Volker Diekert
Dirk Nowotka (Eds.)

LNCS 5583

Developments in Language Theory

13th International Conference, DLT 2009
Stuttgart, Germany, June/July 2009
Proceedings

Springer

Lecture Notes in Computer Science 5583

Volker Diekert Dirk Nowotka (Eds.)

Developments in Language Theory

13th International Conference, DLT 2009
Stuttgart, Germany, June 30-July 3, 2009
Proceedings

Springer

Volume Editors

Volker Diekert
Dirk Nowotka
Universität Stuttgart
Institut für Formale Methoden der Informatik, FMI
Universitätsstrasse 38, 70569 Stuttgart, Germany
E-mail: {diekert,nowotka}@fmi.uni-stuttgart.de

Library of Congress Control Number: 2009929049

CR Subject Classification (1998): F.4.3, F.4.2, F.4, F.3, G.2, F.1, I.2.4, F.2

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

ISSN 0302-9743
ISBN-10 3-642-02736-9 Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02736-9 Springer Berlin Heidelberg New York

springer.com

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12702060 06/3180 5 4 3 2 1 0

Preface

Since 1993 the conference Developments in Language Theory (DLT) has been held in Europe every odd year and, since 2002, outside Europe every even year. The 13th conference in this series was DLT 2009. It took place in Stuttgart from June 30 to July 3. Previous meetings occurred in Turku (1993), Magdeburg (1995), Thessaloniki (1997), Aachen (1999), Vienna (2001), Kyoto (2002), Szeged (2003), Auckland (2004), Palermo (2005), Santa Barbara (2006), Turku (2007), and Kyoto (2008).

The DLT conference has developed into the main forum for language theory and related topics. This has also been reflected in the high quality of the 70 submissions received in 2009. Most submissions were reviewed by four Programme Committee members and their sub-referees. The Programme Committee selected the best 35 papers for presentation during the conference. These 35 papers are also published in this proceedings volume. Members of the Programme Committee were not allowed to submit papers. The work of the Programme Committee was organized using the EasyChair conference system, thanks to Andrei Voronkov.

The conference programme included five invited lectures. They were given by Mikołaj Bojańczyk (Warsaw), Paul Gastin (Cachan), Tero Harju (Turku), Christos Kapoutsis (Nicosia), and Benjamin Steinberg (Ottawa). We are grateful to the invited speakers for accepting the invitation and presenting their lectures and for their contributions to the proceedings.

The Informatik Forum Stuttgart provided a best paper award, which was selected by the Programme Committee. The recipient was:

“Magic Numbers and Ternary Alphabet” by Galina Jiraskova.

The DLT conference was accompanied by the following satellite events:

- Workshop on “Automata and Algorithmic Logic”
- Festkolloquium on the occasion of the “65th birthday of Volker Claus”

The editors thank the authors of all submitted papers for considering DLT 2009 as an appropriate platform for presenting their work. Moreover, we would like to thank the members of the Programme Committee for their professional work in carefully reading and evaluating the submissions and selecting the best contributions. We also thank all members of the Institute for Formal Methods in Computer Science for their help. In particular, we thank Heike Photien and Horst Prote for their great efforts.

Financial support for this conference was provided by the German research foundation DFG and by the Universität Stuttgart. The association Informatik Forum Stuttgart (infos e.V.) helped in the management of the conference and donated the Infos Best Paper Award for DLT 2009. The assistance of these societies is greatly acknowledged.

May 2009

Volker Diekert
Dirk Nowotka

Conference Organization

Conference Chair

Volker Diekert

Programme Committee

Volker Diekert (Chair), Stuttgart
Zoltán Ésik, Tarragona
Massimiliano Goldwurm, Milan
Hendrik Jan Hoogeboom, Leiden
Juraj Hromkovič, Zurich
Masami Ito, Kyoto
Juhani Karhumäki, Turku
Bakhadyr Khoussainov, Auckland
Dalia Krieger, Rehovot
Dietrich Kuske, Leipzig
Klaus-Jörn Lange, Tübingen
Giancarlo Mauri, Milan
Mehryar Mohri, New York
Friedrich Otto, Kassel
Libor Polak, Brno
Gwénaël Richomme, Amiens
Jacques Sakarovitch, Paris
Géraud Sénizergues, Bordeaux
Mikhail Volkov, Jekaerinburg
Sheng Yu, London, Ontario

Local Organization

Volker Claus
Volker Diekert (Chair)
Ulrich Hertrampf
Michael Matthiesen
Dirk Nowotka

External Reviewers

Allauzen, Cyril
Allouche, Jean-Paul
Ananichev, Dmitry
Angrand, Pierre-Yves
Aszalós, László
Breveglieri, Luca
Bárány, Vince
Bertoni, Alberto
Berwanger, Dietmar
Biegler, Franziska
Blanchet-Sadri, Francine
Bloom, Stephen L.
Boasson, Luc
Boldi, Paolo
Bordihn, Henning
Bouyer, Patricia
Broersen, Jan
Bugeaud, Yann
Calude, Cristian
Carayol, Arnaud
Carpi, Arturo
Caucal, Didier
Cherubini, Alessandra
Choffrut, Christian
Crespi Reghizzi, Stefano
Currie, James

