets X of $[m]$ as follows. The function g_X maps X in an one-one, onto fashion to $\{1, \ldots, k\}$ and maps an element of $[m] - X$ to an arbitrary element in $\{k + 1, \ldots, t\}$. Our required family T of functions from $[n]$ to $[t]$ is obtained by composing the families F and G. It is easy to see that T is an s-discriminating (n, t, k)-perfect hash family and has the claimed bound for its size. □

Note that Lemma 6 requires that $t \geq 2k$ and that for Lemma 7 we have no restriction on t. Also observe that had we an explicit construction of a (k, s)-hash family satisfying the bound in Lemma 6, then by setting $s = t = O(k \log B)$, we would have obtained a running time of $O(2^{O(k \log \log B)} \cdot s \log n)$ which is significantly better than that given in Theorem 6. We believe that this is motivation for studying explicit constructions of (k, s)-hash families for an arbitrary range.

2.3 Generalized Costs and Profits

The dynamic programming procedure described for the case of unit profits and costs can be scaled to handle the more general case when costs are positive integers and profits rational numbers ≥ 1 and vice versa. The modifications required are omitted from this extended abstract. We obtain the following result analogous to Theorem 6:

Theorem 8. *Let* $(\mathcal{U}, \mathcal{F}, B, k)$ *be an instance of* BUDGETED UNIQUE COVERAGE *with either integral costs and rational profits* ≥ 1 *or with rational costs* ≥ 1 *and integral profits. Then one can find a subfamily* $\mathcal{F}'$ *of total cost at most* B *that uniquely covers elements with total profit at least* k*, if there exists one, in time* $O(f(k, B) \cdot Bmk \log^2 n)$*, where* $f(k, B) = \min\{2^{O(k \log B)}, 2^{O(k \log k + k \log \log B)}\}$.

3 Faster Deterministic Algorithms for Special Cases

We now present faster deterministic algorithms than the ones presented in [13] for two special cases of BUDGETED UNIQUE COVERAGE: UNIQUE COVERAGE (the unbudgeted version) and BUDGETED MAX CUT.

3.1 Unique Coverage

An instance $(\mathcal{U}, \mathcal{F}, k)$ of UNIQUE COVERAGE can be viewed as an instance of BUDGETED UNIQUE COVERAGE where the costs and profits are all one and the budget $B = k$ as we do not need more than k sets to cover k elements uniquely. Using Theorem 6, we immediately obtain an algorithm with run-time $O(2^{O(k \log k)} \cdot |\mathcal{F}| \cdot k^2 \log n)$. In this subsection we present an algorithm for UNIQUE COVERAGE that runs in deterministic $O(2^{O(k \log \log k)} \cdot |\mathcal{F}| \cdot k + |\mathcal{F}|^2)$ time beating the $O(4^{k^2} \cdot |\mathcal{F}|)$ algorithm in [13]. We first need some lower bounds on the number of elements that can be uniquely covered in any instance of UNIQUE COVERAGE.

Define the *frequency* f_u of an element $u \in \mathcal{U}$ to be the number of sets in the family $\mathcal{F}$ that contain u. Let γ denote the maximum frequency, that is, $\gamma = \max_{u \in \mathcal{U}}\{f_u\}$.

Lemma 8. *There exists a subfamily* $\mathcal{F}' \subseteq \mathcal{F}$ *such that* $\mathcal{F}'$ *covers at least* $n/(4e \log \gamma)$ *elements uniquely. Furthermore, such a subfamily can be found in polynomial time.*

Proof. Similar to the proof of Lemma 1 and appears in the full version [12]. $\square$

Lemma 9. *Let* $M = \max_{S \in \mathcal{F}}\{|S|\}$. *Then there exists a subfamily* $\mathcal{F}'$ *that covers at least* $n/(8e \log M)$ *elements uniquely. Furthermore, such a subfamily can be found in polynomial time.*

Proof. We begin by constructing a subfamily $\mathcal{F}'$ from $\mathcal{F}$ that is *minimal* in the sense that every set in $\mathcal{F}'$ covers at least one element in $\mathcal{U}$ uniquely. Such a subfamily is easily obtained, by going over every set in the family and checking if it has at least one element which is not contained in any other set. Let m' denote the size of the subfamily $\mathcal{F}'$. For the proof of the lemma we distinguish two cases based on m':

Case 1: $m' \geq n/2$. As the subfamily is minimal, by construction, we are immediately able to cover at least $n/2$ elements uniquely. Thus $\mathcal{F}'$ itself satisfies the claim of the lemma.

Case 2: $m' < n/2$. In this case, we first claim that $|\{u \in \mathcal{F}' : f(u) < M\}| \geq n/2$. If not, then there would be more than $n/2$ elements whose frequency is at least M, which implies that $\sum_{S \in \mathcal{F}'} |S| > Mn/2$. On the other hand, $\sum_{S \in \mathcal{F}'} |S|$ is clearly at most $M(n/2 - 1)$ (because there are strictly less than $n/2$ sets in the family and the size of any set in the family is bounded by M). The claim implies that there exists a set of at least $n/2$ elements whose frequency is less than M. Denote this set of elements by $\mathcal{V}$. Consider the family $\mathcal{F}''$ obtained from $\mathcal{F}'$ as follows: $\mathcal{F}'' = \{S \cap \mathcal{V} \mid S \in \mathcal{F}'\}$. Applying Lemma 8 to the instance $(\mathcal{V}, \mathcal{F}'')$, we obtain a subfamily $\mathcal{T}$ of $\mathcal{F}''$ that covers at least $n/(8e \log M)$ elements uniquely. The corresponding subfamily of $\mathcal{F}'$ will clearly cover the same set of elements uniquely in $\mathcal{U}$. This completes the proof of the lemma. □

Using these lower bounds on the number of elements that are uniquely covered, we can upper bound the size of a YES-instance of the UNIQUE COVERAGE problem as a function of the parameter k. Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of UNIQUE COVERAGE. If $k \leq n/8e\log(k-1)$, then there exists a subfamily that covers k elements uniquely. If not, we have $k > n/8e \log k$, which implies that $n < 8ek \log k$.

Lemma 10. *Let $(\mathcal{U}, \mathcal{F}, k)$ be an instance of* UNIQUE COVERAGE. *Then, in polynomial time, we can either find a subfamily covering at least k elements uniquely, or an equivalent instance where the size of the universe is $O(k \log k)$.*

An improved algorithm for UNIQUE COVERAGE first applies Lemma 10 and obtains an instance of UNIQUE COVERAGE, $(\mathcal{U}, \mathcal{F}, k)$, where $n = |\mathcal{U}| \leq O(k \log k)$. Now we examine all k-sized subsets X of the universe $\mathcal{U}$ and check whether there exists a subfamily that covers it uniquely. Let $X = \{u_{i_1}, u_{i_2}, \ldots, u_{i_k}\}$, and let h be a function that maps X injectively to $\{1, \ldots, k\}$ and each element in $\mathcal{U} \setminus X$ to the color $k+1$. Applying Lemma 3 to the instance $(\mathcal{U}, \mathcal{F}, B = k, k)$, with the coloring function h described above gives us an algorithm to find the desired $\mathcal{F}'$ in time $O(2^k k^2 m)$. Note that a factor of k can be avoided by directly applying dynamic programming over subsets of X. The size of $\mathcal{U}$ is upper bounded by $8ek \log k$ and hence the total number of subsets that need to be examined is at most $\binom{8ek \log k}{k}$, which is bounded above by $(8e \log k)^k \leq 2^{4.5k + k \log\log k}$. Combining this with the above discussion results in:

Theorem 9. *Given an instance $(\mathcal{U}, \mathcal{F}, k)$ of* UNIQUE COVERAGE, *one can find a subfamily that uniquely covers at least k elements, if there exists one, in time $O(f(k) \cdot mk + m^2)$, where $f(k) = 2^{5.5k + k \log\log k}$.*

3.2 Budgeted Max Cut

An instance of BUDGETED MAX CUT consists of an undirected graph $G = (V, E)$ on n vertices and m edges; a cost function $c : V \to \mathbb{Z}^+$; a profit function $p : E \to \mathbb{Z}^+$; and positive integers k and B. The question is whether there exists a cut $(T, V - T)$, $\emptyset \neq T \neq V$, such that the total cost of the vertices in T is at most B and the total profit of the edges crossing the cut is at least k. This problem can be modelled as an instance BUDGETED UNIQUE COVERAGE by taking $\mathcal{U} = E$ and $\mathcal{F} = \{S_v : v \in V\}$, where $S_v = \{e \in E : e \text{ is incident on } v\}$.

In [13], an algorithm with run-time $O((B^2 \cdot k \cdot 2^k)^{\min\{B,k\}} \cdot m^{O(1)})$ was described for BUDGETED MAX CUT. Here we develop an algorithm with run-time $O(2^{O(k)} \cdot Bmk \cdot \log^2 n)$. Given $S \subseteq V$, we let $c(S)$ denote the total cost of the elements of S. If $(S, V - S)$ is a cut in a graph G, then $p(S, V - S)$ is the total profit of edges across the cut. Define the profit $\hat{p}(v)$ of a vertex v to be the sum of the profits of all the edges incident on it.

Lemma 11. *If (G, B, k, c, p) is a* YES*-instance of* BUDGETED MAX CUT *then there exists a cut $(S, S - V)$ such that $c(S) \leq B$, $p(S, V - S) \geq k$, and $|\bigcup_{v \in S} S_v| \leq 4k$.*

Proof. Since we are given a YES-instance of the problem, there exists a cut (T, T') such that $c(T) \leq B$ and $p(T, T') \geq k$. Call a vertex v of T *redundant* if $p(T - v, T' \cup v) \geq k$. From (T, T'), obtain a cut (S, S') such that $S \subset T$ and S does not contain any redundant vertices. Observe that $c(S) \leq B$ and $p(S, S') \geq k$. For any $v \in S$, $\hat{p}(v) \leq k - 1$ and $p(S - v, S' \cup v) \leq k - 1$. Therefore $p(S, S') \leq 2k$. For $v \in S$, partition S_v as $I_v \uplus C_v$, where I_v is the set of edges incident on v that lie entirely in S and C_v are the edges that lie across the cut (S, S'). Clearly $p(I_v) \leq p(C_v)$, for otherwise, $p(S - v, S' \cup v) > p(S, S')$, a contradiction to the fact that S has no redundant vertices. Therefore $\sum_{v \in S} p(I_v) \leq \sum_{v \in S} p(C_v) \leq 2k$. This yields $\sum_{v \in S} \hat{p}(v) \leq 4k$. Since the profits are at least one, we have $|\bigcup_{v \in S} S_v| \leq 4k$. $\square$

We use the deterministic algorithm outlined before Theorem 6 with $t = s = 4k$ and a $4k$-uniform perfect hash family by Chen et al. [4]. The running time then works out to $O(6.4^{4k} \log^2 n \cdot \binom{4k}{k} \cdot 2^k Bmk)$ which simplifies to $O(2^{13.8k} \cdot Bmk \cdot \log^2 n)$.

Theorem 10. *Let (G, B, k, c, p) be an instance of* BUDGETED MAX CUT*. Then we can find a cut (S, S') such that $c(S) \leq B$ and $p(S, S') \geq k$, if there exists one, in time $O(2^{13.8k} \cdot Bmk \cdot \log^2 n)$.*

4 Conclusions

In this paper we gave fixed-parameter tractable algorithms for BUDGETED UNIQUE COVERAGE and several of its variants. Our algorithms were based on an application of the well-known method of color-coding. Our randomized algorithms have good running times but the deterministic algorithms make use of either (k, s)-hash families or perfect hash families and this introduces large constants in the running times, a common enough phenomenon when derandomizing randomized algorithms using such function families [2]. Our use of (k, s)-hash families to derandomize our algorithms is perhaps the first application outside the domain of coding theory and it suggests the importance of such hash families. It will be interesting to explicitly construct (k, s)-hash families of size promised by Lemma 6 and explore other applications of our generalization of the color-coding technique.

Acknowledgements. We thank Srikanth Srinivasan for discussions about Lemma 6.

References

1. Alon, N., Cohen, G., Krivelevich, M., Litsyn, S.: Generalized hashing and parent-identifying codes. Journal of Combinatorial Theory Series A 104(1), 207–215 (2003)
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM 42(4), 844–856 (1995)
3. Barg, A., Cohen, G., Encheva, S., Kabatiansky, G., Zémor, G.: A hypergraph approach to the identifying parent property: The case of multiple parents. SIAM Journal of Discrete Mathematics 14(3), 423–431 (2001)
4. Chen, J., Lu, S., Sze, S.-H., Zhang, F.: Improved algorithms for path, matching, and packing problems. In: SODA, pp. 298–307 (2007)
5. Demaine, E.D., Hajiaghayi, M.T., Feige, U., Salavatipour, M.R.: Combination can be hard: approximability of the unique coverage problem. In: SODA, pp. 162–171 (2006)
6. Downey, R.G., Fellows, M.R.: Parameterized complexity. Springer, New York (1999)
7. Erlebach, T., van Leeuwen, E.J.: Approximating geometric coverage problems. In: SODA, pp. 1267–1276 (2008)
8. Flum, J., Grohe, M.: Parameterized Complexity Theory. Texts in Theoretical Computer Science. An EATCS Series. Springer, Berlin (2006)
9. Guruswami, V., Trevisan, L.: The complexity of making unique choices: Approximating 1-in-k sat. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX 2005 and RANDOM 2005. LNCS, vol. 3624, pp. 99–110. Springer, Heidelberg (2005)
10. Khuller, S., Moss, A., Naor, J.: The budgeted maximum coverage problem. Inf. Process. Lett. 70(1), 39–45 (1999)
11. Mehlhorn, K.: On the program size of perfect and universal hash functions. In: Proceedings of the 23th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 170–175. IEEE, Los Alamitos (1982)
12. Moser, H., Misra, N., Raman, V., Saurabh, S., Sikdar, S.: The parameterized complexity of the Unique Coverage problem (2009) (manuscript)
13. Moser, H., Raman, V., Sikdar, S.: The parameterized complexity of the unique coverage problem. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 621–631. Springer, Heidelberg (2007)
14. Niedermeier, R.: Invitation to fixed-parameter algorithms. Oxford Lecture Series in Mathematics and its Applications, vol. 31. Oxford University Press, Oxford (2006)
15. Schmidt, J.P., Siegel, A.: The spatial complexity of oblivious k-probe hash functions. SIAM J. Computing 19(5), 775–786 (1990)

Formal Verification of Gate-Level Computer Systems

Mark Hillebrand[1,⋆] and Sergey Tverdyshev[2,⋆]

[1] German Research Center for Artificial Intelligence (DFKI GmbH), Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
mah@dfki.de

[2] Saarland University, Dept. of Computer Science, 66123 Saarbrücken, Germany
deru@wjpserver.cs.uni-sb.de

Abstract. We present the formal verification of a gate-level computer system, in which a complex processor and external devices run in parallel. The system specification is an instruction set architecture with concurrently running visible devices. To the best of our knowledge this is the first formal treatment of integrating devices into a gate-level computer system.

1 Introduction

Modern computer systems are used in many safety and security critical contexts, e.g. cars or planes, where errors can lead to huge costs and loss of life. Even if sets of components (hardware or software) are functionally correct, bugs may show up when integrating the components into a single system. The perhaps most famous example of such an integration bug led to the crash of an Ariane 5 rocket; it was caused by an integer overflow in a software component that was correct for the predecessor rocket but had its assumptions violated in the new one [1]. Pervasive verification [2,3] attempts to verify systems completely including the interaction of all components, thus minimizing the set of system assumptions.

The basis of a pervasive verification is a verified target architecture. In this paper we present the implementation and verification of a complex gate-level computer system, which consists of a pipelined processor and a number of memory mapped I/O devices. The gate-level system is verified against an instruction set architecture model as seen by an assembly programmer. This model consists of a processor specification (describing the effects of instruction execution) and specifications of the devices. While at the gate-level the processor and the devices run in parallel, at the architecture level they run concurrently, i.e. perform execution steps non-deterministically one after each other.

Context and Related Work. The hardware computer system described in this paper forms the basis of a verified system stack developed and verified in the Verisoft project [2,4]. The hardware core is based on the VAMP processor [5,6],

⋆ This work was supported by the German Federal Ministry of Education and Research (BMBF) in the Verisoft project under grant 01 IS C38.

which is, to the best of our knowledge, the most complex verified processor in the open literature. The VAMP processor was first verified in the interactive theorem prover PVS. Because the rest of the Verisoft systems stack is developed in Isabelle/HOL we developed and verified the VAMP processor again in Isabelle/HOL to guarantee the pervasiveness of our verification effort. While we reused the old top-level proof strategy, we simultaneously succeeded in speeding up the verification by solving lower-level proof goals with our automatic hardware verification environment IHaVeIt integrated into Isabelle [7]. We could decrease user interaction by ca. 40% compared to the purely interactive proofs in PVS.

The closest related work in systems verification is the famous CLI stack [8]. This stack consists of a non-pipelined processor, an assembler, a compiler for a simple high-level language, and an elementary operating system kernel, but is lacking external devices. In 2002 J S. Moore [3], the principal researcher of the CLI stack project, declared the formal verification of a computer system with devices a grand challenge.

Devices are often modeled and verified as stand-alone systems [9, 10]. For instance, a device is modeled at some level and then certain model properties are checked. To use devices in a hardware / software system, though, they have to be modeled at different levels, as we do here.

Hillebrand et al. [11] previously presented the *paper-and-pencil* formalisations of a system with a hard disk drive. They define the system at the gate and the assembly level and sketch correctness arguments relating these models. The arguments, however, are complicated and depend on the concrete device model. By changing the sampling of external interrupt in the implementation we obtain a generic and much cleaner (and formally verified) solution (cf. Sect. 3).

Alkassar et al. [12] present a concurrent assembly-level model for a processor with I/O memory mapped devices and some paper-and-pencil formalizations. The model is used in [13] to formally prove the correctness of a hard disk driver at the assembly level. Moreover, in the Verisoft project the assembly model has been connected to the machine-code model presented here [4].

2 Assembly-Level Computer System

We define the instruction set architecture (ISA) of a processor with a number of memory-mapped devices D_i (cf. Fig. 1). Compared to regular ISA definitions in addition to the processor state, the combined architecture also includes device states. Processor and devices may interact (i) by the processor accessing devices and (ii) by the device causing interrupts. Devices can also make steps on their own when interacting with an external environment (e.g. a network). Therefore we model the computation of ISA with devices as a concurrent computation.

Processor. A processor configuration c_{P} is a tuple consisting of (i) two program counters $c_{\mathrm{P}}.pc$ and $c_{\mathrm{P}}.dpc$ implementing delayed branching, (ii) general purpose, floating point, and special purpose register files $c_{\mathrm{P}}.gpr$, $c_{\mathrm{P}}.fpr$, $c_{\mathrm{P}}.spr$, and (iii) a byte addressable memory $c_{\mathrm{P}}.m$.