Czeizler, Eugen
D'Alessandro, Flavio
Dassow, Jürgen
Demri, Stéphane
Dennunzio, Alberto
Dömösi, Pál
Egorov, Pavel
Ferretti, Claudio
Fagnot, Isabelle
Fernau, Henning
Fleury, Emmanuel
Gaál, Tamás
Gamzova, Julia
Gao, Yuan
Glen, Amy
Grossi, Giuliano
Halava, Vesa
Harju, Tero
Hertrampf, Ulrich
Hirvensalo, Mika
Hoffmann, Benjamin
Holub, Štěpán
Holzer, Markus
Honkala, Juha
Iván, Szabolcs
Jeż, Artur
Kari, Jarkko
Kärki, Tomi
Kirsten, Daniel
Klíma, Ondřej
Kobayashi, Yuji
Kobele, Gregory
Kontorovich, Aryeh
Kortelainen, Juha
Kudlek, Manfred
Kufleitner, Manfred
Kuhlmann, Marco
Kuich, Werner
Kunc, Michal
Kunimochi, Yoshiyuki
Kupke, Clemens
Kutrib, Martin
Laine, Markku
Lampert, Robby
Laun, Jürn
Leroux, Jérôme
Leucker, Martin
Leupold, Peter
Levé, Florence
Lohrey, Markus
Lonati, Violetta
Ly, Olivier
Maletti, Andreas
Mantaci, Sabrina
Margenstern, Maurice
Markey, Nicolas
Matz, Oliver
Meduna, Alexander
Mercas, Robert
Mereghetti, Carlo
Messerschmidt, Hartmut
Mitrana, Victor
Morin, Rémi
Nederhof, Mark-Jan
Nowotka, Dirk
Okhotin, Alexander
Paun, Andrei
Petersen, Holger
Pighizzini, Giovanni
Pradella, Matteo
Radicioni, Roberto
Rahonis, George
Rampersad, Narad
Reidenbach, Daniel
Restivo, Antonio
Riley, Michael
Robson, John Michael
Saarela, Aleksi
Salmela, Petri
Salvati, Sylvain
Santean, Nicolae
Sciortino, Marinella
Serre, Olivier
Shallit, Jeffrey
Silva, Pedro V.
Srba, Jiri
Stamer, Heiko
Sutner, Klaus
Séébold, Patrice
Tesson, Pascal

Torelli, Mauro
Vágvölgyi, Sándor
Vogler, Heiko
Vollmer, Heribert
Wagner, Klaus
Walukiewicz, Igor
Yunès, Jean-Baptiste
Zandron, Claudio
Zhukov, Anton
Zizza, Rosalba

Table of Contents

Invited Talks

Regular Papers

Factorization Forests

Mikołaj Bojańczyk

Warsaw University

Abstract. A survey of applications of factorization forests.

Fix a regular language $L \subseteq A^*$. You are given a word $a_1 \cdots a_n \in A^*$. You are allowed to build a data structure in time $O(n)$. Then, you should be able to quickly answer queries of the form: given $i \leq j \in \{1, \ldots, n\}$, does the infix $a_i \cdots a_j$ belong to L?

What should the data structure be? What does quickly mean? There is natural solution that uses a divide and conquer approach. Suppose that the language L is recognized by a (nondeterministic) automaton with states Q. We can divide the word in two halves, then into quarters and so on. The result is a binary tree decomposition, where each tree node corresponds to an infix, and its children divide the infix into two halves. In a bottom-up pass we decorate each node of the tree with the set $R \subseteq Q^2$ of pairs (source, target) for runs over node's corresponding infix. This data structure can be computed in time linear in the length of the word. Since the height of this tree is logarithmic, a logarithmic number of steps is sufficient to compute the set R of any infix (and the value of R determines membership in L).

The goal of this paper is to popularize a remarkable combinatorial result of Imre Simon [18]. One of its applications is that the data structure above can be modified so that the queries are answered not in logarithmic time, but constant time (the constant is the size of a semigroup recognizing the language).

So, what is the Simon theorem? Let $\alpha : A^* \to S$ be a morphism into a finite monoid[1]. Recall the tree decomposition mentioned in the logarithmic divide and conquer algorithm. This tree decomposes the word using a single rule, which we call the *binary rule*: each word $w \in A^*$ can be split into two factors $w = w_1 \cdot w_2$, with $w_1, w_2 \in A^*$. Since the rule is binary, we need trees of at least logarithmic height (it is a good strategy to choose w_1 and w_2 of approximately same length). To go down to constant height, we need a rule that splits a word into an unbounded number of factors. This is the *idempotent rule*: a word w can be factorized as $w = w_1 \cdot w_2 \cdots w_k$, as long as the images of the factors $w_1, \ldots, w_k \in A^*$ are all equal, and furthermore idempotent:

$$\alpha(w_1) = \cdots = \alpha(w_k) = e \qquad \text{for some } e \in S \text{ with } ee = e.$$

[1] Recall that a monoid is a set with an associative multiplication operation, and an identity element. A morphism is a function between monoids that preserves the operation and identity.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 1–17, 2009.