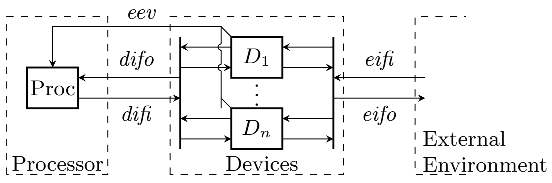


Fig. 1. Overview of a computer system with processor and devices

Devices are mapped into the processor memory and can be accessed by the processor via regular read and write operations. In addition devices can signal an interrupt to the processor via an external event signal (cf. Fig. 1).

Let DA denote the set of memory addresses mapping to devices, which are disjoint from regular physical memory addresses. The processor indicates an access to a device via the device interface input *difi* and receives the device's response on the device interface output *difo*.

Formally, let the predicates $lw(c_\mathrm{P})$ and $sw(c_\mathrm{P})$ indicate load and store word instructions and let $ea(c_\mathrm{P})$ and $RD(c_\mathrm{P})$ denote the address and affected processor register for such operations (see Müller and Paul [14] for full definitions).

The device interface input has the following four components: (i) the read flag $\mathit{difi}.rd = lw(c_\mathrm{P}) \wedge ea(c_\mathrm{P}) \in DA$ is set for a load from a device address, (ii) the write flag $\mathit{difi}.wr = sw(c_\mathrm{P}) \wedge ea(c_\mathrm{P}) \in DA$ is set for a store to a device address, (iii) the address $\mathit{difi}.a = ea(c_\mathrm{P})$ is set to the effective address, with $ea[14:12]$ specifying the accessed device and $ea[11:2]$ specifying the accessed device port (we support up to eight devices with up to 1024 ports of width 32 bits), and finally (iv) the data input $\mathit{difi}.din = c_\mathrm{P}.gpr[RD(c_\mathrm{P})]$ is set to the store operand.

The device interface output $\mathit{difo} \in \{0,1\}^{32}$ contains the device's response for a load operation on a device.

The processor model is defined by the output function Ω_P and the next state function Δ_P. The former takes a processor state c_P and computes a device input *difi* to the device as defined above. The latter takes a processor state c_P, a device output *difo*, and an external event vector *eev* (where $eev[i]$ is set iff device D_i signals an interrupt). It returns the next processor state c'_P.

The configurations of all devices are represented as a mapping c_D from device indices i to the corresponding device configuration.

Our device model is sequential in the sense that a device may progress either due to a processor access or an input from the external environment. To distinguish both cases we extend the set of device indices by the processor index P and denote this set by PD.

The device transition function Δ_D specifies the interaction of the devices with the processor and the external environment. It takes a processor-device index $idx \in PD$, an input from the external environment *eifi*, an input from the processor *difi*, and a combined device configuration c_D. It returns a new device configuration c_D', an output to the processor *difo*, and an external output *eifo*.

Depending on the input index *idx* and the device input *difi*, the transition function Δ_D is defined according to the following three cases: (i) If $idx \neq P$, the device *idx* makes step with the external input *eifi* and the given *difi* is ignored. (ii) If $idx = P \wedge (difi.wr \vee difi.rd)$, a device step is triggered by a processor access. In this case Δ_{D} ignores the given *eifi* and produces an arbitrary *eifo*. The accessed device and the access-type is specified by the given *difi*. (iii) Otherwise the processor does not access any device and Δ_{D} does nothing. The device output function Ω_{D} computes the external event vector *eev* for the processor based on the current device configurations.

Combined System. By combining the processor and device models we obtain a model for the overall system with devices as depicted in Fig. 1. This model allows interaction with environment via *eifi* and *eifo* whereas the communication between processor and devices is not visible from the outside anymore.

A configuration of the combined model consists of a processor configuration $c.c_{\mathrm{P}}$ and device configurations $c.c_{\mathrm{D}}$. We define a transition function Δ_{PD} and an output function Ω_{PD}. Both functions take the same three inputs: a processor-device index *idx*, a combined configuration $c.(c_{\mathrm{P}}, c_{\mathrm{D}})$, and an external input *eifi*.

We introduce some more notation for the transition and the output function. Let $difi = \Omega_{\mathrm{P}}(c.c_{\mathrm{P}})$ be the input from the processor to the devices. Let *eifi* be input from external environment. Let $(c'.c_{\mathrm{D}}, difo, eifo) = \Delta_{\mathrm{D}}(idx, c.c_{\mathrm{D}}, eifi, difi)$ denote the updated device configuration, the device output to the processor, and the external output. Let $eev = \Omega_{\mathrm{D}}(c'.c_{\mathrm{D}})$ denote the external event vector, which is computed based on the updated device configuration. Finally, if $idx = P$ then $c'.c_{\mathrm{P}}$ denotes the updated processor configuration, i.e. $c'.c_{\mathrm{P}} = \Delta_{\mathrm{P}}(c.c_{\mathrm{P}}, eev, difo)$. Otherwise $c'.c_{\mathrm{P}}$ denotes the unchanged processor configuration, i.e. $c'.c_{\mathrm{P}} = c.c_{\mathrm{P}}$.

The transition function Δ_{PD} returns the new configuration $c'.(c_{\mathrm{P}}, c_{\mathrm{D}})$. The output function Ω_{PD} simply returns the output to the external environment *eifo*.

A run of the combined system is defined for a given number of steps and a computational sequence $\sigma \in \mathbb{N} \rightarrow PD$. The latter designates the interleaving of the processor and device steps. A run starts with an initial configuration c^0. During a run the system receives inputs from the external environment as an input sequence $c.eifi \in \mathbb{N} \rightarrow eifi$ that maps step numbers to the corresponding inputs. A run for i steps results in a new configuration $c^{(i,\sigma)}.(c_{\mathrm{P}}, c_{\mathrm{D}})$ and external output sequence $c.eifo^{(i,\sigma)} \in \mathbb{N} \rightarrow eifo$. The new configuration is computed by recursive application of Δ_{PD} and the output sequence by application of Ω_{PD}.

3 Gate-Level Computer System

In this section we define a computer system at the gate level. This system has a similar structure to the system introduce above, i.e. it consists of the processor VAMP and a generic model for the implementation of external devices. The processor and the devices are connected via a common bus. All components of the system run in parallel and are clocked with the same hardware clock.

Processor. The VAMP processor [15] is a pipelined processor with out-of-order execution. Figure 2 gives an overview of the data path of the VAMP processor.

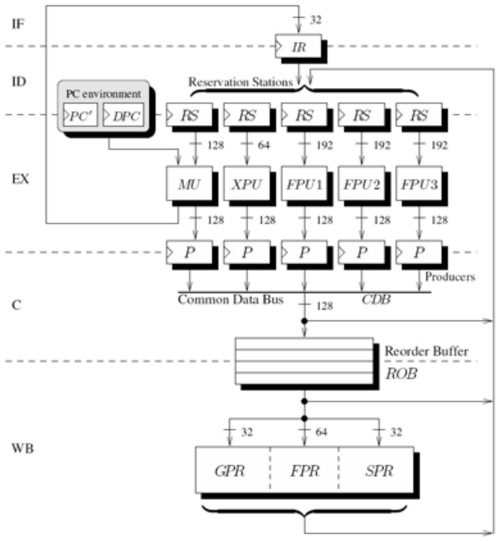


Fig. 2. Data path of the VAMP

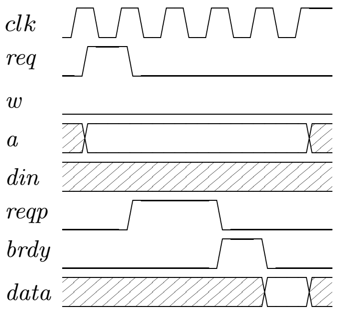


Fig. 3. Timing of a device read

The pipeline of the VAMP consists of five stages. In stage IF an instruction is fetched from the memory. In stage ID the instruction is decoded and put in a reservation station. The source operands are read from the processor's registers or forwarding paths. Both stages operate in order, realizing a pipelined implementation of the delayed-PC architecture [14] with one delay slot per control-flow instruction. The stages EX, C, and WB implement the Tomasulo algorithm [16] for out-of-order execution. In stage EX the instruction is executed, i.e. the result of the instruction is computed. In this stage memory and device accesses, if needed, are executed. The instruction result is then put on the common data bus in stage C. Finally, in stage WB the computed result is written back to the register file in-order. This is implemented via a reorder buffer (ROB). As soon as the oldest instruction result in the ROB becomes valid, it can be written back to the register file. The write back is the last step of the instruction execution and it signals that the instruction is leaving the processor. The ROB is also used to implement precise interrupts. If an instruction is interrupted, previous instructions are written back and later instructions are flushed. The interrupted instruction is written back or flushed depending on the interrupt type (repeat or continue). We distinguish two interrupt sources: internal (e.g. for page fault or overflow) and external (e.g. for reset or I/O interrupts). Internal interrupts are computed during instruction execution while external interrupts are an additional input bus of the VAMP.

For a detailed description of the processor core implementation see [17].

The VAMP Communication Interfaces. The VAMP communicates with two "off-the-chip" components: the external devices and the memory. The processor employs the following communication buses: (i) device interface input and output (*difi* and *difo*), (ii) memory interface input and output (*mifi* and *mifo*), and (iii) an external event vector *eev* with which devices signal their interrupts.

The VAMP accesses devices by setting the bit $difi.req$. The type of access (read or write) is given by bit $difi.w$. The accessed address is set on $difi.a \in \mathbb{B}^{30}$ and data for the write access are on $difi.din \in \mathbb{B}^{32}$. The accessed device sets $difo.reqp$ to signal that it is processing a request and cannot accept a new one. An active value of bit $difo.brdy$ signals that in the next cycle the request is finished and the data on $difo.dout \in \mathbb{B}^{32}$ will be valid. Figure 3 shows a typical processor-device communication. The memory protocol is similar.

Memory. Memory is a non-modeled external component. We define its content by observing memory interfaces. Let M^0 be an initial memory content. Let $mifi^t$ and $mifo^t$ be the state of the memory buses at cycle t. We introduce a function $bw(mifi^t, M^{t-1}(a)) \in \mathbb{B}^{64}$ which computes the new content of the cell on address a. Let predicate $busy^t$ signal the end of a memory access. Let the shorthand $wr^t \triangleq mifi^t.bwb \neq 0^8$ denote if a non zero number of bytes is written. The memory content M^t at cycle $t > 0$ is recursively defined as $M^t(a) = bw(mifi^t, M^{t-1}(a))$ if $mifi^t.a = a \wedge \neg busy^t \wedge wr^t$ and $M^t(a) = M^{t-1}(a)$ else.

Sampling External Interrupts. Previously, the VAMP has sampled external interrupts on the *eev* bus in the WB stage [15]. This is problematic, however, when relating the gate-level implementation to the instruction-set architecture. For example, consider an instruction that clears (acknowledges) a device interrupt. In the implementation, after the device access completes the device lowers its interrupt and the instruction leaves the memory unit. To reach the WB stage, the instruction needs several hardware cycles (cf. Fig. 2). In this time frame the device may reactivate its interrupt, which will then be sampled for the same instruction in the WB stage. From an assembly programmer's point of view, however, the new interrupt belongs to, or should affect, the next instruction.

In [11] an (informal) device-specific solution was proposed for this problem, still sampling all external interrupts in the WB stage. Here, we solve this problem generically and hence much more elegantly. Revisiting the instruction-set architecture, we may distinguish two types of device accesses: active ones that are performed when executing load / store instruction on device addresses and passive reads that occur when external interrupts are sampled. This interpretation reveals that *every* instruction (with external interrupts enabled) executes at least one device access and that active accesses are always accompanied by a passive read. To avoid any "shadow" scenarios devices should be accessed exactly once per instruction. Obviously, the instructions without any active device access already satisfy this requirement. For the other instructions we satisfy the requirement by sampling externals interrupts at the time when the device access completes. Thus, the access result and the external interrupts are read at the same cycle.

Notably, other implementations described in literature do not solve the above problem. For example, in [18] and in the MIPS-R3000 family [19, Chapter 8] external interrupt are sampled before instruction issue and before the memory stage, respectively. Both implementations still may access external devices twice.

Devices. All gate-level devices make a step in every cycle, which is the main difference to the ISA-level devices. Configurations of all devices are combined in a mapping h_{D} from device indices i to the corresponding device configuration. An input from external environment to all devices is represented as a mapping *eifis* from device indices to device inputs. Similarly, an output from all devices is a mapping *eifos* from device indices to device outputs.

The step function δ_{D} defines the behavior of devices. It takes input from the environment for all devices *eifis*, processor input *difi*, state of all devices h_{D}, and *reset* bit. It produces new state h'_{D}, outputs to the processor *difo*, output to the environment *eifos*, and external interrupts for the processor *eev*.

We make an assumption that δ_{D} obey the processor-device protocol (Fig. 3).

Combined System. The gate-level computer system consists of the VAMP and the generic device model. The VAMP and devices communicate via the internal device interface buses *difi*, *difo*, and *eev*. These buses introduce one cycle delay in the communication, e.g. the VAMP places the request on the *difi* and at the next cycle the device reads these data.

The next-state function updates the processor and the devices by the application of their next-state functions. It also defines the communication between the system components and the external world. For the combined VAMP-Devices system the external world consists of the memory chips and the external environment of the devices. Formally, the step function computes the next state $h'_{\mathrm{PD}}.(h_{\mathrm{P}}, h_{\mathrm{D}})$ based on a given system state $h_{\mathrm{PD}}.(h_{\mathrm{P}}, h_{\mathrm{D}})$, inputs from the device environment *eifis* and the memory *mifo*. It also computes the outputs to the external environment *eifos* of the devices and to the memory *mifi*.

A run of the combined model is defined recursively by the application of the next-state function for a given number of hardware cycles. We employ $h^t.\mathit{eifis}$ to denote the device inputs from the external environment at cycle t. We denote the input from the external memory at cycle t as $h^t.\mathit{mifo}$. A run for t cycles starting from a configuration $h_{\mathrm{PD}}^{\mathit{init}}$ results in a new configuration $h^t.(h_{\mathrm{P}}, h_{\mathrm{D}})$ outputs to the external environment $h^t.\mathit{eifos}$, and outputs to the external memory $h^t.\mathit{mifi}$.

4 Computer System Correctness Criterion

We define a scheduling function which relates gate-level runs with their ISA-level counterparts. Its definition is based on special hardware events indicating progress at the gate level, which has to be reflected at the ISA-level. We split events into two groups, processor-sided and external-environment-sided.

Processor-Sided Events. The result of any instruction without memory or device access is finally computed at the write back stage, i.e. at the cycle when the instruction is leaving the VAMP. We use the notation wb^t to denote the value of the hardware write-back signal at cycle t.

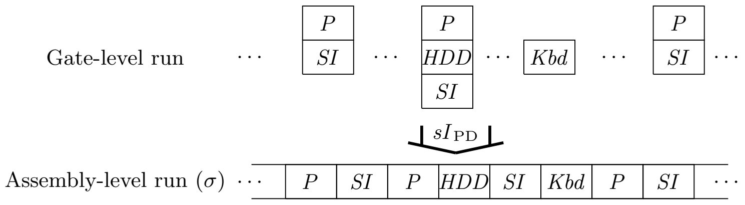


Fig. 4. Abstraction of hardware cycles to a computational sequence

An instruction writing to the memory or accessing a device forces *irreversible* changes of the memory / device when the access ends. The access cycle precedes the write back cycle of the instruction. Thus, external state is altered before the instruction leaves the processor. Therefore, memory and device correctness have to be treated specially. To built a precise relation, we need a cycle where the result of a device access is put on the bus (Fig. 3). We denote this event at cycle t by da^t. Note that da has almost the same semantics as wb, because once a device access is done its effect cannot be rolled back.

External-Environment-Sided Events. Device steps can be triggered by the external environment. We introduce a function *DevIds* which, for a given device state h_{D} and the external inputs *eifis*, computes a computational sequence $\sigma = \mathit{DevIds}(h_{\mathrm{D}}, \mathit{eifis})$ consisting of only device identifiers. Thus, sequence σ corresponds to the devices making step due to that input *eifis*. The definition of this function depends on the concrete device instances and how the gate-level device model has to be abstracted. The only restriction we put on *DevIds* is that it should not return a sequence with duplicate device identifiers. We present two examples for function *DevIds*: (i) modeling every implementation step in the specification, i.e. *DevIds* always returns a computational sequence which contains all device identifiers. (ii) purging those steps of a gate-level device which have no effect, e.g. if the state of the device did not change, its identifier is not in the result of *DevIds*.

We define the scheduling function sI_{PD}. For a given number of hardware cycles it returns a computational sequence σ. Figure 4 depicts how a gate-level run for a system with three devices can be mapped to a computational sequence. Function sI_{PD} is defined by recursion on hardware cycles. Let us abbreviate $\sigma^t = sI_{\mathrm{PD}}(t)$. At cycle zero we return the empty sequence $[\,]$, i.e. $\sigma^0 = [\,]$. For the recursion step let boolean flag was_da^t be true if there was the end of a device access but this instruction is not yet written back, i.e. $was\_da^t = \exists t' < t \,.\, da^{t'} \wedge \forall t'' \in]t' : t[\,.\, \neg wb^{t''}$. We define the recursive step of sI_{PD} as follows:

$$\sigma^{t+1} = \sigma^t \circ \begin{cases} \mathit{DevIds}(h^t.h_{\mathrm{D}}, h^t.\mathit{eifis}) \circ [P] & \text{if } da^t \\ [P] \circ \mathit{DevIds}(h^t.h_{\mathrm{D}}, h^t.\mathit{eifis}) & \text{if } wb^t \wedge \neg was\_da \wedge \neg da^t \\ \mathit{DevIds}(h^t.h_{\mathrm{D}}, h^t.\mathit{eifis}) & \text{otherwise} \end{cases}$$

Assumptions. We can only state and prove a relation between the gate-level and the ISA-level models if they receive the same inputs. We define a predicate *sync_eifis* that tests if inputs from the external environments between cycles ts and te are synchronised. If a device with identifier $\sigma^{te}(j)$ makes a step at $t \mapsto t{+}1$ (with $ts \leq t < te$), this device makes this step with the input $h^t.\mathit{eifis}(\mathit{idx})$. We compute this hardware cycle as $t = \mathit{AddedAt}(j)$. Then, we compare input for the device $h^t.\mathit{eifis}(\mathit{idx})$ with the specification input $c.\mathit{eifi}^j$ for step j. Similarly, we define a predicate *sync_eifos* that tests whether an output sequence to the external environment of implementation and specification match.

The gate-level and the ISA-level device models are based on the generic step functions δ_{D} and Δ_{D}, respectively. We keep them generic to instantiate them with needed devices. To state anything about runs of these models, we have to specify how results of these functions are related with each other.

We introduce a predicate sim_D that holds if the implementation states of all devices are in a relation with their specification. We make a "one-step" assumption specifying a relationship between one step of the-gate level device model and a sequence of steps in the specification of devices with respect to sI_{PD}. We define this assumption for *any* initial state and *any* input sequence.

$$\begin{aligned}&\mathit{sim}_D(h^0.h_{\mathrm{D}}, c^{\sigma^0}.c_{\mathrm{D}}) \wedge \mathit{sync\_eifis}(0, 1, h.\mathit{eifis}, c.\mathit{eifi}) \wedge h.\mathit{difi} = c^{\sigma^1}.\mathit{difi} \implies \\ &\quad \mathit{sim}_D(h^1.h_{\mathrm{D}}, c^{\sigma^1}.c_{\mathrm{D}}) \wedge \mathit{sync\_eifos}(0, 1, h.\mathit{eifos}, c^{\sigma^1}.\mathit{eifo}) \wedge \\ &\quad h^1.\mathit{difo}.\mathit{dout} = c^{\sigma^1}.\mathit{difo} \wedge h^1.\mathit{eev} = c^{\sigma^1}.\mathit{eev}\end{aligned}$$

Software Conditions. Sometimes it is impossible or too expensive to implement handling of some exceptions in the hardware. These special cases restrict the software which can be executed on the developed hardware. We call these restrictions *software conditions* [5]. We present two software conditions for the VAMP processor, which also hold for the gate-level computer system.

The first condition excludes RAW hazards for the self-modifying assembly code [15,20]. This condition makes use of so-called *sync*-instructions, which drain the processor pipeline. Other than that a sync-instruction should act as a no-op. In the VAMP the instruction *movs2i IEEEf R0* has sync semantics. Hillebrand [21] observed that jumping to and returning from an interrupt service routine has sync semantics, too. The software condition requires at least one sync instruction between any two instructions which produce a RAW hazard for instruction fetch.

In the presence of external devices, we need another software condition which guarantees the absence of accesses to the undefined address space. The main issue is the liveness because neither memory nor devices respond to an access to the undefined address space, and hence, such an access never terminates.

The Simulation Theorem. Our correctness criterion states that every implementation run can be simulated by a specification run. The time notions, hardware cycles and computational sequences, are related via a function sI_{PD}. Device states and system outputs are related via sim_D and *sync_eifos*, respectively. We introduce a predicate $\mathit{Rconf}(h_{\mathrm{P}}, c_{\mathrm{P}})$ which tests processor states.

The VAMP has more registers than the ISA processor. The registers, which are present in both models, are called the *visible registers* (from a programmer's point of view). These include general purpose registers and program counters. Other implementation registers are called *invisible*. They hold partial results of instruction execution, e.g. the internal registers of the functional units. The predicate $Rconf(h_{\mathrm{P}}, c_{\mathrm{P}})$ tests the visible registers of the VAMP and the ISA.

Some VAMP components, e.g. the program counters, may have values that never occur in the specification. Therefore, *the* correctness criterion of the VAMP is stated after every interrupt [15], which is signaled by signal *JISR*. At these cycles all visible registers of VAMP and ISA must have equal values.

We can now formulate the simulation theorem. Let inputs for the processor-device models be synchronised with respect to *sync_eifis* and let the initial states be equivalent. Let the assembly code satisfy the software conditions. Let $\sigma^t = sI_{\mathrm{PD}}(t)$. We show that the gate-level model after t cycles and ISA-level model after executing σ^t have equivalent states and produce equal outputs.

$$\begin{aligned}
&Rconf(h^0.h_{\mathrm{P}}, c^{[\,]}.c.c_{\mathrm{P}}) \wedge M^0 = c^{[\,]}.c.c_{\mathrm{P}}.M \wedge \\
&sim_D(h^0.h_{\mathrm{D}}, c^{\sigma^0}.c.c_{\mathrm{D}}) \wedge sync\_eifis(0, t, h.eifis, c.eifi) \implies \\
&\quad Rconf(h^t.h_{\mathrm{P}}, c^{\sigma^t}.c.c_{\mathrm{P}}) \wedge M^t = c^{\sigma^t}.c.c_{\mathrm{P}}.M \wedge \\
&\quad sim_D(h^t.h_{\mathrm{D}}, c^{\sigma^t}.c.c_{\mathrm{D}}) \wedge sync\_eifos(0, t, h.eifos, c^{\sigma^t}.eifo)
\end{aligned}$$

The theorem proof can be found in [22]. Note that in the proof of this theorem the shadow scenarios for sampling interrupts (Sect. 3) show up.

5 Summary

We have presented a formally verified computer system at the gate-level, consisting of a processor and external devices. To the best of our knowledge, this is the first formal treatment of device integration at the gate and ISA level.

The base of the computer system is a pipelined processor, which is called VAMP. It is a 32-bit RISC processor featuring out-of-order execution with five functional units, precise interrupts, and address translation. In contrast to previous work, we verified the VAMP in the context of pervasive system verification, considering also external devices. Moreover, the formal verification of the combined system, allowed us to establish a clean semantics of the external interrupts. External devices can be easily integrated in the computer system. For example, the integration of a controller for a time-triggered bus was shown in [22]; the resulting system can be used an electronic control unit (ECU) in a distributed automotive system. These results are formalized and mechanically proved in the interactive theorem prover Isabelle/HOL. We also synthesised and ran the verified ECU on an FPGA and the unit size is ca. 5M gate equivalents.[1]

Proof Comparison and Statistics. We compare our results with the previous work on the VAMP processor. The original VAMP project is carried out in the

[1] This is joint work with Andrey Shadrin.

Table 1. Verification efforts in PVS and Isabelle/HOL with IHaVeIt

Prover	Verification target	Person years	Theorems	Proof steps
PVS	VAMP (no FPU, MU)	3	966	37666
Isabelle	VAMP (no FPU, MU)	1.5	1206	20455
	Devices	0.5	52	967
	Combining systems	0.7	118	2714
	Total	2.7	1376	24316

interactive theorem prover PVS. In Table 1 we show verification efforts in PVS and in Isabelle/HOL in terms of person years, number of theorems, and number of proof steps. This comparison cannot be 100% "fair", because the nature and power of these systems are too different, our work in Isabelle/HOL is based on the proof strategy developed in PVS. The latter reason explains why we could finish the proofs in less person years. The difference in proof steps is due to the fact that PVS proofs are done purely interactively while proofs in Isabelle/HOL are partially automated. In terms of proofs steps, ca. 40% of user work can be saved thanks to IHaVeIt [7]. For the same reason there are more theorems in Isabelle/HOL since all automatically proven results (e.g. subgoals of a theorem) are formulated as separate lemmas.

Future Work. We see several interesting topics for future work. Alekhin [23] verified a memory unit with a translation look-aside buffer to speed up address translation. We plan to formally integrate his results into our system. There are formal liveness proofs for the Tomasulo scheduler and the VAMP functional units, but no combined liveness proof yet for the entire processor. Finally, the device model could be extended with direct memory accesses.

References

1. Nuseibeh, B.: Ariane 5: Who dunnit? IEEE Softw. 14, 15–16 (1997)
2. The Verisoft Consortium: The Verisoft Project (2003), http://www.verisoft.de/
3. Moore, J.S.: A grand challenge proposal for formal methods: A verified stack. In: Aichernig, B.K., Maibaum, T.S.E. (eds.) Formal Methods at the Crossroads. From Panacea to Foundational Support. LNCS, vol. 2757, pp. 161–172. Springer, Heidelberg (2003)
4. Alkassar, E., Hillebrand, M.A., Leinenbach, D.C., Schirmer, N.W., Starostin, A., Tsyban, A.: Balancing the load: Leveraging semantics stack for systems verification. JAR: Special Issue on Operating Systems Verification (to appear, 2009)
5. Dalinger, I., Hillebrand, M., Paul, W.: On the verification of memory management mechanisms. In: Borrione, D., Paul, W. (eds.) CHARME 2005. LNCS, vol. 3725, pp. 301–316. Springer, Heidelberg (2005)
6. Beyer, S., Jacobi, C., Kroening, D., Leinenbach, D., Paul, W.: Putting it all together: Formal verification of the VAMP. International Journal on Software Tools for Technology Transfer 8, 411–430 (2006)

7. Tverdyshev, S., Alkassar, E.: Efficient bit-level model reductions for automated hardware verification. In: TIME 2008, pp. 164–172. IEEE, Los Alamitos (2008)
8. Bevier, W.R., Hunt, W.A., Moore, S., Young, W.D.: An approach to systems verification. Journal of Automated Reasoning 5, 411–428 (1989)
9. Berry, G., Kishinevsky, M., Singh, S.: System level design and verification using a synchronous language. In: ICCAD, pp. 433–440 (2003)
10. ALDEC – The Design Verification Company: UART nVS (2006), http://www.aldec.com/products/ipcores/_datasheets/nSys/UART_nVS.pdf
11. Hillebrand, M., In der Rieden, T., Paul, W.: Dealing with I/O devices in the context of pervasive system verification. In: ICCD 2005, pp. 309–316. IEEE, Los Alamitos (2005)
12. Alkassar, E., Hillebrand, M., Knapp, S., Rusev, R., Tverdyshev, S.: Formal device and programming model for a serial interface. In: Beckert, B. (ed.) VERIFY 2007. CEUR Workshop Proceedings, vol. 259, pp. 4–20. CEUR-WS.org (2007)
13. Alkassar, E., Hillebrand, M.A.: Formal functional verification of device drivers. In: Shankar, N., Woodcock, J. (eds.) VSTTE 2008. LNCS, vol. 5295, pp. 225–239. Springer, Heidelberg (2008)
14. Mueller, S.M., Paul, W.J.: Computer Architecture: Complexity and Correctness. Springer, Heidelberg (2000)
15. Beyer, S.: Putting It All Together: Formal Verification of the VAMP. PhD thesis, Saarland University, Saarbrücken (2005)
16. Tomasulo, R.M.: An efficient algorithm for exploiting multiple arithmetic units. IBM Journal of Research and Development 11, 25–33 (1967)
17. Kröning, D.: Formal Verification of Pipelined Microprocessors. PhD thesis, Saarland University, Saarbrücken (2001)
18. Smith, J.E., Pleszkun, A.R.: Implementing precise interrupts in pipelined processors. IEEE Trans. Comput. 37, 562–573 (1988)
19. Brüss, R.J.: RISC Die MIPS-R3000-Familie. Architektur, Systembausteine, Compiler, Tools, Anwendungen. Siemens (1991)
20. Dalinger, I.: Formal Verification of a Processor with Memory Management Units. PhD thesis, Saarland University, Saarbrücken (2006)
21. Hillebrand, M.: Address Spaces and Virtual Memory: Specification, Implementation, and Correctness. PhD thesis, Saarland University, Saarbrücken (2005)
22. Tverdyshev, S.: Formal Verification of Gate-Level Computer Systems. PhD thesis, Saarland University, Saarbrücken (2009)
23. Alekhin, A.: The VAMP memory unit: Hardware design and formal verification effort. Master's thesis, Saarland University, Saarbrücken, Germany (2008)

On Models of a Nondeterministic Computation

Mikhail N. Vyalyi*

Dorodnitsyn Computing Center of RAS,
Vavilova, 40, Moscow 119991, Russia
vyalyi@gmail.com

Abstract. In this paper we consider a nondeterministic computation performed by deterministic multi-head 2-way automata with a read-only access to an auxiliary memory. The memory contains additional data (a guess) and the computation is successful iff it is successful for some memory content.

Also we consider the case of restricted guesses in which a guess should satisfy some constraint.

We show that the standard complexity classes such as L, NL, P, NP, PSPACE can be characterized in terms of these models of nondeterministic computation. These characterizations differ from the well-known ones by absence of alternation.

Keywords: automaton, nondeterminism, language, complexity class.

The standard way to define a nondeterministic computation by an automaton or a Turing machine is to change a transition function by a transition relation. In a nondeterministic state of a computational device a computation branches into several computation paths.

There is another way to introduce a nondeterminism. Suppose that a computational device has an additional data (a *guess* or a *certificate* or a *proof of correctness*) and performs a deterministic computation operating with an input data and a guess data.

Sometimes these variants of introducing nondeterminism lead to equivalent computational models. The class NP, for example, can be defined in both ways using Turing machines.

If we restrict computational power then these variants may differ drastically. The aim of this paper is to investigate models of nondeterminism based on the second variant for multi-head 2-way automata.

It is well-known[1] that computation abilities of deterministic multi-head 2-way automata are equivalent to Turing machines with a logarithmically bounded auxiliary memory. In other words, they recognize languages from the class L.

Nondeterministic (in the sense of transition relation) multi-head 2-way automata recognize languages from the class NL. One can rewrite a definition of a

* The work is supported by the RFBR grants 08–01–00414, 09-01-00709 and the grant NS 5294.2008.1.

[1] O. H. Ibarra [12] attributed this result to A. Cobham and coauthors referring to an unpublished manuscript.

nondeterministic automaton in terms of guess data. For this purpose the 1-way read-only guess tape should be used.

Here we introduce a more general model of an auxiliary read-only memory (see definitions in Section 1). Guess data are stored in cells of a memory and at each moment of time an automaton has an access to the exactly one memory cell. Possible moves between memory cells form a directed graph (*a memory graph*). An automaton can choose between finite number of variants only. So the natural condition on the memory graph is a finite fan-out in each vertex (i.e. a memory cell).

The most natural variant of an auxiliary memory is a 2-way tape. This model appears to be very close to nonerasing nondeterministic stack automata (NENSA) [12,10]. The automata with the 2-way read-only guess tape recognize the same class of languages PSPACE as NENSA do.

A read-only memory may be useful in nondeterministic settings. Nevertheless, automata with a read-only nondeterministic memory can be related with a special variant of deterministic computation with an auxiliary memory, which is called WORM-automata (see Subsection 1.1). The WORM-automata with the 2-way guess tape are similar to the nonerasing deterministic stack automata (NEDSA) and also recognize the languages from the class PSPACE.

Also we introduce a nondeterministic computation with a restricted guess. An example of restricted guess is a *sparse guess*. Sparseness of a guess means that a guess tape contains the only one (or finitely many) non-empty symbol and the rest symbols stored on the tape are empty.

We focus our attention on a more restricted memory model, so-called 1.5-way tape. It was used in research of quantum automata [1]. For classic automata 1.5-way tape means an 1-way tape with a reset option, i.e. a possibility to make move into the initial cell from any memory cell.

The main results of this paper concern the 1.5-way tape memory.

The automata with the 1.5-way guess tape recognize the class PSPACE (Theorem 2 below) as the 2-way guess tape automata do. But the WORM-automata with this memory type recognize the class P only (Theorem 1). 1.5-way automata with sparse guesses recognize the class NP(Theorem 3). These results show that the 1.5-way guess tape is potentially more suitable to characterize various complexity classes.

An interesting feature of all these results is a formal absence of resource bounds in characterizations of resource-bounded classes such as P, NP and so on. It should be noted that there is a primary result of this sort: many heads are equivalent to logarithmic space. The rest of the results are based on this fact.

The main technical tool in study of the 1.5-way tape is calculations modulo polynomially bounded integer. These calculations can be performed on a logarithmic space. To compute a length of a part of the guess tape we use the simple algorithm: go along the part and increase a counter modulo p. The latter operation can be done on logarithmic space.

There are many results on characterizations of complexity classes in terms of some sort of automata. The classes L, NL, P, PSPACE have the well-known characterizations by deterministic, nondeterministic, alternating and synchronized alternating 2-way automata [5,11,7]. There are also characterizations of NP, the polynomial hierarchy and some other complexity classes in terms of alternating auxiliary stack automata [9].

Our results differ from these characterization because the models considered in this paper do not use alternation.

Some results in theory of tree-walking automata can be translated to our framework. In this case a memory model is a tree. For example, it follows from [3] that the automata with a read-only access to a tree memory recognize the languages from the class EXP.

It is worth to mention a paper [4], which contains the characterizations of P, NP and PSPACE in terms of nondeterminism only. The difference is in the nature of nondeterminism introduced. In [4] nondeterministic colorings of n-dimensional words are considered, where n is the input size. Contrary, our main results concern the case of 1-dimensional guess memory, which is potentially infinite. Using a potentially infinite tape makes more difficult an interpretation of the results in terms of descriptive complexity theory (see, e.g., the book [8]). For example, the results in [4] are directly related with Fagin's theorem that characterizes the class NP in terms of the second-order logic. To establish a similar relation to our characterization of the class NP one need specify suitably restricted infinite models. Up to the moment we know no way to implement this idea.

The rest of paper is organized as follows. In Section 1 we introduce our basic computational model: multi-head 2-way automata with a nondeterministic auxiliary memory. Section 2 contains results about the 1-way, the 1.5-way and the 2-way guess tapes. In Section 3 we introduce a model of a restricted guess and give characterizations of NP in terms of this model. In Section 4 we make some remarks on a more general memory model, which is called a monoid memory.

Details of proofs are omitted here due to space limitations. They can be found in the preprint [15].

1 Automata with an Auxiliary Read-Only Memory

In this section we provide definitions for a model of nondeterministic computation by automata with an auxiliary read-only memory. The definitions fix an informal idea explained in the introductory section.

Definition 1. A *memory model* is a directed graph (M, E), an initial cell $m_0 \in M$ and a marking map $g\colon E \to G$ from the edges of the graph to some finite set G. The marking map satisfies the following conditions:

- $g(u, v) \neq g(u, w)$ for $v \neq w$ (edges outgoing from the vertex can be distinguished by their marks);
- for each $u \in M$ and $a \in G$ there is an edge $(u, v) \in E$ such that $g(u, v) = a$.

In other words, the map g restricted to the set of edges outgoing from a vertex is a bijection.

For any finite alphabet Δ *a memory content* μ is a map $\mu\colon M \to \Delta$.

Definition 2. An *h-head automaton A with an auxiliary memory of model M* (an M-automaton for brevity) is characterized by

- a finite state set Q,
- a finite input alphabet $I = \Sigma \cup \{\triangleleft, \triangleright\}$, $\Sigma \cap \{\triangleleft, \triangleright\} = \varnothing$,
- a finite memory alphabet Δ,
- a transition function $\delta \colon Q \times I^h \times \Delta \to Q \times \{-1, 0, 1\}^h \times (G \cup \mathsf{0})$, which maps an $(h+2)$-tuple (the current state, symbols the heads on the input tape, the symbol in the current memory cell) to an $(h+2)$-tuple (a new state, a motion command for each head, a command of changing memory cell),
- an initial state $q_0 \in Q$,
- a set of accepting states $Q_a \subset Q$.

A configuration of the automaton A is an $(h+2)$-tuple $(q, i_1, \dots, i_h, m)$ (the state, the positions of the heads, the memory cell). *A surface configuration* of the automaton A is an $(h+1)$-tuple $(q, i_1, \dots, i_h)$.

The transition function defines a transformation on the set of the configurations. A *motion command* for a head is an element from the set $\{-1, 0, +1\}$ indicating the shift of the head along the input tape. A *command of changing memory cell* is an element of the marking set G or an empty command $\mathsf{0}$. In the case of a non-empty command $g \in G$ the automaton moves out the current memory cell along the edge marked by g. The empty command do not change the memory cell.

An automaton A operates on an input word $w \in \Sigma^*$ in natural way. We assume that the input word is extended by the endmarkers $\{\triangleleft, \triangleright\}$ indicating the beginning and the end of the word. The automaton starts from the initial state q_0, the initial position of each head is the leftmost symbol of the input word, the initial memory cell is m_0. On each step of operation the automaton changes the configuration as described above. The automaton stops iff either it reaches an accepting state or a head goes out the area bounded by the endmarkers.