An *α-factorization forest* for a word $w \in A^*$ is an unranked tree, where each leaf is labelled by a single letter or the empty word, each non-leaf node corresponds to either a binary or idempotent rule, and the rule in the root gives w.

Theorem 1 (Factorization Forest Theorem of Simon [18]). *For every morphism $\alpha : A^* \to S$ there is a bound $K \in \mathbb{N}$ such that all words $w \in A^*$ have an α-factorization forest of height at most K.*

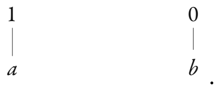
Here is a short way of stating Theorem 1. Let X_i be the set of words that have an α-factorization forest of height i. These sets can be written as

$$X_1 = A \cup \{\epsilon\} \qquad X_{n+1} = X_n \cdot X_n \cup \bigcup_{\substack{e \in S \\ ee=e}} (X_n \cap \alpha^{-1}(e))^* \ .$$

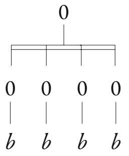
The theorem says that the chain $X_1 \subseteq X_2 \subseteq \cdots$ stabilizes at some finite level.

Let us illustrate the theorem on an example. Consider the morphism $\alpha : \{a, b\}^* \to \{0, 1\}$ that assigns 0 to words without an a and 1 to words with an a. We will use the name *type of w* for the image $\alpha(w)$. We will show how that any word has an α-factorization forest of height five.

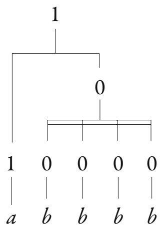
Consider first the single letter words a and b. These have α-factorization forests of height one (the node is decorated with the value under α):



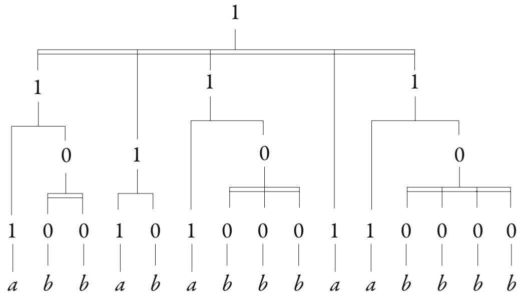
Next, consider words in b^+. These have α-factorization forests of height two: one level is for the single letters, and the second level applies the idempotent rule, which is legal, since the type 0 of b is idempotent:



In the picture above, we used a double line to indicate the idempotent rule. The binary rule is indicated by a single line, as in the following example:



As the picture above indicates, any word in ab^+ has an α-factorization forest of height three. Since the word a needs height at most one, we conclude that all words in ab^* need height at most three. Since the type of ab^+ is the idempotent 1, we can apply the idempotent rule to get a height at most four α-factorization forest for any word in $(ab^*)^*$:



This way, we have covered all words in $\{a, b\}^*$, except for words in $b^+(ab^*)^+$. For these, first use the height four factorization forest for the part $(ab^*)^+$, and then attach the prefix b^+ using the binary rule.

A relaxed idempotent rule. Recall that the idempotent rule requires the word w to be split into parts $w = w_1 \cdots w_k$ with the same idempotent type. What if we relaxed this rule, by only requiring all the parts to have the same type, but not necessarily an idempotent type? We claim that relaxing the idempotent rule would not make the Factorization Forest Theorem any simpler. The reason is that in any finite monoid S, there is some power $m \in \mathbb{N}$ such s^m is idempotent for any $s \in S$. Therefore, any application of the relaxed rule can be converted into a height $\log m$ tree with one idempotent rule, and a number of binary rules.

1 Proof of the Theorem

This section contains a proof of the Factorization Forest Theorem, based on a proof by Manfred Kufleitner [12], with modifications suggested by Szymon Toruńczyk. The proof is self-contained. Implicitly it uses Green's relations, but these are not explicitly named.

We define the *Simon height* $\|S\|$ of a finite monoid S to be the smallest number K such that for every morphism $\alpha : A^* \to S$, all words in A^* have an α-factorization forest of height at most K. Our goal is to show that $\|S\|$ is finite for a finite monoid S. The proof is by induction on the number of elements in S. The induction base, when S has one element, is obvious, so the rest of the proof is devoted to the induction step.

Each element $s \in S$ generates three ideals: the left ideal Ss, the right ideal sS and the two-sided ideal SsS. All of these are submonoids and contain s. Elements of S are called *$\mathcal{H}$-equivalent* if they have the same left and right ideals. First, we show a lemma, which bounds the height $\|S\|$ based on a morphism $\beta : S \to T$.

We use this lemma to reduce the problem to monoids where there is at most one nonzero two-sided ideal (nonzero ideals are defined later). Then we use the lemma to further reduce the problem to monoids where $\mathcal{H}$-equivalence is trivial, either because all elements are equivalent, or because all distinct elements are nonequivalent. Finally, we consider the latter two cases separately.