Definition 3. An automaton A *accepts* an input word w iff for some memory content μ it stops in an accepting state.

The automaton *recognizes* the language L iff for any $w \in L$ it accepts w and for any $w \notin L$ it do not accept w.

We denote by M-NFA the class of languages recognized by automata with an auxiliary memory of model M.

1.1 Determinization

A nontrivial use of an auxiliary read-only memory is inevitably nondeterministic. Changing the read-only mode by the read-write mode in many cases leads to a broader language class. This contradicts an intuition that a deterministic model is weaker than a nondeterministic one.

In this subsection we describe a variant of deterministic use of an auxiliary memory of model M which gives a subclass of M-NFA. It is the WORM (write once, read many) mode. A WORM-M automaton should fill a new memory cell by a symbol when it enter the cell the first time. In further operation the automaton can not change the content of the cell.

Below we give a formal definition compatible with the above nondeterministic model.

Definition 4. A *WORM-memory automaton on memory model M* (a WORM-M automaton for brevity) is characterized by

- a finite state set Q,
- a finite input alphabet $I = \Sigma \cup \{\triangleleft, \triangleright\}$, $\Sigma \cap \{\triangleleft, \triangleright\} = \varnothing$,
- a finite memory alphabet $\Delta \cup \{\mathsf{void}\}$,
- a transition function δ, which maps a $(h+2)$-tuple (the current state, symbols of the input word under the heads, the symbol in the current memory cell) to a $(h+2)$-tuple (a new state, a motion command for each head, a command of changing memory cell),
- the initial state $q_0 \in Q$,
- the set of accepting states $Q_a \subset Q$,
- the set of writing states $Q_w \subset Q$.
- a filling memory function $\varphi\colon Q_f \to \Delta$,

At the start of operation all memory cells are void. A WORM-memory automaton operates in the same way as a nondeterministic M-automaton except the moments of entering a writing state. In that moment the filling function is applied to the current state of the automaton. If the current memory cell is visited at first time then the value of the filling function is assigned to the cell and the automaton continues operation by application of the transition function. An attempt to change the content of a cell visited before causes the error as well as an attempt to apply the transition function at a void cell. In the case of an error the automaton stops the operation and do not accept the input word.

So, during a successful operation the automaton enters a new memory cell in a writing state. Also note that if the automaton writes the non-void symbol d to the cell containing the symbol d then no error occurs. We call this property 'a freedom of writing the same'.

We denote by M-WORM the class of languages recognized by deterministic automata with an auxiliary WORM-memory of model M.

Lemma 1. *M-WORM $\subseteq$ M-NFA.*

The idea of the proof is simple: a nondeterministic M-automaton A simulating a WORM-M memory automaton B expects a memory content consistent with the operation of B. Details can be found in [15].

Due to Lemma 1 one can regard the WORM-M automata as a specific case of M-automata.

2 Complexity Classes Recognized by Automata with an Auxiliary Tape Memory

2.1 1-Way Tape

Let W_1 be an infinite 1-way tape (Fig. 1). The class W_1-NFA is just the class NL. Indeed, a W_1-automaton can read a symbol from the guess tape once. This symbol can be used to make a nondeterministic choice in a transition relation.

Note also that W_1-WORM = L because a symbol from the W_1-tape can not be reread.

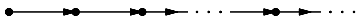


Fig. 1. 1-way tape W_1

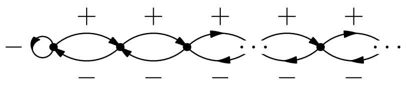


Fig. 2. 2-way tape W_2

2.2 2-Way Tape

Let W_2 be an infinite 2-way tape (Fig. 2). For graphs of fan-out > 1 we should also indicate the marking of edges. In the case of W_2 the marking is natural: mark '+' is placed on the edges going from a vertex n to the vertex $n + 1$, mark '−' is placed on the edges going to the opposite direction.

It was mentioned above that W_2-NFA = PSPACE because W_2-automata is almost the same as nonerasing nondeterministic stack automata and NENSA recognize the class PSPACE [12].

NENSA is able to make arbitrary nondeterministic transitions while an W_2-automaton should follow data read from the guess tape. It means that W_2-automata are weaker than NENSA, so W_2-NFA $\subseteq$ PSPACE. The reverse inclusion is valid even for WORM-W_2 automata. Indeed, a WORM-W_2 automaton is able to write a computational history of a Turing machine computation on a polynomially bounded space. For this purpose the automaton should move on distances polynomially bounded by the input size. This can be done by implementing polynomially bounded counters.

Thus, W_2-NFA $\subseteq$ PSPACE $\subseteq$ W_2-WORM $\subseteq$ W_2-NFA (the last inclusion is due to Lemma 1).

2.3 1.5-Way Tape

The memory model $W_{1.5}$ is pictured on the Fig. 3. Edges going to the right are marked by '+' and edges going to the initial vertex are marked by '−'.

Theorem 1. *$W_{1.5}$-WORM = P.*

The inclusion P $\subseteq$ $W_{1.5}$-WORM follows from the fact that a WORM-$W_{1.5}$ automaton A is able to simulate a WORM-W_2 automaton B on a polynomially

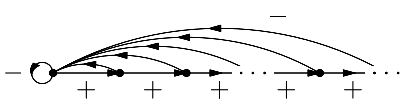


Fig. 3. 1.5-way tape $W_{1.5}$

bounded space by implementing a polynomially bounded counter that keeps a position of B on the W_2-tape.

The reverse inclusion follows from two simple observations. The first observations holds for general $W_{1.5}$-automata.

Lemma 2. *Let A be a $W_{1.5}$-automaton and $\#Q$ be the number of its states. Then any accepting computation of A includes no more than $\#Q$ moves to the initial cell.*

Proof. After each return move the automaton A scans the same tape content and its behavior is deterministic. So, if A starts the scan process from the same state twice it loops and never reach an accepting state.

Thus, the number of return moves is no more than the number of the states. □

The second observation is specific to the WORM-$W_{1.5}$ automata.

Proposition 1. *Let A be a WORM-$W_{1.5}$ automaton, h be the number of heads, n be the length of the input word w and $\#Q$ is the number of the states of A. If A accepts w then between two subsequent return moves the automaton visits no more than $n^h\#Q$ new cells.*

Proof. There are no more than $n^h\#Q$ surface configurations of A. If the automaton pass through more than $n^h\#Q$ new cells, some surface configuration occurs twice. It means that the automaton loops and moves to the right infinitely. □

These facts immediately imply that an operation of a WORM-$W_{1.5}$ automaton can be simulated by a Turing machine in polynomial time.

Thus the WORM-$W_{1.5}$-automata are weaker than the WORM-W_2 automata. As for nondeterministic automata, 1.5-way tape provides the same computational power as 2-way tape.

Theorem 2. $W_{1.5}$-NFA = PSPACE.

In one direction the inclusion is obvious:

$$W_{1.5}\text{-NFA} \subseteq W_2\text{-NFA} = \text{PSPACE}\,. \tag{1}$$

To prove the inclusion PSPACE $\subseteq W_{1.5}$-NFA we show that a $W_{1.5}$-automaton is able to check correctness of a computational history for a Turing machine computation on a polynomially bounded space. Configurations of the Turing machine are represented in special form using arithmetic encoding of binary words [13,14]. Namely, a word $w \in \{0,1\}^*$ is encoded by a positive integer $c(w)$ written in binary as $1w$.

A TM configuration $\ell q a r$ will be encoded by a 4-tuple $(c(\ell), q, a, c(r^R))$, where r^R denote the reversal of the word r.

It is easy to verify that the integers $c(\ell)$ and $c(r^R)$ before and after each step of the TM computation are related by equations

$$y = 2x,\ y = 2x + 1,\ x = 2y + 1,\ x = 2y\,, \tag{2}$$

where x is the old value and y is the new value of $c(\ell)$ or $c(r^R)$. The exact choice of a relation depends on the pair q, a and parities of $c(\ell)$, $c(r^R)$.

For a computation on a polynomial space it is sufficient to check relations (2) modulo polynomially bounded integers. A particular modular check can be done by a W_1-automaton that scans the description of the computational history from the left to the right. So a $W_{1.5}$-automaton can perform all modular checks jumping back to the initial cell before starting the next modular check.

The correctness of this procedure follows from the Chinese remainder theorem and the prime number theorem [2].

Details of the proof outlined above can be found in [15].

3 The Restricted Guess Case

In this section we introduce a generalization of the nondeterminism model. Namely, we will put a restriction on the form of a guess. The restriction can change a computational power of the model.

Definition 5. Let $\mathcal{T} \subseteq \Delta^M$ be a subset of memory contents. We say that a M-automaton A *accepts a word w with a $\mathcal{T}$-restricted guess* iff it accepts w operating on some memory content μ from the set $\mathcal{T}$.

We denote by $M(\mathcal{T})$-NFA the corresponding class of languages recognizable by M-automata with a $\mathcal{T}$-restricted guess.

Of course, in general $M(\mathcal{T})$-NFA $\not\subseteq$ M-NFA. For example, let $\mathcal{T}$ is the set of all valid computational histories of a Turing machines. Then $W_2(\mathcal{T})$-NFA contains all recursively enumerable languages.

We are interested in restrictions that describe subclasses of M-NFA. To guarantee the inclusion $M(\mathcal{T})$-NFA $\subseteq$ M-NFA it is sufficient to construct an automaton V that checks compatibility of memory content η in visited cells with the set $\mathcal{T}$. Compatibility means that η can be extended to some $\tau \in \mathcal{T}$. As an example of this kind of restriction we introduce *sparse guesses*.

Definition 6. Let $\Delta = \{0\} \cup \Delta'$. A *$k$-sparse guess* contains no more than k symbols from the Δ'.

We denote by U_k the set of k-sparse guesses.

Below we consider sparse guesses for tape memories.

3.1 Sparse Guesses for 1.5-Way Tape

The following facts can be verified easily by use of an informal idea of guess verification described above.

Lemma 3. *$W_{1.5}(U_k)$-NFA $\subseteq W_{1.5}$-NFA for any k.*

Lemma 4. *$W_{1.5}(U_1)$-NFA $\subseteq W_{1.5}(U_k)$-NFA.*

Proofs can be found in [15].

Now we give a characterization of the classes $W_{1.5}(U_k)$-NFA.

Theorem 3. *$W_{1.5}(U_k)$-NFA $=$ NP for $k \geq 1$.*

The proof of Theorem 3 is divided into two parts.

Lemma 5. *NP $\subseteq W_{1.5}(U_1)$-NFA for $k \geq 1$.*

An U_1-guess can mark a cell on the $W_{1.5}$-tape by a nonzero symbol. The distance d between the initial and the marked cell can be used to encode an information. The Chinese remainder theorem guarantee that one can encode polynomially many bits by residues modulo polynomially bounded primes. A $W_{1.5}$-automaton A recognizing an NP-language L expects that these bits form a computational history of a nondeterministic computation by a nondeterministic Turing machine recognizing the language L. To verify a guess the automaton A needs to extract a bit indexed by a prime p from the encoded data. This can be done computing the distance d modulo p.

Detailed exposition of the proof is contained in [15].

Lemma 6. *$W_{1.5}(U_k)$-NFA $\subseteq$ NP for any k.*

Due to Lemma 2 to find out the result of a $W_{1.5}$-automaton A operation on a U_k-guess one can divide the operation into polynomially many phases such that during each phase the $W_{1.5}$-automaton moves to the right and behaves like a W_1-automaton.

Note that the size of the set S of surface configurations of the $W_{1.5}$-automaton A is polynomially bounded by the input size. Changing a surface configuration after the reading of the symbol 0 is described by a map $\alpha_0 \colon S \to S$. Iterations of the map α_0 stabilize on some cycle: $\alpha_0^{i+\ell} = \alpha_0^i$ for sufficiently large i. an The length ℓ of the cycle is polynomially bounded. From these observations one can easily conclude that if the $W_{1.5}$-automaton A accepts on some U_k-guess then it accepts on a U_k-guess of exponential length, where the length of guess is the maximum of distances between the initial cell and cells marked by nonzero symbols.

So the positions of cells marked by nonzero symbols can be specified nondeterministically by NTM running in polynomial time. Given the positions one can compute the result of operation of the $W_{1.5}$-automaton A in deterministic polynomial time. For this purpose an algorithm of fast computation of matrix exponentiation can be applied.

Details of the proof can be found in [15].

Now Theorem 3 follows from Lemmata 4, 5, 6:

$$\mathrm{NP} \subseteq W_{1.5}(U_1)\text{-NFA} \subseteq W_{1.5}(U_k)\text{-NFA} \subseteq \mathrm{NP}\,. \tag{3}$$

3.2 Sparse Guesses for 2-Way Tape

U_k-guesses can be verified on the 2-way tape also.

Lemma 7. $W_2(U_k)$-NFA $\subseteq W_2$-NFA *for any* k*.*

This analog of Lemma 4 is proved in a straightforward way.

It is appeared that the class $W_2(U_1)$-NFA is rather weak.

Theorem 4. Aux2DC $\subseteq W_2(U_1)$-NFA $\subseteq$ Aux2NC $\subset$ P.

Here Aux2NC (Aux2DC) is the class of languages recognized by nondeterministic (deterministic) 2-way counter automata with a logarithmic auxiliary memory.

This effect is due to the absence of the root label in the initial cell. Using a non-zero symbol as the root label a W_2-automaton can simulate a deterministic counter automaton. Thus we obtain the first inclusion in Theorem 4.

To prove the second inclusion note that a position of the unique nonzero symbol on the tape can be found by a nondeterministic counter automaton (NCA) nondeterministically. After that the automaton can simulate the operation of a W_2-automaton A on a U_1-guess using the value of the counter to indicate the position of the W_2-automaton on the 2-way tape. This works well while the W_2-automaton is to the right of the cell marked by the nonzero symbol.

It is appeared that a behavior of the W_2-automaton while it moves between the initial and the marked cell can be simulated by the NCA nondeterministically using a logarithmic space. Details of the simulation can be found in [15].

The last inclusion in Theorem 4 follows from the Cook theorem [6]. The Cook theorem implies

$$\mathrm{Aux2PDA} = \mathrm{AuxN2PDA} = \mathrm{P}\,, \tag{4}$$

where Aux2PDA is the class of languages recognized by deterministic 2-way pushdown automata with a logarithmic auxiliary memory and AuxN2PDA is the class of languages recognized by nondeterministic 2-way pushdown automata with a logarithmic auxiliary memory.

For $k \geq 2$ the classes $W_2(U_k)$-NFA coincide with NP.

Theorem 5. $W_2(U_k)$-NFA $=$ NP *for* $k \geq 2$*.*

Note that Theorem 3 implies that $W_2(U_2)$-NFA $\supseteq$ NP because one nonzero symbol can be used to mark the initial cell and the other can be used for a simulation of a U_1-guess for a $W_{1.5}$-automaton.

The reverse inclusion can be proved in a way similar to the the proof of Lemma 6.

4 Monoid Memory

In this final section we briefly outline a natural extension of tape memories. The results mentioned below can be easily obtained by translation some well-known folklore facts to our settings.

Let G be a monoid generated by a set $G' = \{g_1, \dots, g_n\}$. Then the memory of type (G, G') is defined by the Cayley graph of the monoid M: the vertex set is G, an edge marked g_k goes from a vertex x to the vertex xg_k.

1-way and 2-way tapes are examples of monoid memory. It follows immediately from definitions that W_1-NFA $= (\mathbb{N}, \{+1\})$-NFA. Also it is easy to see that W_2-NFA $= (\mathbb{Z}, \{+1, -1)\})$-NFA.

There is a weak upper bound for the classes M-NFA of a monoid memory M.

Theorem 6. *Let M be a monoid. If the word problem for M is decidable then M-NFA $\subseteq \Sigma_1$, where Σ_1 is the class of recursively enumerable languages.*

The proof follows from the observation that a decision procedure for the word problem for the monoid M can be used to enumerate accepting computation histories of a M-automaton.

For many monoids and groups the bound of Theorem 6 is exact.

Take, for example, a $\mathbb{Z}^2$ memory. The generators of $\mathbb{Z}^2$ are chosen naturally: $(\pm 1, 0)$ and $(0, \pm 1)$.

The word problem for $\mathbb{Z}^2$ is decidable. So by Theorem 6 $\mathbb{Z}^2$-NFA $\subseteq \Sigma_1$.

On the other hand, a $\mathbb{Z}^2$-automaton is able to verify the correctness of computational history of an arbitrary Turing machine computation. The automaton expects a guess containing subsequent Turing machine configurations in subsequent rows of $\mathbb{Z}^2$. Correctness of computational history in this form is a conjunction of local conditions that can be verified by the automaton walking on $\mathbb{Z}^2$.

Thus $\mathbb{Z}^2$-NFA $= \Sigma_1$. As a corollary we get the following theorem.

Theorem 7. *Let G be a group with decidable word problem and $\mathbb{Z}^2$ is a subgroup of the group G. Then G-NFA $= \Sigma_1$.*

Acknowledgments

The author is deeply indebted to S. Tarasov for numerous valuable discussions and helpful references.

References

1. Amano, M., Iwama, K.: Undecidability on Quantum Finite Automata. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pp. 368–375. ACM, New York (1999)
2. Bach, E., Shallit, J.: Algorithmic Number Theory. Efficient Algorithms, vol. I. MIT Press, Cambridge (1996)

3. Bojańczyk, M.: Tree-Walking Automata, http://www.mimuw.edu.pl/~bojan/papers/twasurvey.pdf
4. Borchert, B.: Formal Language Characterizations of P, NP, and PSPACE (2003) (submitted)
5. Chandra, A.K., Kozen, D.C., Stockmeyer, L.J.: Alternation. J. of ACM 28, 114–133 (1981)
6. Cook, S.A.: Characterization of Pushdown Machines in Terms of Time-Bounded Computers. J. of ACM 18, 4–18 (1971)
7. Geffert, V.: A Communication Hierarchy of Parallel Computations. Theoretical Computer Science 198, 99–130 (1998)
8. Grädel, E., Kolaitis, P., Libkin, L.: Finite Model Theory and its Applications. Springer, Heidelberg (2007)
9. Holzer, M., McKenzie, P.: Alternating and Empty Alternating Auxiliary Stack Automata. Theoretical Computer Science 299, 307–326 (2003)
10. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (1979)
11. Hromkovič, J., Karhumäki, J., Rovan, B., Slobodová, A.: On the Power of Syncronization in Parallel Computations. Discr. Appl. Math. 32, 155–182 (1991)
12. Ibarra, O.H.: Characterizations of Some Tape and Time Complexity Classes of Turing Machines of Multihead and Auxiliary Stack Automata. J. of Comp. and Sys. Sci. 5, 88–117 (1971)
13. Shen, A., Vereshchagin, N.: Mathematical Logic and Computation Theory. Languages and Calculi. In: MCCME, Moscow (1999) (in Russian)
14. Smullyan, R.M.: Theory of Formal Systems. Princeton Univ. Press, Princeton (1961)
15. Vyalyi, M.N.: On Models of a Nondeterministic Computation. Electronic preprint, arXiv:0811.2586 (2008)

New Plain-Exponential Time Classes for Graph Homomorphism

Magnus Wahlström

Max-Planck-Institut für Informatik, Saarbrücken, Germany
wahl@mpi-inf.mpg.de

Abstract. A homomorphism from a graph G to a graph H (in this paper, both simple, undirected graphs) is a mapping $f : V(G) \to V(H)$ such that if $uv \in E(G)$ then $f(u)f(v) \in E(H)$. The problem $\mathrm{Hom}(G, H)$ of deciding whether there is a homomorphism is NP-complete, and in fact the fastest known algorithm for the general case has a running time of $O^*\left(n(H)^{cn(G)}\right)$,[1] for a constant $0 < c < 1$. In this paper, we consider restrictions on the graphs G and H such that the problem can be solved in plain-exponential time, i.e. in time $O^*\left(c^{n(G)+n(H)}\right)$ for some constant c. Previous research has identified two such restrictions. If $H = K_k$ or contains K_k as a core (i.e. a homomorphically equivalent subgraph), then $\mathrm{Hom}(G, H)$ is the k-coloring problem, which can be solved in time $O^*\left(2^{n(G)}\right)$ (Björklund, Husfeldt, Koivisto); and if H has treewidth at most k, then $\mathrm{Hom}(G, H)$ can be solved in time $O^*\left((k+3)^{n(G)}\right)$ (Fomin, Heggernes, Kratsch, 2007). We extend these results to cases of bounded cliquewidth: if H has cliquewidth at most k, then we can count the number of homomorphisms from G to H in time $O^*\left((2k+1)^{\max(n(G),n(H))}\right)$, including the time for finding a k-expression for H. The result extends to deciding $\mathrm{Hom}(G, H)$ when H has a core with a k-expression, in this case with a somewhat worse running time.