Lemma 1. *Let S, T be finite monoids and let $\beta : S \to T$ be a morphism.*

$$\|S\| \leq \|T\| \cdot \max_{\substack{e \in T \\ ee=e}} \|\beta^{-1}(e)\|$$

Proof
Let $\alpha : A^* \to S$ be morphism, and $w \in A^*$ a word. We want to find an α-factorization forest of height bounded by the expression in the lemma. We first find a $(\beta \circ \alpha)$-factorization forest f for w, of height bounded by $\|T\|$. Why is f not an α-factorization? The reason is that f might use the idempotent rule to split a word u into factors $u_1, \ldots, u_n$. The factors have the same (idempotent) image under $\beta \circ \alpha$, say $e \in T$, but they might have different images under α. However, all the images under α belong to the submonoid $\beta^{-1}(e)$. Treating the words $u_1, \ldots, u_n$ as single letters, we can find an α-factorization for $u_1 \cdots u_n$ that has height $\|\beta^{-1}(e)\|$. We use this factorization instead of the idempotent rule $u = u_1 \cdots u_n$. Summing up, we replace each idempotent rule in the factorization forest f by a new factorization forest of height $\|\beta^{-1}(e)\|$. □

For an element $s \in S$, consider the two-sided ideal SsS. The equivalence relation $\sim_s$, which collapses all elements from SsS into a single element, is a monoid congruence. Therefore, mapping an element $t \in S$ to its equivalence class under $\sim_s$ is a monoid morphism β, and we can apply Lemma 1 to get

$$\|S\| \leq \|S_{/\sim_s}\| \cdot \|SsS\| \ .$$

When can we use the induction assumption? In other words, when does this inequality above use smaller monoids on the right side? This happens when SsS has at least two elements, but is not all of S. Therefore, it remains to consider the case when for each s, the two-sided ideal SsS is either S or has either one element s. This case is treated below.

At most one nonzero two-sided ideal. From now on, we assume that all two-sided ideals are either S or contain a single element. Note that if $SsS = \{s\}$ then s is a zero, i.e. satisfies $st = ts = s$ for all $t \in S$. There is at most one zero, which we denote by 0. Therefore a two-sided ideal is either S or $\{0\}$.

Note that multiplying on the right either decreases or preserves the right ideal, i.e. $stS \subseteq sS$. We first show that the right ideal cannot be decreased without decreasing the two-sided ideal.

$$\text{if } SsS = SstS \qquad \text{then} \qquad sS = stS \tag{1}$$

Indeed, if the two-sided ideals of s and st are equal, then there are $x, y \in S$ with $s = xsty$. By applying this n times, we get $s = x^n s(ty)^n$. If n is chosen so that $(ty)^n$ is idempotent, which is always possible in a finite monoid, we get

$$s = x^n s(ty)^n = x^n s(ty)^n (ty)^n = s(ty)^n,$$

which gives $sS \subseteq stS$, and therefore $sS = stS$.

We now use (1) to show that $\mathcal{H}$-equivalence is a congruence. In other words, we want to show that if s, u are in $\mathcal{H}$-equivalent, then for any $t \in S$, the elements st, ut are $\mathcal{H}$-equivalent and the elements ts, tu are $\mathcal{H}$-equivalent. By symmetry, we only need to show that st, ut are $\mathcal{H}$-equivalent. The left ideals Sst, Sut are equal by assumption on $Ss = Su$, so it remains to prove equality of the right ideals stS, utS. The two-sided ideal $SstS = SutS$ can be either $\{0\}$ or S. In the first case, $st = ut = 0$. In the second case, $SsS = SstS$, and therefore $sS = stS$ by (1). By the same reasoning, we get $uS = utS$, and therefore $utS = stS$.

Since $\mathcal{H}$-equivalence is a congruence, mapping an element to its $\mathcal{H}$-class (i.e. its $\mathcal{H}$-equivalence class) is a morphism β. The target of β is the quotient of S under $\mathcal{H}$-equivalence, and the inverse images $\beta^{-1}(e)$ are $\mathcal{H}$-classes. By Lemma 1,

$$\|S\| \leq \|S_{/\mathcal{H}}\| \cdot \max_{\substack{s \in S \\ \beta(ss)=\beta(s)}} \|[s]_{\mathcal{H}}\|.$$

We can use the induction assumption on smaller monoids, unless: a) there is one $\mathcal{H}$-class; or b) all $\mathcal{H}$-classes have one element. These two cases are treated below.

All $\mathcal{H}$-classes have one element. Take a morphism $\alpha : A^* \to S$. For $w \in A^*$, we will find an α-factorization forest of size bounded by S. We use the name *type of* w for the image $\alpha(w)$. Consider a word $w \in A^*$. Let v be the longest prefix of w with a type other than 0 and let va be the next prefix of w after v (it may be the case that $v = w$, for instance when there is no zero, so va might not be defined). We cut off the prefix va and repeat the process. This way, we decompose the word w as

$$w = v_1 a_1 v_2 a_2 \cdots v_n a_n v_{n+1} \qquad \begin{array}{l} v_1, \ldots, v_{n+1} \in A^*, a_1 \ldots, a_n \in A \\ \alpha(v_1), \ldots \alpha(v_{n+1}) \neq 0 \quad \alpha(v_1 a_1), \ldots, \alpha(v_n a_n) = 0. \end{array}$$

The factorization forests for $v_1, \ldots, v_{n+1}$ can be combined, increasing the height by three, to a factorization forest for w. (The binary rule is used to append a_i to v_i, the idempotent rule is used to combine the words $v_1 a_1, \ldots, v_n a_n$, and then the binary rule is used to append v_{n+1}.) How do we find a factorization forest for a word v_i? We produce a factorization forest for each v_i by induction on how many distinct infixes $ab \in A^2$ appear in v_i (possibly $a = b$). Since we do not want the size of the alphabet to play a role, we treat ab and cd the same way if the left ideals (of the types of) of a and c are the same, and the right ideals of b and d are the same. What is the type of an infix of v_i? Since we have ruled out 0, then we can use (1) to show that the right ideal of the first letter determines the right ideal of the word, and the left ideal of the last letter determines the left ideal of the word. Since all $\mathcal{H}$-classes have one element, the left and right ideals determine the type. Therefore, the type of an infix of v_i is determined by its first and last letters (actually, their right and left ideals, respectively). Consider all appearances of a two-letter word ab inside v_i:

$$v_i = u_0 ab u_1 ab \cdots ab u_{m+1}$$

By induction, we have factorization forests for $u_0, \ldots, u_{m+1}$. These can be combined, increasing the height by at most three, to a single forest for v_i, because the types of the infixes $bu_1a, \ldots, bu_ma$ are idempotent (unless $m = 1$, in which case the idempotent rule is not needed).

There is one $\mathcal{H}$-class.[2] Take a morphism $\alpha : A^* \to S$. For a word $w \in A^*$ we define $P_w \subseteq S$ to be the types of its non-trivial prefixes, i.e. prefixes that are neither the empty word or w. We will show that a word w has an α-factorization forest of height linear in the size of P_w. The induction base, $P_w = \emptyset$, is simple: the word w has at most one letter. For the induction step, let s be some type in P_w, and choose a decomposition $w = w_0 \cdots w_{n+1}$ such that the only prefixes of w with type s are $w_0, w_0w_1, \ldots, w_0 \cdots w_n$. In particular,

$$P_{w_0},\ s \cdot P_{w_1},\ s \cdot P_{w_2},\ \ldots\ , s \cdot P_{w_n} \quad \subseteq \quad P_w \setminus \{s\}\ .$$

Since there is one $\mathcal{H}$-class, we have $sS = S$. By finiteness of S, the mapping $t \mapsto st$ is a permutation, and therefore the sets sP_{w_i} have fewer elements than P_w. Using the induction assumption, we get factorizations for the words $w_0, \ldots, w_{n+1}$. How do we combine these factorizations to get a factorization for w? If $n = 0$, we use the binary rule. Otherwise, we observe the types of $w_1, \ldots, w_n$ are all equal, since they satisfy $s \cdot \alpha(w_i) = s$, and $t \mapsto st$ is a permutation. For the same reason, they are all idempotent, since

$$s \cdot \alpha(w_1) \cdot \alpha(w_1) = s \cdot \alpha(w_1) = s.$$

Therefore, the words $w_1, \ldots, w_n$ can be joined in one step using the idempotent rule, and then the words w_0 and w_{n+1} can be added using the binary rule.

Comments on the proof. Actually $\|S\| = 3|S|$. To get this bound, we need a slightly more detailed analysis of what happens when Lemma 1 is applied (omitted here). Another important observation is that the proof yields an algorithm, which computes the factorization in linear time in the size of the word.

2 Fast String Algorithms

In this section, we show how factorization forests can be used to obtain fast algorithms for query evaluation. The idea[3] is to use the constant height of factorization forests to get constant time algorithms.

[2] Actually, in this case the monoid is a group.

[3] Suggested by Thomas Colcombet.

2.1 Infix Pattern Matching

Let $L \subseteq A^*$ be a regular language. An *L-infix query* in a word w is a query of the form "given positions $i \leq j$ in w, does the infix $w[i..j]$ belong to L?'

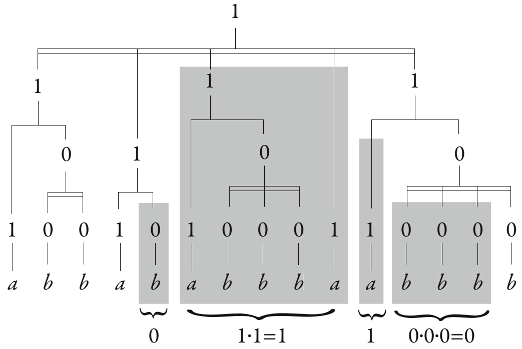
Below we state formally the theorem which was described in the introduction.

Theorem 2. *Let $L \subseteq A^*$ be a language recognized by $\alpha : A^* \to S$. Using an α-factorization forest f for a word $w \in A^*$, any L-infix query can be answered in time proportional to the height of f.*

Note that since f can be computed in linear time, the above result shows that, after a linear precomputation, infix queries can be evaluated in constant time. The constants in both the precomputation and evaluation are linear in S.