If G has cliquewidth at most k, then a similar result holds, with a worse dependency on k: We are able to count $\mathrm{Hom}(G, H)$ in time roughly $O^*\left((2k+1)^{n(G)} + 2^{2kn(H)}\right)$, and this also extends to when G has a core of cliquewidth at most k with a similar running time.

1 Introduction

A homomorphism from a graph G to a graph H is a mapping of the vertices of G to the vertices of H that preserves adjacency (i.e. neighbours in G are mapped to neighbours in H); we write $G \to H$ if one exists. The graph homomorphism problem $\mathrm{Hom}(G, H)$, of deciding whether $G \to H$, occupies an interesting place in terms of exact time complexity; let us make an overview of what is known and not.

[1] The notation $O^*(\cdot)$ signifies that polynomial factors have been ignored.

In the following, $\mathcal{G}$ and $\mathcal{H}$ are graph classes restricting the input: $\mathrm{Hom}(\mathcal{G}, \mathcal{H})$ is the class of problems $\mathrm{Hom}(G, H)$ for $G \in \mathcal{G}$, $H \in \mathcal{H}$. A dash (as in $\mathrm{Hom}(-, \mathcal{H})$) represents the class of all graphs, i.e. no restrictions. We are assuming that all graphs are simple and undirected. For more on the homomorphism problem in general, we refer to the book by Hell and Nešetřil [16]. Also, in addition to the decision and counting problems considered in this paper, optimization variants have been considered and studied in [14,13,12] and in [18].

1.1 Background

Beginning in some sense from the bottom, the polynomial cases of $\mathrm{Hom}(\mathcal{G}, -)$ and $\mathrm{Hom}(-, \mathcal{H})$ are completely known (assuming standard complexity theoretical assumptions). For $\mathrm{Hom}(-, \mathcal{H})$, there is a dichotomy due to Hell and Nešetřil [15]: $\mathrm{Hom}(-, \mathcal{H})$ is in P if every $H \in \mathcal{H}$ is bipartite, in which case the problem $\mathrm{Hom}(G, H)$ corresponds to checking whether G is bipartite, and NP-complete otherwise (including $\mathrm{Hom}(-, \{H\})$ for a single non-bipartite graph H). For example, $\mathrm{Hom}(-, \{K_3\})$ corresponds to 3-coloring.

For $\mathrm{Hom}(\mathcal{G}, -)$, there is a dichotomy due to Grohe [11]: $\mathrm{Hom}(\mathcal{G}, -)$ is in P if every $G \in \mathcal{G}$ either has bounded treewidth or is *homomorphically equivalent* to a graph of bounded treewidth (its *core* – see Section 2 for definitions). In every other case, the problem is W[1]-hard, thus not in P unless FPT=W[1] (the classes FPT and W[1] come from the field of parameterized complexity [6,9]). In particular, $\mathrm{Hom}(\mathcal{G}, -)$ is trivially in P for every finite $\mathcal{G}$.

There may be other polynomial cases of $\mathrm{Hom}(\mathcal{G}, \mathcal{H})$ where both sides are restricted, but we are not aware of this having been studied.

Moving on, the most famous special cases of homomorphism correspond to restricting $\mathcal{G}$ or $\mathcal{H}$ to being cliques. Specifically, $\mathrm{Hom}(G, K_k)$ is the problem of finding a k-colouring of G (thus explaining why the graph homomorphism problem is also known as H-colouring). For this problem, algorithms were recently, famously, presented by Björklund, Husfeldt, and Koivisto [1] which solve it in time $O^*\left(2^{n(G)}\right)$ for all k (older results, with different methods, exist with running times $O^*\left(c^{n(G)}\right)$ for $c > 2.4$ [19,7,2]). On the other side, $\mathrm{Hom}(K_k, H)$ amounts to finding a k-clique in H, which can of course be done in time $O^*\left(2^{n(H)}\right)$ for all k. These two cases also correspond to two other lower bounds on the time complexity of graph homomorphism (from [10]):

– If $\mathrm{Hom}(G, H)$ could be solved in time $O\left(p(n(G)) \cdot f(n(H))\right)$ for a polynomial $p(n)$ and any function $f(n)$, then there would be a polynomial-time algorithm for the NP-complete k-colouring problems (implying P=NP).
– If $\mathrm{Hom}(G, H)$ could be solved in time $O\left(f(n(G)) \cdot p(n(H))\right)$ for a polynomial $p(n)$ and any function $f(n)$, then the k-clique problem would be in FPT, implying FPT=W[1].

However, there is still a lot of room between these lower bounds and the best upper bound, which is $O^*\left(n(H)^{cn(G)}\right)$ for a constant $c < 1$, using an algorithm by Williams [21]. In particular, as asked by Fomin, Heggernes, and Kratsch [10], can $\mathrm{Hom}(G, H)$ be solved in time $O^*\left(c^{n(G)+n(H)}\right)$ for a constant c?

This question is still open, but for a generalisation of graph homomorphism we know (again under a complexity theoretical assumption) that the answer is negative. A $(d, 2)$-*CSP* instance consists of n variables, with d possible values, and arbitrary binary constraints (i.e. constraints on the values of pairs of variables). Thus, $\mathrm{Hom}(G, H)$ is an instance of $(n(H), 2)$-CSP. Traxler [20] recently showed that assuming ETH (i.e. unless 3-SAT has subexponential time algorithms [17]), $(d, 2)$-CSP requires time $\Omega^*(d^{cn})$ for some constant $c > 0$.[2]

1.2 Our Contributions

In this paper, we focus on restricted cases of graph homomorphism for which we can show the existence of plain-exponential time algorithms, i.e. cases of $\mathrm{Hom}(-, \mathcal{H})$ for which we can show algorithms running in time $O^*\left(c^{n(G)}\right)$ (and to a lesser extent the converse for $\mathrm{Hom}(\mathcal{G}, -)$). We are aware of two previous results of this type. One is the k-coloring problems, and instances equivalent to them (if $\chi(H) = \omega(H) = k$, then $\mathrm{Hom}(G, H)$ is again equivalent to k-coloring). The other is the case of bounded treewidth: if the treewidth of H is at most k, then Fomin, Heggernes, and Kratsch showed an algorithm for deciding $\mathrm{Hom}(G, H)$ with a running time of $O^*\left((k + 3)^{n(G)}\right)$ [10].

We connect the question to the concept of cliquewidth [4,5]: we show that if H has cliquewidth at most k (written $\mathrm{cwd}(H) \leq k$), then $\mathrm{Hom}(G, H)$ can be counted in time $O^*\left((2k + 1)^{\max(n(G),n(H))}\right)$, and if the core of H has cliquewidth at most k, then $\mathrm{Hom}(G, H)$ can be decided in time $(O(k))^{\max(n(G),n(H))}$. Both results include the time for finding a k-expression. As far as the existence of plain-exponential time algorithms goes, this extends both previous results, as cliques have cliquewidth 2, and the cliquewidth of a graph is bounded by a function of its treewidth [5].

For restrictions $\mathrm{Hom}(\mathcal{G}, -)$, we show a plain-exponential algorithm for the same case: if $\mathrm{cwd}(G) \leq k$ for every $G \in \mathcal{G}$, then $\mathrm{Hom}(\mathcal{G}, -)$ can be counted in time $O^*\left(c_k^{\max(n(G),n(H))}\right)$, where the constant c_k depends on k. Here, the dependency is worse: we get $c_k = 2^{2k}$.

The organisation of the paper is as follows: Section 2 contains preliminaries, and a simple exponential-time algorithm for finding k-expressions for bounded k. Section 3 presents our result for $\mathrm{Hom}(-, \mathcal{H})$, Section 4 presents our result for $\mathrm{Hom}(\mathcal{G}, -)$, and Section 5 contains some conclusions and further questions.

2 Preliminaries

2.1 Graph Homomorphism

Definition 1. *Given graphs* $G = (V_G, E_G)$ *and* $H = (V_H, E_H)$*, a* homomorphism *from* G *to* H *is a mapping* $f : V_G \to V_H$ *such that for any edge* uv *in* G*,*

[2] Let us also remark that [20] excludes the existence of algorithms with a running time of $O^*\left(c^{d+n}\right)$ for $(d, 2)$-CSP, as performing the variable reduction of [20] with a target domain size of $\log n$ would produce a running time like $d^{O(n/\log\log n)}$ for $(d, 2)$-CSP, which would count as subexponential.

$f(u)f(v)$ is an edge in H. If one exists, we say that G is homomorphic to H, written $G \to H$. The graph homomorphism problem $\mathrm{Hom}(G, H)$ *is the problem of deciding whether $G \to H$. For classes of graphs $\mathcal{G}$ and $\mathcal{H}$,* $\mathrm{Hom}(\mathcal{G}, \mathcal{H})$ *is the graph homomorphism problem restricted to inputs $G \in \mathcal{G}$, $H \in \mathcal{H}$. A graph class given as $-$ (e.g.* $\mathrm{Hom}(-, \mathcal{H})$ *and* $\mathrm{Hom}(\mathcal{G}, -)$*) means the class of all graphs.*

We need a few basic facts about homomorphisms (see [16] for more).

Proposition 1. *Homomorphisms compose: if $G \to H$ and $H \to H'$, then $G \to H'$. Also, $G \to K_k$ if and only if G is k-colourable.*

Definition 2. *G,H are* homomorphically equivalent *if $G \to H$ and $H \to G$. A* core *is a graph G which is not homomorphically equivalent to any proper subgraph of itself; a core of a graph G is a minimal subgraph of G which is homomorphically equivalent to G.*

Proposition 2. *All cores of a graph G are isomorphic, thus we speak of* the *core of G. Also, the core of G is an induced subgraph of G.*

2.2 Cliquewidth

Definition 3. *A k-*expression *for a graph G is an expression using k different labels that constructs G using the following steps [4,5]:*

- *$\cdot_i$: Construct a graph with one vertex, giving the vertex the label i.*
- *$\rho_{i\to j}(G)$: Relabel all vertices of G with label i to label j.*
- *$\eta_{ij}(G)$, for $i \neq j$: Add edges between all vertices of labels i and j in G, i.e. add edges uv for any vertices u, v where u has label i and v has label j.*
- *$G_1 \oplus G_2$: Create the disjoint union of G_1 and G_2.*

A graph G has cliquewidth *k, denoted* $\mathrm{cwd}(G) = k$*, if k is the smallest number such that there exists a k-expression for G. A k-expression is* linear *if one side of every disjoint union operation is a single vertex (e.g. $H' \oplus \cdot_i$ rather than a general $H' \oplus H''$). The* linear cliquewidth *of H is the smallest number such that there exists a linear k-expression for H.*

Example 1. A 2-expression for K_4 is $\eta_{12}(\rho_{2\to1}(\eta_{12}(\rho_{2\to1}(\eta_{12}(\cdot_1 \oplus \cdot_2)) \oplus \cdot_2)) \oplus \cdot_2)$. This expression is linear. In general, all cliques K_k with $k \geq 2$ have linear cliquewidth and cliquewidth both 2.

Definition 4. *We say that a k-expression, constructing a graph G, is in* standard form *if for every operation $G_1 \oplus G_2$, $G_i = G[V(G_i)]$ for $i = 1, 2$, i.e. all edges in G between vertices of G_1 (and G_2) have already been created before the $\oplus$ operation. We will throughout this paper assume that all k-expressions are in standard form. Note that a k-expression can be easily transformed to an equivalent k-expression in standard form (by recursively adding η-operations to the expressions for the graphs G_1, G_2 on both sides of a union operation $G_1 \oplus G_2$), and that the length of the result is still polynomial in n.*

Proposition 3 ([5]). *If G is a graph and $G[S]$ for $S \subseteq V(G)$ an induced subgraph, then* $\mathrm{cwd}(G[S]) \leq \mathrm{cwd}(G)$.

The problem of finding the cliquewidth of a graph is NP-hard in general [8], and the complexity of finding a k-expression for a fixed $k > 3$ is unknown – it is not even known whether it is polynomial. (A graph has a 1-expression only if it has no edges, and a 2-expression if and only if it has no induced P_4 (path on four vertices); for $k = 3$ a polynomial-time algorithm exists [3].) However, in this paper, we will not need polynomial-time constructions: since our algorithms take $(O(k))^n$ time, we can afford to use an exact algorithm with the same behaviour for finding a k-expression. This algorithm is given next.

2.3 Constructing a k-Expression for Bounded k

We now show how to find a k-expression for a graph G, provided that we are allowed to use time $(O(k))^n$. We want to stress that we have not made any particular effort to minimize the running time of this construction; at the moment, it is fast enough that it is not the bottleneck of our homomorphism algorithm, which is all we need.

Theorem 1. *If* $\mathrm{cwd}(G) \leq k$*, then a k-expression for G can be found in time* $O^*\left((2k+1)^{n(G)}\right)$.

Proof. We will create a table of size $(k+1)^{n(G)}$ containing a k-expression for each partitioning $S_1, \ldots, S_k$ of each vertex set $S \subseteq V(G)$ (with the vertices of label i being S_i). Let the sets S be considered in order of increasing cardinality. For the base cases $|S| = 1$, the expression is trivial; if $|S| > 1$, then the corresponding expression must contain a union operation $G' \oplus G''$, for graphs $G' = G[S']$ and $G'' = G[S'']$ corresponding to a split of S into vertex sets S', S'', so G' and G'' are graphs for which we already know k-expressions. We will create the expressions for all partitionings of S in two phases, first going through all expressions without relabelling operations ρ (e.g. for creating partitionings of S with k non-empty label classes), then using this data to find expressions which do contain relabelling operations. Thus, first split S all $(2k)^{|S|}$ ways into partitionings of S' and S'', verify that there are k-expressions for these partitionings for the resulting G' and G'', and that the existence of edges connecting G' and G'' follows the pattern of vertex labels. If so, register a k-expression for the resulting partitioning of S (keeping only one expression per partitioning). This phase takes $O^*\left((2k)^{|S|}\right)$ time.

In the second phase, go through the possible partitionings of S in order of decreasing number of non-empty partitions, and consider for each partitioning of S each possible single relabelling operation $\rho_{i\to j}$ in turn. If appending $\rho_{i\to j}$ to a k-expression leads to a k-expression for a partitioning of S that previously did not have one, register the new k-expression. This phase uses only polynomial time for each partitioning of S, thus time $O^*\left(k^{|S|}\right)$. Also note that every partitioning of S for which a k-expression is possible will have received a k-expression at the end of this process.

Ignoring polynomial factors, the whole algorithm runs in time

$$\sum_{S \subseteq V} (2k)^{|S|} = \sum_{i=0}^{n} \binom{n}{i} (2k)^i = (2k+1)^n$$

and creates a k-expression for G. $\square$

The case of linear k-expressions is easier:

Theorem 2. *If G has a linear k-expression, then one can be found in time $O^*\big((k+2)^{n(G)}\big)$.*

Proof. The algorithm runs as the previous proof, except that in the first phase for each S, instead of $(2k)^{|S|}$ ways to split $G[S]$ into $G[S'] \oplus G[S'']$, there are now only $(k+1)^{|S|}$ ways to split (k labels on one side of $\oplus$, plus one on the other). The second phase is then identical. $\square$

3 Homomorphism to Graphs with Bounded Cliquewidth

In this section, we present our main result: $\mathrm{Hom}(-, \mathcal{H})$ can be counted in time $O^*\big((2k+1)^{\max(n(G),n(H))}\big)$ when every $H \in \mathcal{H}$ has cliquewidth at most k. Per Theorem 1, we can find a k-expression for a given H in this time; we start by showing how to use this expression to solve $\mathrm{Hom}(G, H)$, then consider some variations.

3.1 Homomorphism to a Graph with a k-Expression

We now show how to decide or count $\mathrm{Hom}(G, H)$ in time $O^*\big((2k+1)^{n(G)}\big)$ assuming that a k-expression for H has already been given. The method is a direct application of dynamic programming techniques.

Theorem 3. *Given a k-expression for H,* $\mathrm{Hom}(G, H)$ *can be counted in time $O^*\big((2k+1)^{n(G)}\big)$.*

Proof. Let the k-expression for H be h. The algorithm will work as a recursion over the disjoint union operations $\oplus$ of h through dynamic programming: for every operation $H' \oplus H''$, we will use caches for H' and H'' giving for each partitioning $S_1, \dots, S_k$ of a set $S \subseteq V(G)$ the number of homomorphisms from $G[S]$ to H' (resp. to H'') mapping S_i to vertices of label i. We will call a k-expression *trivial* if it contains no disjoint union operation, i.e. the graph H is a single vertex, assume with label i. These cases form the base cases of the recursion, and the corresponding cache is easily constructed (if any $S_j \neq \emptyset$ for $j \neq i$, or if $E(G[S]) \neq \emptyset$ then the answer is 0, otherwise 1).

Assuming that h is not trivial, let h be a sequence of non-$\oplus$ operations, referred to as the *head* of h, applied to some expression $h' \oplus h''$ (where h' and h'' are k-expressions generating the graphs H' resp. H'' with vertex sets V'_H resp. V''_H). As stated, we assume that all edges in H between vertices of H' are added by h'

(and likewise for H''). Let there be caches for h' and h'' as described above. We can then create a cache for h that answers the same questions for $\mathrm{Hom}(G[S], H)$ for all possible partitionings $S_1, \ldots, S_k$ of each $S \subseteq V(G)$ by looping through all $(2k+1)^{n(G)}$ assignments of $V(G)$ to either not be included in S, or to one of the $2k$ combinations of vertex label and corresponding graph (H' or H'').