Proof

The proof is best explained by the following picture, which shows how the type of any infix can be computed from a constant number of labels in the factorization forest:



Below follows a more formal proof. We assume that each position in the word contains a pointer to the leaf of f that contains letter in that position. We also assume that each node in f comes with the number of its left siblings, the type of the word below that node, and a pointer to its parent node.

In the following x, y, z are nodes of f. The distance of x from the root is written $|x|$. We say a node y is *to the right* of a node x if y is not a descendant of x, and y comes after x in left-to-right depth-first traversal. A node y is *between x and z* if y is to the right of x and z is to the right of y. The word $bet(x,y) \in A^*$ is obtained by reading, left to right, the letters in the leaves between x and y. We claim that at most $|x|+|y|$ steps are needed to calculate the type of $bet(x,y)$. The claim gives the statement of the theorem, since membership in L only depends on the type of a word. The proof of the claim is by induction on $|x| + |y|$.

Consider first the case when x and y are siblings. Let $z_1, \ldots, z_n$ be the siblings between x and y. We use $sub(z)$ for the word obtained by reading, left to right, the leaves below z. We have

$$bet(x,y) = sub(z_1) \cdots sub(z_n) \ .$$

If $n = 0$, the type of $bet(x, y)$ is the identity in S. Otherwise, the parent node must be an idempotent node, for some idempotent $e \in S$. In this case, each $sub(z_i)$ has type e and by idempotency the type of $bet(x, y)$ is also e.

Consider now the case when x and y are not siblings. Either the parent of x is to the left of y or x is to the left of the parent of y. By symmetry we consider only the first case. Let z be the parent of x and let $z_1, \ldots, z_n$ be all the siblings to the right of x. We have

$$bet(x, y) = sub(z_1) \cdots sub(z_n) \cdot bet(z, y)$$

As in the first case, we can compute the type of $sub(z_1) \cdots sub(z_n)$ in a single step. The type of $bet(z, y)$ is obtained by induction assumption. □

The theorem above can be generalized to more general queries than infix queries[4]. An *n-ary query* Q for words over an alphabet A is a function that maps each word $w \in A^*$ to a set of tuples of word positions $(x_1, \ldots, x_n) \in \{1, \ldots, |w|\}^n$. We say such a query Q can be *evaluated with linear precomputation and constant delay* if there is an algorithm, which given an input word w:

- Begins by doing a precomputation in time linear in the length of w.
- After the precomputation, starts outputting all the tuples in $Q(w)$, with a constant number of operations between tuples.

The tuples will be enumerated in lexicographic order (i.e. first sorted left-to-right by the first position, then by the second position, and so on).

One way of describing an n-ary query is by using a logic, such as monadic second-order logic. A typical query would be: "the labels in positions $x_1, \ldots, x_n$ are all different, and for each $i, j \in \{1, \ldots, n\}$, the distance between x_i and x_j is even". By applying the ideas from Theorem 2, one can show:

Theorem 3. *An query definable in monadic second-order logic can be evaluated with linear precomputation and constant delay.*

2.2 Avoiding Factorization Forests

Recall that the constants in Theorem 2 were linear in the size of the monoid S. If, for instance, the monoid S is obtained from an automaton, then this can be a problem, since the translation from automata (even deterministic) to monoids incurs an exponential blowup. In this section, we show how to evaluate infix queries without using monoids and factorization forests.

Theorem 4. *Let $L \subseteq A^*$ be a language recognized by a deterministic automaton with states Q. For any word $w \in A^*$, one can calculate a data structure in time $O(|Q| \cdot |w|)$ such that any L-infix query can be answered in time $O(|Q|)$.*

It is important that the automaton is deterministic. There does not seem to be any easy way to modify the construction below to work for nondeterministic automata.

[4] The idea for this generalization was suggested by Luc Segoufin.

Let the input word be $w = a_1 \cdots a_n$. A *configuration* is a pair $(q, i) \in Q \times \{0, \ldots, n\}$, where i is called the *position* of the configuration. The idea is that (q, i) says that the automaton is in state q between the letters a_i and a_{i+1}. The *successor* of a configuration (q, i), for $i < n$, is the unique configuration on position $i + 1$ whose state coordinate is obtained from q by applying the letter a_{i+1}. A partial run is a set of configurations which forms a chain under the successor relation. Using this set notation we can talk about subsets of runs.

Below we define the data structure, show how it can be computed in time $O(|Q| \cdot |w|)$, and then how it can be used to answer infix queries in time $O(|Q|)$.

The data structure. The structure stores a set R partial runs, called *tapes*. Each tape is assigned a rank in $\{1, \ldots, |Q|\}$.

1. Each configuration appears in exactly one tape.
2. For any position i, the tapes that contain configurations on position i have pairwise different ranks.
3. Let (q, i) be a configuration appearing in tape $\rho \in R$. The tape of its successor configuration is either ρ or has smaller rank than ρ.

The data structure contains a record for each tape, which stores its rank as well as a pointer to its last configuration. Each configuration in the word stores a pointer to its tape, i.e. there is a two-dimensional array of pointers to tapes, indexed states q and by word positions i. We have a second two-dimensional array, indexed by word positions i and ranks j, which on position (i, j) stores the unique configuration on position i that belongs to a tape of rank j.

Computing the data structure. The data structure is constructed in a left-to-right pass through the word. Suppose we have calculated the data structure for a prefix $a_1 \cdots a_i$ and we want to extend it to the prefix $a_1 \cdots a_{i+1}$. We extend all the tapes that contain configurations for position i with their successor configurations. If two tapes collide by containing the same configuration on position $i + 1$, then we keep the conflicting configuration only in the tape with smaller rank and remove it from the tape with larger rank. We start new tapes for all configurations on position $i + 1$ that are not successors of configurations on position i, and assign to them ranks that have been freed due to collisions.

Using the data structure. Let (q, i) be a configuration. For a position $j \geq i$, let π be the run that begins in (q, i) and ends in position j. We claim that $O(|Q|)$ operations are enough to find the configuration from π on position j. How do we do this? We look at the last configuration (p, m) in the unique tape ρ that contains (q, i) (each tape has a pointer to its last configuration). If $m \geq j$, then $\rho \supseteq \pi$, so all we need to do is find the unique configuration on position j that belongs to a tape with the same rank as ρ (this will actually be the tape ρ). For this, we use the second two-dimensional array from the data structure. If $m < j$, we repeat the algorithm, by setting (q, i) to be the successor configuration of (p, m). This terminates in at most $|Q|$ steps, since each repetition of the algorithm uses a tape ρ of smaller rank.

Comments. After seeing the construction above, the reader may ask: what is the point of the factorization forest theorem, if it can be avoided, and the resulting construction is simpler and more efficient? There are two answers to this question. The first answer is that there are other applications of factorization forests. The second answer is more disputable. It seems that the factorization forest theorem, like other algebraic results, gives an insight into the structure of regular languages. This insight exposes results, which can then be proved and simplified using other means, such as automata. To the author's knowledge, the algorithm from Theorem 2 came before the algorithm from Theorem 4, which, although straightforward, seems to be new.

3 Well-Typed Regular Expressions

In this section, we use the Factorization Forest Theorem to get a stronger version of the Kleene theorem. In the stronger version, we produce a regular expression which, in a sense, respects the syntactic monoid of the language.

Let $\alpha : A^* \to S$ be a morphism. As usual, we write *type of* w for $\alpha(w)$. A regular expression E is called well-typed for α if for each of its subexpressions F (including E), all words generated by F have the same type.

Theorem 5. *Any language recognized by a morphism $\alpha : A^* \to S$ can be defined by a union of regular expression that are well-typed for α.*

Proof
By induction on k, we define for each $s \in S$ a regular expression $E_{s,k}$ generating all words of type s that have an α-factorization forest of height at most k:

$$E_{s,1} := \bigcup_{\substack{a \in A \cup \{\epsilon\} \\ \alpha(a)=s}} a \qquad E_{s,k+1} := \bigcup_{\substack{u,t \in S \\ ut=s}} E_{u,k} \cdot E_{t,k} \underbrace{\cup \quad (E_{s,k})^+}_{\text{if } s = ss} .$$

Clearly each expression $E_{s,k}$ is well-typed for α. The Factorization Forests Theorem gives an upper bound K on the height of α-factorizations needed to get all words. The well-typed expression for a language $L \subseteq A^*$ recognized by α is the union of all expressions $E_{s,K}$ for $s \in \alpha(L)$. □

3.1 An Effective Characterization of $\Sigma_2(<)$

In this section, we use Theorem 5 to get an effective characterization for Σ_2. First, we explain what we mean by effective characterization and Σ_2.

Let $\mathcal{L}$ be a class of regular languages (such as the class of finite languages, or the class of star-free languages, etc.). We say $\mathcal{L}$ has an *effective characterization* if there is an algorithm, which decides if a given regular language L belongs to the class $\mathcal{L}$. As far as decidability is concerned, the representation of L is not important, here we use its syntactic morphism. There is a large body of research on effective characterizations of classes of regular languages. Results are difficult to obtain, but the payoff is often a deeper understanding of the class $\mathcal{L}$.

Often the class $\mathcal{L}$ is described in terms of a logic. A prominent example is first-order logic. The quantifiers in a formula range over word positions. The signature contains a binary predicate $x < y$ for the order on word positions, and unary a predicate $a(x)$ for each letter $a \in A$ of the alphabet that tests the label of a position. For instance, the word property "the first position has label a" can be defined by the formula $\exists x\big(a(x) \wedge (\forall y\ y \geq x)\big)$. A theorem of McNaughton and Papert [14] says that first-order logic defines the same languages as star-free expressions, and Schützenberger [16] gives an effective characterization of the star-free languages (and therefore also of first-order logic).