Specifically, for each such considered assignment of vertices, verify in polynomial time that the split of S into $S' = S'_1, \ldots, S'_k$ mapping to H' and $S'' = S''_1, \ldots, S''_k$ mapping to H'' does not break any edge (i.e. for each edge $u'u''$ in G such that u' is assigned to label i in H' and u'' to label j in H'', check that an η_{ij} operation exists in the head of h), query the caches for the corresponding numbers of homomorphisms $\mathrm{Hom}(G[S'], H')$ and $\mathrm{Hom}(G[S''], H'')$ following the partitionings of S', S'', and if all tests pass, add the resulting number of homomorphisms to the right entry of the cache for h (remembering to account for relabelling operations ρ in the head of h).

As constructing the cache for a trivial expression $\mathrm{Hom}(G[S], \cdot_i)$ is trivial, and every further cache can be constructed from previous caches, by induction the cache corresponding to the outermost query $\mathrm{Hom}(G, H)$ can be created in time $O^*\big((2k+1)^{n(G)}\big)$, and the number of homomorphisms $G \to H$ can be calculated from this in time $O^*\big(k^{n(G)}\big)$. □

Remark 1. Variations of the problem can be solved using the same method; we sketch two here. First, a homomorphism $f : V(G) \to V(H)$ is *vertex-surjective* if $f^{-1}(v) \neq \emptyset$ for every $v \in V(H)$. We can make our method consider only vertex-surjective homomorphisms by enforcing the condition in the base cases $\cdot_i$; the property will then be maintained in the inductive steps.

Second, Gutin et al. [14] considered an optimization variant, where for every $u \in V(G)$ and $v \in V(H)$ there is an associated *cost* $c_v(u)$ of mapping u to v, and a homomorphism f minimizing $\sum_{u \in V(G)} c_{f(v)}(u)$ is sought. The property that the caches only contain information on minimum-cost homomorphisms for each query can be enforced in the base cases (where each suggested mapping has one fix cost only – the operations $\cdot_i$ are assumed to now also give the name of the new vertex), and maintained through the inductive steps (where the cost of a homomorphism to H_0 built from $H' \oplus H''$ is the sum of the costs of the homomorphisms to H', H'').

Corollary 1. $\mathrm{Hom}(G, H)$ *can be counted in time* $O^*\big((2k+1)^{\max(n(G),n(H))}\big)$ *if the cliquewith of* H *is at most* k*.*

If we settle for linear k-expressions, the base $(2k+1)$ in the running time can be improved to $(k+2)$. The proof is omitted, as it is very close to Theorem 3.

Theorem 4. $\mathrm{Hom}(G, H)$ *when* H *has a linear* k*-expression can be counted in time* $O^*\big((k+2)^{\max(n(G),n(H))}\big)$*.*

3.2 Extension to Core of H

If H is itself not a core, then it may be that H has large cliquewidth but that the core H_C of H has bounded cliquewidth, such as the example that $\chi(H) = \omega(H)$

(in which case a clique of H is the core). For the decision case, we can handle these cases as well, as $G \to H$ if and only if $G \to H_C$, at the cost of a bigger base of the running time.

Theorem 5. *If a graph G has a core G_C with $\operatorname{cwd}(G_C) \leq k$, then G_C and a k-expression for G_C can be found in time $O^*\left((2(2k+1))^{n(G)}\right)$.*

Proof. Since $\operatorname{cwd}(G[S]) \leq \operatorname{cwd}(G)$ for induced subgraphs $G[S]$ of G, every induced subgraph of G_C will have a cliquewidth bounded by k. Thus, we can construct k-expressions for progressively larger induced subgraphs $G[S]$, as in Theorem 1, getting in time $O^*\left((2k+1)^{n(G)}\right)$ a list of k-expressions for those induced subgraphs $G[S]$ that have any. Then, for each such $G[S]$ we check $\operatorname{Hom}(G, G[S])$ using Theorem 3, using time bounded by $O^*\left((2(2k+1))^{n(G)}\right)$. If the core G_C has $\operatorname{cwd}(G_C) \leq k$, then eventually we will reach $G_C = G[S]$ and find a homomorphism, thus confirming $G[S] = G_C$. □

Corollary 2. $\operatorname{Hom}(G, H)$ *when the core of H has cliquewidth at most k can be decided in time $O(k)^{\max(n(G),n(H))}$.*

4 Left-Side Restrictions

Although our main focus is restrictions on $\mathcal{H}$, as such restrictions more commonly represent typical problem classes (in $\operatorname{Hom}(\mathcal{G}, \mathcal{H})$, and more generally in the case of relational homomorphisms, commonly $\mathcal{H}$ can be seen as representing a problem type, and $\mathcal{G}$ representing an instance of this problem), we show in this section that restricting the cliquewidth of $\mathcal{G}$ also leads to plain-exponential time cases, although with a rather worse dependency on the cliquewidth: if $\operatorname{cwd}(G) = k$, then we can solve $\operatorname{Hom}(G, H)$ in time $O^*\left(2^{2kn(H)}\right)$, plus the time for constructing a k-expression for G. Note, however, that simply converting the trivial running time of $O^*\left(n(H)^{n(G)}\right)$ to $O^*\left(n(H)^{f(k)}\right)$ would have lead to new polynomial cases, including a polynomial-time algorithm for k-clique, and so could not be expected.

Theorem 6. *If a k-expression for G is known, then* $\operatorname{Hom}(G, H)$ *can be counted in time $O^*\left(2^{2kn(H)}\right)$.*

Proof. The problem is once again solved through dynamic programming over the components of the k-expression, though this time, we need to store more information in the caches. Specifically, our cache for a graph G' or G'' occurring in an expression $G' \oplus G''$ will be used to give, for k not necessarily disjoint subsets V_i of $V(H)$, the number of homomorphisms from G' to H such that the targets in $V(H)$ of vertices of label i in G' are exactly V_i (i.e. homomorphisms f such that $V_i = \{f(u) \mid u \in V(G) \text{ has label } i\}$ for $i = 1, \ldots, k$). Once again, the caches for the base cases $\cdot_i$ are trivial (since G' is then a single vertex, it can be mapped to any single target in $V(H)$).

Thus, assume that we are working on a k-expression g_0 creating a graph G_0, consisting of a *head* of edge creations η and relabelling operations ρ, applied to the result of a disjoint union operation $G' \oplus G''$. Every pair of homomorphisms $f' : V(G') \to V(H)$ and $f'' : V(G'') \to V(H)$ compose into one mapping $f : V(G_0) \to V(H)$, and our task is to check whether f is a homomorphism, i.e. to verify that all edges created in the head of g_0 are mapped to edges in H. This can be done using the information in the caches: Assume that all edges between labels i and j are created in the head of g_0, and let V'_i resp. V''_j be the targets of label i in G' resp. label j in G''. If there are $u \in V'_i$, $v \in V''_j$ such that uv is not an edge of H (in particular, if $u = v$), then some edge in G_0 is broken by f: Since V'_i and V''_j are the exact targets, there are vertices $u' \in V'_i$ with $f(u') = u$ and $v'' \in V''_j$ with $f(v'') = v$, and an edge $u'v''$ in G_0 which is broken by f. The same check holds for label j in G' and label i in G'', and if both checks pass, then no edge between labels i and j are broken by f. Thus, for a particular combination of targets V'_i of labels of G' and V''_i of labels of G'', we can decide using a polynomial number of polynomial-time checks whether the created mappings f would be homomorphisms or not.

Now the cache for G_0 can be assembled using queries to the caches for G', G'': With $2^{n(H)}$ options for each set of targets, the time for this step is bounded by $O^*\left((2^{n(H)})^{2k}\right) = O^*\left(2^{2kn(H)}\right)$. By induction, the cache for G can be created, and the total number of homomorphisms can be counted from this. □

Corollary 3. *If G has cliquewidth at most k, then* Hom(G, H) *can be counted in plain-exponential time $O^*\left((2k+1)^{n(G)} + 2^{2kn(H)}\right)$. Under the weaker assumption that the core of G has cliquewidth at most k,* Hom(G, H) *can be decided in time $O^*\left((2(2k+1))^{n(G)} + 2^{2kn(H)}\right)$.*

5 Conclusions

We have showed that homomorphisms from G to H can be counted in plain-exponential time $O^*\left(c_k^{n(G)+n(H)}\right)$ when either G or H has cliquewidth at most k (c_k depending on k). This unifies and extends previous plain-exponential time classes for the problem.

Nevertheless, there is a simple case of plain-exponential time for Hom$(-, \mathcal{H})$ which our results do not cover: if the graphs $H \in \mathcal{H}$ have bounded degree $\Delta(H) \leq d$, then we can trivially solve the problem through branching in time $O^*\left(d^{n(H)}\right)$. Is there yet another case of plain-exponential time Hom$(-, \mathcal{H})$ which covers both the cliquewidth and the bounded degree cases?

Also, we have to ask about homomorphism for directed graphs, and relational homomorphism in general [16,11]. These problems cover the CSP problems, so we know that plain-exponential time algorithms are impossible in general [20]. Can we identify plain-exponential time classes for these problems?

References

1. Björklund, A., Husfeldt, T., Koivisto, M.: Set partitioning via inclusion–exclusion. SIAM J. Comput. (to appear); prelim. versions in FOCS 2006
2. Byskov, J.M.: Exact Algorithms for Graph Colouring and Exact Satisfiability. PhD thesis, University of Aarhus (2005)

3. Corneil, D.G., Habib, M., Lanlignel, J.-M., Reed, B.A., Rotics, U.: Polynomial time recognition of clique-width ≤ 3 graphs (extended abstract). In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 126–134. Springer, Heidelberg (2000)
4. Courcelle, B., Engelfriet, J., Rozenberg, G.: Handle-rewriting hypergraph grammars. J. Comput. System Sci. 46, 218–270 (1993)
5. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. Discrete Applied Mathematics 101(1-3), 77–114 (2000)
6. Downey, R.G., Fellows, M.: Parameterized Complexity. Monographs in Computer Science. Springer, Heidelberg (1999)
7. Eppstein, D.: Small maximal independent sets and faster exact graph coloring. J. Graph Algorithms Appl. 7(2), 131–140 (2003)
8. Fellows, M.R., Rosamond, F.A., Rotics, U., Szeider, S.: Clique-width minimization is NP-hard. In: STOC 2006, pp. 354–362 (2006)
9. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006)
10. Fomin, F.V., Heggernes, P., Kratsch, D.: Exact algorithms for graph homomorphisms. Theory of Comput. Syst. 41(2), 381–393 (2007)
11. Grohe, M.: The complexity of homomorphism and constraint satisfaction problems seen from the other side. J. ACM 54(1) (2007)
12. Gutin, G., Hell, P., Rafiey, A., Yeo, A.: A dichotomy for minimum cost graph homomorphisms. European J. Combin. 29, 900–911 (2008)
13. Gutin, G., Rafiey, A., Yeo, A.: Minimum cost and list homomorphisms to semicomplete digraphs. Discrete Applied Mathematics 154(6), 890–897 (2006)
14. Gutin, G., Rafiey, A., Yeo, A., Tso, M.: Level of repair analysis and minimum cost homomorphisms of graphs. Discrete Applied Mathematics 154(6), 881–889 (2006)
15. Hell, P., Nešetřil, J.: On the complexity of H-coloring. J. Comb. Theory, Ser. B 48(1), 92–110 (1990)
16. Hell, P., Nešetřil, J.: Graphs and Homomorphisms. Oxford University Press, Oxford (2004)
17. Impagliazzo, R., Paturi, R., Zane, F.: Which problems have strongly exponential complexity? J. Comput. Syst. Sci. 63(4), 512–530 (2001)
18. Jonsson, P., Nordh, G., Thapper, J.: The maximum solution problem on graphs. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 228–239. Springer, Heidelberg (2007)
19. Lawler, E.: A note on the complexity of the chromatic number problem. Information Processing Letters 5, 66–67 (1976)
20. Traxler, P.: The time complexity of constraint satisfaction. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 190–201. Springer, Heidelberg (2008)
21. Williams, R.: A new algorithm for optimal 2-constraint satisfaction and its implications. Theor. Comput. Sci. 348(2-3), 357–365 (2005)

Languages Recognized with Unbounded Error by Quantum Finite Automata*

Abuzer Yakaryilmaz and A.C. Cem Say

Boğaziçi University, Department of Computer Engineering,
Bebek 34342 İstanbul, Turkey
{abuzer,say}@boun.edu.tr

Abstract. We prove the following facts about the language recognition power of Kondacs-Watrous quantum finite automata in the unbounded error setting: One-way automata of this kind recognize all and only the stochastic languages. When the tape head is allowed two-way (or even "1.5-way") movement, more languages become recognizable. This leads to the conclusion that quantum Turing machines are more powerful than probabilistic Turing machines when restricted to constant space bounds.

1 Introduction

Several alternative models [3, 4, 6, 10, 13, 17, 19] of quantum finite automata (QFA's) have been studied in the recent years. Most of the attention in this regard has been focused on the classes of languages recognized by these machines with bounded error [3, 5, 10, 13]. In this paper, we examine the computational power of one of the most popular QFA models, the measure-many (Kondacs-Watrous) QFA, in the unbounded error setting. We give a complete characterization of the class of languages recognized by one-way QFA's of this kind; they turn out to recognize all and only the stochastic languages. We also show that allowing the tape head to "stay put" for some steps during its left-to-right traversal of the input increases the language recognition power of these QFA's. This contrasts the situation in the classical probabilistic models, where two-way and one-way automata are equivalent in power in this setting [12]. We conclude that quantum Turing machines are strictly more powerful than their probabilistic counterparts when restricted to constant space bounds.

The rest of this paper is structured as follows: Section 2 contains the relevant background information. Section 3 presents our results. Section 4 is a conclusion.

2 Preliminaries

In this section, we give a brief review of the definitions and facts that will be used in the rest of the paper.

* This work was partially supported by the Boğaziçi University Research Fund with grant 08A102 and by the Scientific and Technological Research Council of Turkey (TÜBİTAK) with grant 108E142.

2.1 Classical Automata

A *1-way probabilistic finite automaton* (1PFA) [22] with $n \in \mathbb{Z}^+$ states is a 5 tuple $\mathcal{P} = (S, \Sigma, \{\mathsf{A}_\sigma \mid \sigma \in \Sigma\}, \mathsf{v}_0, F)$, where

1. $S = \{s_1, \cdots, s_n\}$ is the set of states,
2. Σ is the finite input alphabet,
3. A_σ is the $n \times n$ real-valued stochastic transition matrix for symbol σ, that is, $\mathsf{A}_\sigma(i,j)$ is the value of the transition probability from state s_i to state s_j when reading symbol σ,
4. v_0 is a $1 \times n$ vector representing the initial distribution of the states, that is, the i^{th} entry of v_0 is the probability that $\mathcal{P}$ is initially in state s_i, and,
5. $F \subseteq S$ is the set of accepting states.

For an input string $w \in \Sigma^*$, the computation of $\mathcal{P}$ can be traced using the relations

$$\mathsf{v}_j = \mathsf{v}_{j-1}\mathsf{A}_{w_j} \tag{1}$$

$$\mathsf{A}_w = \mathsf{A}_{w_1}\mathsf{A}_{w_2}\cdots\mathsf{A}_{w_{|w|}} \tag{2}$$

$$\mathsf{v}_{|w|} = \mathsf{v}_0\mathsf{A}_w, \tag{3}$$

where w_i is the i^{th} symbol of w and v_j is the vector of states after step j $(1 \leq i,j \leq |w|)$. The acceptance probability of w is

$$f_{\mathcal{P}}(w) = \sum_{s_i \in F} \mathsf{v}_{|w|}(i),$$

where $\mathsf{v}_{|w|}(i)$ denotes the i^{th} entry of $\mathsf{v}_{|w|}$. The language $L \subseteq \Sigma^*$ recognized by $\mathcal{P}$ with cut-point $\lambda \in [0,1)$ is defined as

$$L = \{w \mid w \in \Sigma^*, f_{\mathcal{P}}(w) > \lambda\}.$$

Languages recognized by 1PFA's with cut-point form the class of **stochastic languages**.

Theorem 1. *[21] If L is a stochastic language, then for any $\lambda \in (0,1)$, there exists a 1PFA that recognizes L with cut-point λ.*

The generalized probabilistic finite automaton (GPFA) [23] model is a generalization of the 1PFA model where neither the transition matrices nor the state vectors need to be stochastic. Additionally, a GPFA has a final vector f, which is a column vector with real entries, instead of the set of accepting states. Formally, a *generalized probabilistic finite automaton* (GPFA) with $n \in \mathbb{Z}^+$ states is a 5 tuple $\mathcal{G} = (S, \Sigma, \{\mathsf{A}_\sigma \mid \sigma \in \Sigma\}, \mathsf{v}_0, \mathsf{f})$, where

1. S is the set of n states,
2. Σ is the finite input alphabet,
3. A_σ is the $n \times n$ real-valued transition matrix for symbol σ,

4. v_0 is the real-valued initial $1 \times n$ vector, and,
5. f is the real-valued final $n \times 1$ vector.

For an input string $w \in \Sigma^*$, the acceptance probability of w by GPFA $\mathcal{G}$ is calculated as

$$f_{\mathcal{G}}(w) = \mathsf{v}_0 \mathsf{A}_w \mathsf{f} = \mathsf{v}_{|w|}\mathsf{f},$$

where A_w is as defined in Equation 2. Note that the range of $f_{\mathcal{G}}$ is $\mathbb{R}$. The language $L \subseteq \Sigma^*$ recognized by $\mathcal{G}$ with cut-point $\lambda \in \mathbb{R}$ is defined as

$$L = \{w \mid w \in \Sigma^*, f_{\mathcal{P}}(w) > \lambda\}.$$

It is known that the class of languages recognized by GPFA's with cut-point is precisely the class of stochastic languages [23].

Theorem 2. *[23] If L is recognized with cut-point $\lambda_1 \in \mathbb{R}$ by a GPFA with n states, then there exist a 1PFA $\mathcal{P}$ with $O(n^2)$ states and a cut-point $\lambda_2 \in [0, 1)$ such that $\mathcal{P}$ recognizes L with cut-point λ_2.*

Another generalization of the 1PFA is the *two-way probabilistic automaton* (2PFA) model [14], in which the input string is viewed as written on a tape flanked by two end-markers, and the machine has a tape head which can move to the left or stay put, as well as moving to the right. This additional capability does not increase the recognition power; the class of languages recognized by 2PFA's with cut-point is again equal to the stochastic languages [12].

2.2 Quantum Automata

A *1-way Kondacs-Watrous quantum finite automaton* (KWQFA) [13] with $n \in \mathbb{Z}^+$ states is a 5-tuple $\mathcal{M} = (Q, \Sigma, \{\mathsf{U}_\sigma \mid \sigma \in \Sigma \cup \{¢, \$\}\}, Q_{acc}, Q_{rej})$, where

1. $Q = \{q_1, \cdots q_n\}$ is the set of states, such that q_1 is the initial state,
2. Σ is the finite input alphabet not containing the symbols ¢ and \$,
3. U_σ is the $n \times n$ complex-valued unitary transition matrix for symbol $\sigma \in \Sigma \cup \{¢, \$\}$ such that $\mathsf{U}_\sigma(j, i) = \alpha$ if the amplitude of the transition from q_i to q_j is α when reading the symbol σ,
4. Q_{acc} and Q_{rej}, disjoint subsets of Q, are the sets of accepting and rejecting states.

Additionally, $Q_{non} = Q \setminus (Q_{acc} \cup Q_{rej})$ is the set of non-halting states; $\mathsf{P}_{\mathsf{non}}$, $\mathsf{P}_{\mathsf{acc}}$, and $\mathsf{P}_{\mathsf{rej}}$ are diagonal zero-one projection matrices, which project the state vector onto the subspaces of non-halting, accepting, and rejecting states, such that $\mathsf{P}_{\mathsf{non}}(j, j) = 1$ if $q_j \in Q_{non}$, $\mathsf{P}_{\mathsf{acc}}(j, j) = 1$ if $q_j \in Q_{acc}$, and $\mathsf{P}_{\mathsf{rej}}(j, j) = 1$ if $q_j \in Q_{rej}$, respectively; $\Gamma = \Sigma \cup \{¢, \$\}$ is the tape alphabet where ¢ and \$ are the end-markers, and for input string $w \in \Sigma^*$, the tape contains $¢w\$$.

The state vector, by an unfortunate twist of convention between probabilistic and quantum automata, is a column vector, and is denoted as $|\mathsf{u}\rangle$. Its conjugate transpose is a row vector, and is denoted by $\langle \mathsf{u}|$.

The computation of $\mathcal{M}$ proceeds as follows: $|\mathsf{u}_0\rangle$ is the initial state vector, where $|\mathsf{u}_0\rangle(1) = 1$ and all other entries are zeros, and $|\mathsf{u}_0^n\rangle = |\mathsf{u}_0\rangle$. For a given input string $w \in \Sigma^*$, $\mathbf{w} = ¢w\$$,

$$|\mathsf{u}_i\rangle = \mathsf{U}_{\mathbf{w}_i}|\mathsf{u}_{i-1}^n\rangle, \tag{4}$$

$$|\mathsf{u}_i^n\rangle = \mathsf{P}_{\mathsf{non}}|\mathsf{u}_i\rangle, |\mathsf{u}_i^a\rangle = \mathsf{P}_{\mathsf{acc}}|\mathsf{u}_i\rangle, |\mathsf{u}_i^r\rangle = \mathsf{P}_{\mathsf{rej}}|\mathsf{u}_i\rangle, \tag{5}$$

$$P_{\mathcal{M},acc}(i) = P_{\mathcal{M},acc}(i-1) + \langle \mathsf{u}_i^a|\mathsf{u}_i^a\rangle, \tag{6}$$

$$P_{\mathcal{M},rej}(i) = P_{\mathcal{M},rej}(i-1) + \langle \mathsf{u}_i^r|\mathsf{u}_i^r\rangle, \tag{7}$$

where $1 \leq i \leq |\mathbf{w}|$, $|\mathsf{u}_i\rangle$ is the state vector after the i^{th} step, and $P_{\mathcal{M},acc}$ and $P_{\mathcal{M},rej}$ are finite sequences that trace the acceptance and rejection probabilities that have accumulated so far during the computation, with initial values $P_{\mathcal{M},acc}(0) = 0$ and $P_{\mathcal{M},rej}(0) = 0$, respectively. Equations 4 and 5 mean that the machine undergoes two operations in each step. First, its state vector evolves according to the unitary transformation dictated by the scanned symbol. Then, it is measured to see whether it has accepted, rejected, or not halted yet. As seen in Equations 6 and 7, the acceptance and rejection probabilities are calculated using the amplitudes of the relevant states. Halting states "drop out" of the state vector, and computation continues with only the non-halting states having nonzero amplitude.

The probability that $\mathcal{M}$ accepts input w is

$$f_{\mathcal{M}}(w) = P_{\mathcal{M},acc}(|¢w\$|),$$

and the language $L \subseteq \Sigma^*$ recognized by $\mathcal{M}$ with cut-point $\lambda \in [0, 1)$ is defined as

$$L = \{w \mid w \in \Sigma^*, f_{\mathcal{M}}(w) > \lambda\}.$$

The class of languages recognized by KWQFA's with cut-point (i.e. with unbounded error) has been studied by Brodsky and Pippenger [10], who named it **UMM**, and proved some closure properties[1]. Analogously to Theorem 1, any language in UMM can be recognized by a KWQFA with cutpoint $\frac{1}{2}$.

KWQFA's can be generalized by allowing the tape head more freedom of movement.

A *1.5-way quantum finite automaton* (1.5QFA) [3] can be seen as a generalized KWQFA where the tape head is allowed to stay put, and does not have to move right in all steps. The transition function $\delta : Q \times \Gamma \times Q \times \{0, 1\} \to \mathbb{C}$ of a 1.5QFA, where Q and Γ are as defined above, is interpreted as follows: For each $q, q^{'} \in Q$, $\delta \in \Gamma$ and $d \in \{0, 1\}$, $\delta(q, \sigma, q^{'}, d)$ is the amplitude with which the machine currently in state q and scanning symbol σ will change to state $q^{'}$ and move its head d symbols to the right.

Not every transition function of the form described above can be used in a 1.5QFA. The additional restrictions imposed by quantum theory on δ are

[1] Note that the proof in [10] that UMM is closed under complementation contains an error, as the construction presented there does not work for the case of input strings whose acceptance probabilities equal the cut-point.

described in detail in [13]. In the particular 1.5QFA we will describe in the proof of Theorem 4, every transition entering the same state involves the tape head moving in the same direction (either right or stationary). We represent this feature of a state q using the appropriate one of the notations $\overrightarrow{q}, \downarrow q$ for this state in the machine description. With this simplification, considering the Hilbert space $\ell_2(Q)$, a syntactically correct 1.5QFA can be specified easily by just providing a unitary operator $U_\sigma : \ell_2(Q) \rightarrow \ell_2(Q)$ for each symbol $\sigma \in \mathit{\Gamma}$, exactly as described earlier for one-way KWQFA's. δ can then be specified to be just

$$\delta(q,\sigma,q^{'},d) = \begin{cases} \langle q^{'}|U_\sigma|q\rangle & \text{if } D(q^{'}) = d \\ \quad 0 & \text{if } D(q^{'}) \neq d \end{cases},$$

where the function $D : Q \rightarrow \{0,1\}$ maps each state to the single direction in which all incoming transitions to this state move the tape head.

A pair of the form *(state, head position)* is called a *configuration* of a 1.5QFA. A 1.5QFA which is working on a tape of n symbols (including the end-markers) has therefore $n|Q|$ different configurations. Initially, the head is on the left end-marker, and so the machine starts in the configuration $(q_0, 0)$. At later steps of the computation, the machine may exist in superpositions of more than one configuration. It is sometimes useful to visualize such superpositions of multiple configurations as snapshots of the machine running in multiple parallel computation paths.

As described above for KWQFA's, each step of the computation consists of a unitary transformation of the current superposition according to the transition function, followed by a measurement to see whether the machine has accepted, rejected, or not halted yet. The probability of each outcome is determined by the amplitudes of the relevant configurations in the present superposition. The observation of accepting (rejecting) configurations cause the accumulated acceptance (rejection) probability to be updated accordingly. The machine continues running from a superposition of the non-halting configurations.

In the *2-way quantum finite automaton* (2QFA) [13] model, the tape head is allowed to go left, as well as staying put and going to the right, and the exposition above regarding 1.5QFA's can be generalized in a straightforward manner to 2QFA's.

3 Main Results

Our first result is a complete characterization of UMM.

Lemma 1. *Any language recognized with cutpoint $\frac{1}{2}$ by a 1PFA with n states can be recognized with cutpoint $\frac{1}{2}$ by a KWQFA with $O(n)$ states.*

Proof. Due to space constraints, we will just give a sketch of the proof. Given a 1PFA $\mathcal{P}$ with state set $\{s_1, s_2, \cdots, s_n\}$ that recognizes language L with cutpoint $\frac{1}{2}$, we first modify it so that it treats the end-marker symbols ¢ and \$ correctly, with the acceptance and rejection probabilities at the end of the computation accumulating in a single accept and a single reject state, named s_{n+1} and s_{n+2}, respectively. We then construct a KWQFA $\mathcal{M}$ with $O(n)$ states that recognizes L with cutpoint $\frac{1}{2}$.

For each original state s_i of $\mathcal{P}$, there will be a corresponding non-halting state q_i of $\mathcal{M}$ $(1 \leq i \leq n)$. $\mathcal{M}$ will have an accepting state named q_{n+1} corresponding to s_{n+1}, and a rejecting state named q_{n+2} corresponding to s_{n+2}. The remaining states of $\mathcal{M}$ are accepting and rejecting states that will be described shortly.

The idea behind the construction is to convert the stochastic transition matrices of $\mathcal{P}$ to unitary transition matrices of $\mathcal{M}$ by adding new rows and columns (corresponding to new halting states), so that the distribution of the probabilities in the state vector of $\mathcal{P}$ is "imitated" by the distribution of the amplitudes of the non-halting states of $\mathcal{M}$: For any position t of the input tape, there will be a positive real number k_t such that the amplitude of q_i just after $\mathcal{M}$ reads the t^{th} symbol equals k_t times the probability of s_i just after $\mathcal{P}$ reads that symbol $(1 \leq i \leq n)$. The newly added halting states will come in accept/reject pairs, so that transitions to them during the computation will add equal amounts to the overall acceptance and rejection probabilities, and therefore will not affect the decision on the membership of the input in L.

For an input string $w \in \Sigma^*$, let $\mathsf{w} = ¢w\$$ denote the corresponding tape content including the end-markers, and let $f_{\mathcal{P}}(w)$ and $f_{\mathcal{M}}(w)$ represent the acceptance probabilities of $\mathcal{P}$ and $\mathcal{M}$, respectively, for w.

After reading symbol \$, the amplitude of q_{n+1} is $k_{|\mathsf{w}|} f_{\mathcal{P}}(w)$, recalling that $f_{\mathcal{P}}(w)$ is summed up in s_{n+1}. The amplitude of q_{n+2} at that point equals $k_{|\mathsf{w}|}(1 - f_{\mathcal{P}}(w))$. The total acceptance probability of $\mathcal{M}$ turns out [25] to be

$$f_{\mathcal{M}}(w) = \frac{1}{2} + \frac{k_{|\mathsf{w}|}^2}{2}(2f_{\mathcal{P}}(w) - 1),$$

where $0 < k_{|\mathsf{w}|} \leq 1$. If $w \in L$, then $f_{\mathcal{P}}(w) > \frac{1}{2}$, and so $f_{\mathcal{M}}(w) > \frac{1}{2}$. If $w \notin L$, then $f_{\mathcal{P}}(w) \leq \frac{1}{2}$, and so $f_{\mathcal{M}}(w) \leq \frac{1}{2}$. □

The standard definitions of PFA's and QFA's that we gave in Section 2 allow arbitrary real numbers as transition amplitudes, and therefore the related language classes include undecidable languages [22]. To rectify this situation, one may restrict the amplitudes to efficiently computable numbers, as in [7]. Our results in this section remain valid in that case as well.

Lemma 2. *Let $\mathcal{M}$ be a KWQFA with n states and $f_{\mathcal{M}} : \Sigma^* \to [0, 1]$ be its acceptance probability function. Then there exists a GPFA $\mathcal{G}$ with $O(n^2)$ states such that $f_{\mathcal{M}}(w) = f_{\mathcal{G}}(w)$ for all $w \in \Sigma^*$.*

Proof. The proof is omitted due to space constraints, see [25]. Note that this result also follows from independent work by Li and Qiu [15]. □

Theorem 3. *UMM equals the class of stochastic languages.*

Proof. Follows from Lemma 1, Lemma 2, and Theorem 2. □

We now show that, unlike classical probabilistic finite automata, allowing the tape head to "stay put" for some steps during its left-to-right traversal of the input increases the language recognition power of quantum finite automata in the unbounded error case.

Theorem 4. *UMM is a proper subset of the class of languages recognized by 1.5QFA's with unbounded error.*

Proof. By the definition of 1.5QFA's and Theorem 3, every stochastic language can be recognized with unbounded error by 1.5QFA's. We will show that the context-free nonstochastic language [18]

$$\mathcal{L}_1 = \{a^x b a^{y_1} b a^{y_2} b \cdots a^{y_t} b \mid x, t, y_1, \cdots, y_t \in \mathbb{Z}^+ \text{ and } \exists k \ (1 \leq k \leq t), x = \sum_{i=1}^{k} y_i\}$$

over the alphabet $\{a, b\}$ can be recognized with unbounded error by a 1.5QFA with cutpoint $\frac{1}{2}$. Consider the 1.5QFA $\mathcal{Q} = (Q, \Sigma, \delta, Q_{acc}, Q_{rej})$, where $\Sigma = \{a, b\}$ and the state sets are as follows:

$$\begin{aligned} Q_{non} = &\{\overrightarrow{q_0}\} \cup \{\overrightarrow{q_i} \mid 1 \leq i \leq 6\} \cup \{\overrightarrow{p_i} \mid 1 \leq i \leq 6\} \cup \{\overrightarrow{a_i} \mid 1 \leq i \leq 4\} \\ &\cup \{\overrightarrow{r_i} \mid 1 \leq i \leq 4\} \cup \{\downarrow w_i \mid 1 \leq i \leq 6\}, \\ Q_{acc} = &\{\downarrow A_i \mid 1 \leq i \leq 10\}, \ Q_{rej} = \{\downarrow R_i \mid 1 \leq i \leq 18\}. \end{aligned}$$

Let each U_σ act as indicated in Figures 1 and 2, and extend each to be unitary. Let δ be related to the U_σ as described in Section 2.

Machine $\mathcal{Q}$ starts computation on symbol ¢ by branching into two paths, path_1 and path_2, with equal probability. Each path and their subpaths, to be described later, check whether the input is of the form $(aa^*b)(aa^*b)(aa^*b)^*$. The different stages of the program indicated in Figures 1 and 2 correspond to the subtasks of this regular expression check. Stage I ends successfully if the input begins with (aa^*b). Stage II checks the second (aa^*b). Finally, Stage III controls whether the input ends with $(aa^*b)^*$.

The reader will note that many transitions in the machine are of the form

$$U_\sigma|q_i\rangle = |\psi\rangle + \alpha|A_k\rangle + \alpha|R_k\rangle,$$

where $|\psi\rangle$ is a superposition of configurations such that $\langle\psi|\psi\rangle = 1 - 2\alpha^2$, $A_k \in Q_{acc}$, $R_k \in Q_{rej}$. The equal-probability transitions to the "twin halting states" A_k and R_k are included to ensure that the matrices are unitary, without upsetting the "accept/reject balance" until a final decision about the membership of the input in $\mathcal{L}_1$ is reached. If the regular expression check mentioned above fails, each path in question terminates in a rejecting configuration, and the overall probability of acceptance of the machine turns out to be less than $\frac{1}{2}$. If the input is indeed of the form $(aa^*b)(aa^*b)(aa^*b)^*$, the acceptance probability is at least $\frac{1}{2}$, and whether $\frac{1}{2}$ will be exceeded or not depends on the following additional tasks performed by the computation paths in order to test for the equality mentioned in the definition of $\mathcal{L}_1$:

Stages	$U_{¢}, U_a$	$U_\$$
	$U_{¢}\vert\overrightarrow{q_0}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{q_1}\rangle + \frac{1}{\sqrt{2}}\vert\overrightarrow{p_1}\rangle$	
I (path$_1$)	$U_a\vert\overrightarrow{q_1}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{q_2}\rangle + \frac{1}{2}\vert{\downarrow}A_1\rangle + \frac{1}{2}\vert{\downarrow}R_1\rangle$ $U_a\vert\overrightarrow{q_2}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{q_2}\rangle - \frac{1}{2}\vert{\downarrow}A_1\rangle - \frac{1}{2}\vert{\downarrow}R_1\rangle$	$U_\$\vert\overrightarrow{q_1}\rangle = \vert{\downarrow}R_3\rangle$ $U_\$\vert\overrightarrow{q_2}\rangle = \vert{\downarrow}R_4\rangle$ $U_\$\vert\overrightarrow{q_3}\rangle = \vert{\downarrow}R_5\rangle$
I (path$_2$)	$U_a\vert\overrightarrow{p_1}\rangle = \vert{\downarrow}w_1\rangle$ $U_a\vert{\downarrow}w_1\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{p_2}\rangle + \frac{1}{2}\vert{\downarrow}A_2\rangle + \frac{1}{2}\vert{\downarrow}R_2\rangle$ $U_a\vert\overrightarrow{p_2}\rangle = \vert{\downarrow}w_2\rangle$ $U_a\vert{\downarrow}w_2\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{p_2}\rangle - \frac{1}{2}\vert{\downarrow}A_2\rangle - \frac{1}{2}\vert{\downarrow}R_2\rangle$	$U_\$\vert\overrightarrow{p_1}\rangle = \vert{\downarrow}R_6\rangle$ $U_\$\vert\overrightarrow{p_2}\rangle = \vert{\downarrow}R_7\rangle$ $U_\$\vert\overrightarrow{p_3}\rangle = \vert{\downarrow}R_8\rangle$
II (path$_1$)	$U_a\vert\overrightarrow{q_3}\rangle = \vert{\downarrow}w_3\rangle$ $U_a\vert{\downarrow}w_3\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{q_4}\rangle + \frac{1}{2}\vert{\downarrow}A_3\rangle + \frac{1}{2}\vert{\downarrow}R_3\rangle$ $U_a\vert\overrightarrow{q_4}\rangle = \vert{\downarrow}w_4\rangle$ $U_a\vert{\downarrow}w_4\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{q_4}\rangle - \frac{1}{2}\vert{\downarrow}A_3\rangle - \frac{1}{2}\vert{\downarrow}R_3\rangle$	$U_\$\vert\overrightarrow{q_4}\rangle = \vert{\downarrow}R_9\rangle$ $U_\$\vert\overrightarrow{q_5}\rangle = \frac{1}{2}\vert{\downarrow}A_1\rangle + \frac{1}{2}\vert{\downarrow}R_1\rangle$
II (path$_2$)	$U_a\vert\overrightarrow{p_3}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{p_4}\rangle + \frac{1}{2}\vert{\downarrow}A_4\rangle + \frac{1}{2}\vert{\downarrow}R_4\rangle$ $U_a\vert\overrightarrow{p_4}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{p_4}\rangle - \frac{1}{2}\vert{\downarrow}A_4\rangle - \frac{1}{2}\vert{\downarrow}R_4\rangle$	$U_\$\vert\overrightarrow{p_4}\rangle = \vert{\downarrow}R_{10}\rangle$ $U_\$\vert\overrightarrow{p_5}\rangle = \frac{1}{2}\vert{\downarrow}A_2\rangle + \frac{1}{2}\vert{\downarrow}R_2\rangle$
III (path$_1$)	$U_a\vert\overrightarrow{q_5}\rangle = \vert{\downarrow}w_5\rangle$ $U_a\vert{\downarrow}w_5\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{q_6}\rangle + \frac{1}{2}\vert{\downarrow}A_5\rangle + \frac{1}{2}\vert{\downarrow}R_5\rangle$ $U_a\vert\overrightarrow{q_6}\rangle = \vert{\downarrow}w_6\rangle$ $U_a\vert{\downarrow}w_6\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{q_6}\rangle - \frac{1}{2}\vert{\downarrow}A_5\rangle - \frac{1}{2}\vert{\downarrow}R_5\rangle$	$U_\$\vert\overrightarrow{q_6}\rangle = \vert{\downarrow}R_{11}\rangle$
III (path$_2$)	$U_a\vert\overrightarrow{p_5}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{p_6}\rangle + \frac{1}{2}\vert{\downarrow}A_6\rangle + \frac{1}{2}\vert{\downarrow}R_6\rangle$ $U_a\vert\overrightarrow{p_6}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{p_6}\rangle - \frac{1}{2}\vert{\downarrow}A_6\rangle - \frac{1}{2}\vert{\downarrow}R_6\rangle$	$U_\$\vert\overrightarrow{p_6}\rangle = \vert{\downarrow}R_{12}\rangle$
III (path$_{\text{accept}}$)	$U_a\vert\overrightarrow{a_1}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{a_2}\rangle + \frac{1}{2}\vert{\downarrow}A_7\rangle + \frac{1}{2}\vert{\downarrow}R_7\rangle$ $U_a\vert\overrightarrow{a_2}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{a_2}\rangle - \frac{1}{2}\vert{\downarrow}A_7\rangle - \frac{1}{2}\vert{\downarrow}R_7\rangle$ $U_a\vert\overrightarrow{a_3}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{a_4}\rangle + \frac{1}{2}\vert{\downarrow}A_8\rangle + \frac{1}{2}\vert{\downarrow}R_8\rangle$ $U_a\vert\overrightarrow{a_4}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{a_4}\rangle - \frac{1}{2}\vert{\downarrow}A_8\rangle - \frac{1}{2}\vert{\downarrow}R_8\rangle$	$U_\$\vert\overrightarrow{a_1}\rangle = \vert{\downarrow}A_3\rangle$ $U_\$\vert\overrightarrow{a_3}\rangle = \vert{\downarrow}A_4\rangle$ $U_\$\vert\overrightarrow{a_2}\rangle = \vert{\downarrow}R_{13}\rangle$ $U_\$\vert\overrightarrow{a_4}\rangle = \vert{\downarrow}R_{14}\rangle$
III (path$_{\text{reject}}$)	$U_a\vert\overrightarrow{r_1}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{r_2}\rangle + \frac{1}{2}\vert{\downarrow}A_9\rangle + \frac{1}{2}\vert{\downarrow}R_9\rangle$ $U_a\vert\overrightarrow{r_2}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{r_2}\rangle - \frac{1}{2}\vert{\downarrow}A_9\rangle - \frac{1}{2}\vert{\downarrow}R_9\rangle$ $U_a\vert\overrightarrow{r_3}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{r_4}\rangle + \frac{1}{2}\vert{\downarrow}A_{10}\rangle + \frac{1}{2}\vert{\downarrow}R_{10}\rangle$ $U_a\vert\overrightarrow{r_4}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{r_4}\rangle - \frac{1}{2}\vert{\downarrow}A_{10}\rangle - \frac{1}{2}\vert{\downarrow}R_{10}\rangle$	$U_\$\vert\overrightarrow{r_1}\rangle = \vert{\downarrow}R_{15}\rangle$ $U_\$\vert\overrightarrow{r_3}\rangle = \vert{\downarrow}R_{16}\rangle$ $U_\$\vert\overrightarrow{r_2}\rangle = \vert{\downarrow}R_{17}\rangle$ $U_\$\vert\overrightarrow{r_4}\rangle = \vert{\downarrow}R_{18}\rangle$

Fig. 1. Specification of the transition function of $\mathcal{Q}$ (part 1)

1. path$_1$ walks over the a's at the speed of one tape square per step until reading the first b. After that point, path$_1$ pauses for one step over each a before moving on to the next symbol.
2. path$_2$ pauses for one step over each a until reading the first b. After that point, path$_2$ walks over each a at the speed of one square per step.
3. On each b except the first one, path$_1$ and path$_2$ split to take the following two courses of action with equal probability:
   (a) In the first alternative, path$_1$ and path$_2$ perform a two-way quantum Fourier transform (QFT) [13]:
      i. The targets of the QFT are two new computational paths, i.e.,path$_{\text{accept}}$ and path$_{\text{reject}}$. Disregarding the equal-probability transitions to the

Stages	U_b
I (path_1)	$U_b\vert\overrightarrow{q_1}\rangle = \vert{\downarrow}R_5\rangle$ $U_b\vert\overrightarrow{q_2}\rangle = \vert\overrightarrow{q_3}\rangle$ $U_b\vert\overrightarrow{q_3}\rangle = \vert{\downarrow}R_6\rangle$
I (path_2)	$U_b\vert\overrightarrow{p_1}\rangle = \vert{\downarrow}R_7\rangle$ $U_b\vert\overrightarrow{p_2}\rangle = \vert\overrightarrow{p_3}\rangle$ $U_b\vert\overrightarrow{p_3}\rangle = \vert{\downarrow}R_8\rangle$
II (path_1)	$U_b\vert\overrightarrow{q_4}\rangle = \frac{1}{2}\vert\overrightarrow{q_5}\rangle + \frac{1}{2\sqrt{2}}\vert\overrightarrow{a_1}\rangle + \frac{1}{2\sqrt{2}}\vert\overrightarrow{r_1}\rangle + \frac{1}{2}\vert{\downarrow}A_1\rangle + \frac{1}{2}\vert{\downarrow}R_1\rangle$ $U_b\vert\overrightarrow{q_5}\rangle = \vert{\downarrow}R_9\rangle$
II (path_2)	$U_b\vert\overrightarrow{p_4}\rangle = \frac{1}{2}\vert\overrightarrow{p_5}\rangle + \frac{1}{2\sqrt{2}}\vert\overrightarrow{a_1}\rangle - \frac{1}{2\sqrt{2}}\vert\overrightarrow{r_1}\rangle + \frac{1}{2}\vert{\downarrow}A_2\rangle + \frac{1}{2}\vert{\downarrow}R_2\rangle$ $U_b\vert\overrightarrow{p_5}\rangle = \vert{\downarrow}R_{10}\rangle$
III (path_1)	$U_b\vert\overrightarrow{q_6}\rangle = \frac{1}{2}\vert\overrightarrow{q_5}\rangle + \frac{1}{2\sqrt{2}}\vert\overrightarrow{a_1}\rangle + \frac{1}{2\sqrt{2}}\vert\overrightarrow{r_1}\rangle - \frac{1}{2}\vert{\downarrow}A_1\rangle - \frac{1}{2}\vert{\downarrow}R_1\rangle$
III (path_2)	$U_b\vert\overrightarrow{p_6}\rangle = \frac{1}{2}\vert\overrightarrow{p_5}\rangle + \frac{1}{2\sqrt{2}}\vert\overrightarrow{a_1}\rangle - \frac{1}{2\sqrt{2}}\vert\overrightarrow{r_1}\rangle - \frac{1}{2}\vert{\downarrow}A_2\rangle - \frac{1}{2}\vert{\downarrow}R_2\rangle$
III ($\mathsf{path}_{\mathsf{accept}}$)	$U_b\vert\overrightarrow{a_2}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{a_3}\rangle + \frac{1}{2}\vert{\downarrow}A_3\rangle + \frac{1}{2}\vert{\downarrow}R_3\rangle$ $U_b\vert\overrightarrow{a_1}\rangle = \vert{\downarrow}R_{11}\rangle$ $U_b\vert\overrightarrow{a_4}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{a_3}\rangle - \frac{1}{2}\vert{\downarrow}A_3\rangle - \frac{1}{2}\vert{\downarrow}R_3\rangle$ $U_b\vert\overrightarrow{a_3}\rangle = \vert{\downarrow}R_{12}\rangle$
III ($\mathsf{path}_{\mathsf{reject}}$)	$U_b\vert\overrightarrow{r_2}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{r_3}\rangle + \frac{1}{2}\vert{\downarrow}A_4\rangle + \frac{1}{2}\vert{\downarrow}R_4\rangle$ $U_b\vert\overrightarrow{r_1}\rangle = \vert{\downarrow}R_{13}\rangle$ $U_b\vert\overrightarrow{r_4}\rangle = \frac{1}{\sqrt{2}}\vert\overrightarrow{r_3}\rangle - \frac{1}{2}\vert{\downarrow}A_4\rangle - \frac{1}{2}\vert{\downarrow}R_4\rangle$ $U_b\vert\overrightarrow{r_3}\rangle = \vert{\downarrow}R_{14}\rangle$

Fig. 2. Specification of the transition function of $\mathcal{Q}$ (part 2)

twin halting states mentioned above, the QFT is realized as:

$$\mathsf{path}_1 \rightarrow \frac{1}{\sqrt{2}}\mathsf{path}_{\mathsf{accept}} + \frac{1}{\sqrt{2}}\mathsf{path}_{\mathsf{reject}}$$

$$\mathsf{path}_2 \rightarrow \frac{1}{\sqrt{2}}\mathsf{path}_{\mathsf{accept}} - \frac{1}{\sqrt{2}}\mathsf{path}_{\mathsf{reject}}$$

ii. $\mathsf{path}_{\mathsf{accept}}$ and $\mathsf{path}_{\mathsf{reject}}$ continue computation at the speed of path_2, walking over the b's without performing the QFT any more.

(b) In the second alternative, path_1 and path_2 continue computation without performing the QFT.

4. On symbol \$, $\mathsf{path}_{\mathsf{accept}}$ enters an accepting state, $\mathsf{path}_{\mathsf{reject}}$ enters a rejecting state, path_1 and path_2 enter accepting and rejecting states with equal probability.

Suppose that the input is of the form

$$w = a^x b a^{y_1} b a^{y_2} b \cdots a^{y_t} b,$$

where $x, t, y_1, \cdots, y_t \in \mathbb{Z}^+$.

path_1 reaches the first b earlier than path_2. Once it has passed the first b, path_2 becomes faster, and may or may not catch up with path_1, depending on the number of a's in the input after the first b. The two paths can meet on the symbol following the x'th a after the first b, since at that point path_1 will have paused for the same number of steps as path_2. Only if that symbol is a b, the two paths will perform a QFT in the same place and at the same time. To paraphrase, if there exists a k $(1 \leq k \leq t)$ such that $x = \sum_{i=1}^{k} y_i$, path_1 and path_2 meet over the $(k+1)^{th}$ b and perform the QFT at the same step. If there is no such k, the paths either never meet, or meet over an a without a QFT.

The $\mathsf{path}_{\mathsf{accept}}$ and $\mathsf{path}_{\mathsf{reject}}$s that are offshoots of path_1 continue their traversal of the string faster than path_1. On the other hand, the offshoots of path_2 continue their traversal at the same speed as path_2.

By definition, the twin halting states reached during the computation contribute equal amounts to the acceptance and rejection probabilities. path_1 and path_2 accept and reject equiprobably when they reach the end of the string. If path_1 and path_2 never perform the QFT at the same time and in the same position, every QFT produces two equal-probability paths which perform identical tasks, except that one accepts and the other one rejects at the end.

The overall acceptance and rejection probabilities are equal, $\frac{1}{2}$, unless a $\mathsf{path}_{\mathsf{reject}}$ with positive amplitude and a $\mathsf{path}_{\mathsf{reject}}$ with negative amplitude can meet and therefore cancel each other. In such a case, the surviving $\mathsf{path}_{\mathsf{accept}}$'s will contribute the additional acceptance probability that will tip the balance. As described above, such a cancellation is only possible when path_1 and path_2 perform the QFT together.

Therefore, if $w \in \mathcal{L}_1$, the overall acceptance probability is greater than $\frac{1}{2}$. If $w \notin \mathcal{L}_1$, the overall acceptance probability does not exceed $\frac{1}{2}$. □

4 Concluding Remarks

In this paper, we examined the capabilities of measure-many (Kondacs-Watrous) quantum finite automata in the unbounded error setting. We gave a full characterization of the class of languages recognized by one-way QFA's of this kind; they turn out to recognize all and only the stochastic languages. We also showed that allowing the tape head to "stay put" for some steps during its left-to-right traversal of the input increases the language recognition power of quantum finite automata in the unbounded error case. This means that two-way (and even "1.5-way") QFA's are strictly more powerful than 1QFA's; whereas 2PFA's are known to be equivalent in power to 1PFA's [12] in this setting.

The relationship between the computational powers of quantum Turing machines and probabilistic Turing machines with unbounded error were examined by Adleman, DeMarrais and Huang [2] and Ablayev and Gainutdinova [1] in the context of time bounds, and by Watrous [24] for space bounds. Watrous showed that, for s satisfying $s(n) = \Omega(\log n)$, space $O(s)$ bounded QTM's and PTM's are equivalent in power, and left the question of comparing these models for $s = o(\log n)$ open. Since two-way finite automata with suitable restrictions (which do not affect our constructions) on the transition amplitudes are equivalent to Turing machines restricted to constant space bounds, our results provide an answer to this question; QTM's are strictly more powerful than PTM's in this case[2].

Theorem 3 has some obvious ramifications about upper bounds on the numbers of states of KWQFA's that recognize regular languages. All the well-known results [16, 21] about the "state economy" provided by NFA's and PFA's over DFA's carry over easily to KWQFA's. Lemma 1 can be specialized to show that an n-state DFA can be simulated by an unbounded-error KWQFA with $O(n)$ states, and cut-point 0.

The class of languages recognized with unbounded error by measure-once QFA's [17] is known [8] to be a proper subset of the stochastic languages. In work subsequent to the completion of this paper [26], we were able to show that the related classes for several one-way QFA variants ([9, 19, 20], and the one-way version of the machines of [6],) that are at least as general as the KWQFA equal UMM as well. One important model for which no such characterization is yet known is the Latvian QFA [4]. Another question left open in this work is the relationship between the computational powers of 1.5QFA's and 2QFA's.

Acknowledgement

We thank Rūsiņš Freivalds for his helpful answers to our questions regarding nonstochastic languages.

References

1. Ablayev, F.M., Gainutdinova, A.: Classical simulation complexity of quantum machines. In: Lingas, A., Nilsson, B.J. (eds.) FCT 2003. LNCS, vol. 2751, pp. 296–302. Springer, Heidelberg (2003)
2. Adleman, L.M., DeMarrais, J., Huang, M.-D.A.: Quantum computability. SIAM Journal on Computing 26(5), 1524–1540 (1997)
3. Amano, M., Iwama, K.: Undecidability on quantum finite automata. In: STOC 1999: Proceedings of the thirty-first annual ACM symposium on Theory of computing, pp. 368–375. ACM, New York (1999)
4. Ambainis, A., Beaudry, M., Golovkins, M., Ķikusts, A., Mercer, M., Thérien, D.: Algebraic results on quantum automata. Theory of Computing Systems 39(1), 165–188 (2006)
5. Ambainis, A., Freivalds, R.: 1-way quantum finite automata: strengths, weaknesses and generalizations. In: FOCS 1998: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, Palo Alto, California, pp. 332–341 (1998)
6. Ambainis, A., Watrous, J.: Two–way finite automata with quantum and classical states. Theoretical Computer Science 287(1), 299–311 (2002)
7. Bernstein, E., Vazirani, U.: Quantum complexity theory. SIAM Journal on Computing 26(5) (1997)

[2] Note that Freivalds and Karpinski have shown [11] that no PTM using space $o(\log\log n)$ can recognize the language $\mathcal{L}_1$ of Theorem 4 with unbounded error.

8. Bertoni, A., Carpentieri, M.: Analogies and differences between quantum and stochastic automata. Theoretical Computer Science 262(1-2), 69–81 (2001)
9. Bertoni, A., Mereghetti, C., Palano, B.: Quantum computing: 1-way quantum automata. In: Ésik, Z., Fülöp, Z. (eds.) DLT 2003. LNCS, vol. 2710, pp. 1–20. Springer, Heidelberg (2003)
10. Brodsky, A., Pippenger, N.: Characterizations of 1–way quantum finite automata. SIAM Journal on Computing 31(5), 1456–1478 (2002)
11. Freivalds, R., Karpinski, M.: Lower space bounds for randomized computation. In: Shamir, E., Abiteboul, S. (eds.) ICALP 1994. LNCS, vol. 820, pp. 580–592. Springer, Heidelberg (1994)
12. Kaņeps, J.: Stochasticity of the languages acceptable by two-way finite probabilistic automata. Diskretnaya Matematika 1, 63–67 (1989) (in Russian)
13. Kondacs, A., Watrous, J.: On the power of quantum finite state automata. In: FOCS 1997: Proceedings of the 38th Annual Symposium on Foundations of Computer Science, Miami, Florida, pp. 66–75 (1997)
14. Kuklin, Y.I.: Two-way probabilistic automata. Avtomatika i vyčistitelnaja tekhnika 5, 36 (1973) (in Russian)
15. Li, L., Qiu, D.: Determining the equivalence for one-way quantum finite automata. Theoretical Computer Science 403(1), 42–51 (2008)
16. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: 12th Annual Symposium on Switching and Automata Theory, East Lansing, MI, USA, pp. 188–191 (1971)
17. Moore, C., Crutchfield, J.P.: Quantum automata and quantum grammars. Theoretical Computer Science 237(1-2), 275–306 (2000)
18. Nasu, M., Honda, N.: A context-free language which is not acceptable by a probabilistic automaton. Information and Control 18(3), 233–236 (1971)
19. Nayak, A.: Optimal lower bounds for quantum automata and random access codes. In: FOCS 1999: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, Washington, DC, USA, pp. 369–376. IEEE Computer Society, Los Alamitos (1999)
20. Paschen, K.: Quantum finite automata using ancilla qubits. Technical report, University of Karlsruhe (2000)
21. Paz, A.: Introduction to Probabilistic Automata. Academic Press, New York (1971)
22. Rabin, M.O.: Probabilistic automata. Information and Control 6, 230–243 (1963)
23. Turakainen, P.: Generalized automata and stochastic languages. Proceedings of the American Mathematical Society 21, 303–309 (1969)
24. Watrous, J.: Space-bounded quantum complexity. Journal of Computer and System Sciences 59(2), 281–326 (1999)
25. Yakaryılmaz, A., Cem Say, A.C.: Languages recognized with unbounded error by quantum finite automata. Technical report (2008) arXiv:0809.0073v2
26. Yakaryılmaz, A., Cem Say, A.C.: Language recognition by generalized quantum finite automata with unbounded error. In: 4th Workshop on Theory of Quantum Computation, Communication, and Cryptography, TQC 2009, Waterloo, Ontario, Canada (2009) (forthcoming)

Author Index