A lot of attention has been devoted to the quantifier alternation hierarchy in first-order logic, where each level counts the alterations between $\forall$ and $\exists$ quantifiers in a first-order formula in prenex normal form. Formulas that have $n-1$ alternations (and therefore n blocks of quantifiers) are called Σ_n if they begin with $\exists$, and Π_n if they begin with $\forall$. For instance, the language "nonempty words with at most two positions that do not have label a" is defined by the Σ_2 formula

$$\exists x_1 \exists x_2 \forall y. \quad (y \neq x_1 \wedge y \neq x_2) \quad \Rightarrow \quad a(y)\ .$$

Effective characterizations are known for levels Σ_1 (a language has to be closed under adding letters), and similarly for Π_1 (the language has to be closed under removing letters). For languages that can be defined by a boolean combination of Σ_1 formulas, an effective characterization is given by Simon [17]. The last levels with a known characterization are Σ_2 and Π_2. For all higher finite levels, starting with boolean combinations of Σ_2, finding an effective characterization is an important open problem.

Below, we show how the well-typed expressions from Theorem 5 can be used to give an effective characterization of Σ_2. The idea to use the Factorization Forests Theorem to characterize Σ_2 first appeared in [15], but the proof below is based on [2]. Fix a regular language $L \subseteq A^*$. We say a word w *simulates* a word w' if the language L is closed under replacing w' with w. That is, $uw'v \in L$ implies $uwv \in L$ for any for any $u, v \in A^*$. Simulation is an asymmetric version of syntactic equivalence: two words are syntactically equivalent if and only if they simulate each other both ways.

Theorem 6. *Let $L \subseteq A^*$ be a regular language, and $\alpha : A^* \to S$ be its syntactic morphism. The language L can be defined in Σ_2 if and only if*

> *(*) For any words w_1, w_2, w_3 mapped by α to the same idempotent $e \in S$ and v a subsequence of w_2, the word $w_1 v w_3$ simulates $w_1 w_2 w_3$.*

Although it may not be immediately apparent, condition (*) can be decided when given the syntactic morphism of L. The idea is to calculate, using a fixpoint algorithm, for each $s, t \in S$ if some word of type s has a subsequence of type t.

The "only if" implication is done using a standard logical argument, and we omit it here. The more difficult "if" implication will follow from Lemma 2. The lemma uses *overapproximation*: we say a set of words K overapproximates a subset $K' \subseteq K$ if every word in K simulates some word in K'.

Lemma 2. *Assume (*). Any regular expression that is well-typed for α can be overapproximated by a language in Σ_2.*

Before proving the lemma, we show how it gives the "if" part in Theorem 6. Thanks to Theorem 5, the language L can be defined as a finite union of well-typed expressions. By Lemma 2, each of these can be overapproximated in Σ_2. The union of overapproximations gives exactly L: it clearly contains L, but contains no word outside L by definition of simulation.

Proof (of Lemma 2)
Induction on the size of the regular expression. The induction base is simple. In the induction step, we use closure of Σ_2 under union and concatenation.

Union in the induction step is simple: the union of overapproximations for E and F is an overapproximation of the union of E and F. For concatenation, we observe that simulation is compatible with concatenation: if w simulates w' and u simulates u', then wu simulates $w'u'$. Therefore, the concatenation of overapproximations for E and F is an overapproximation of $E \cdot F$.

The interesting case is when the expression is F^+. Since F is well typed, all words in F have type, say $e \in S$. Since F^+ is well-typed, e must be idempotent. Let M be an overapproximation of F obtained from the induction assumption. Let A_e be the set of all letters that appear in words of type e. As an overapproximation for F^+, we propose

$$K = \qquad M \quad \cup \quad M(A_e)^*M \ .$$

A Σ_2 formula for K can be easily obtained from a Σ_2 formula for M. Since every word in F is built from letters in A_e, we see that K contains F^+. To complete the proof of the lemma, we need to show that every word in K simulates some word in F^+. Let then w be a word in K. If w is in M, we use the induction assumption. Otherwise, w can be decomposed as $w = w_1vw_3$, with $w_1, w_3 \in M$ and v a word using only letters from A_e. By induction assumption, w_1 simulates some word $w_1' \in F$ and w_3 simulates some word $w_3' \in F$. Since simulation is compatible with concatenation, w_1vw_3 simulates $w_1'vw_3'$. Since e is idempotent, each word in $(A_e)^*$ is a subsequence of some word of type e. In particular, v is a subsequence of some word v' of type s. By condition (*), $w_1'vw_2'$ simulates $w_1'v'w_2' \in F^+$. The result follows by transitivity of simulation. □

Corollary 1. *A language is definable in Σ_2 if and only if it is a union of languages of the form*

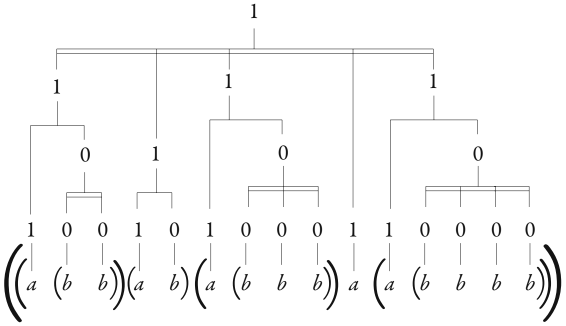
$$A_0^*a_1A_1^* \cdots A_{n-1}^*a_nA_n^* \tag{2}$$

Proof
The "if" part is immediate, since each expression as in (2) can be described in Σ_2. The "only if" part follows by inspection of the proof of Lemma 2 where, instead of a formula of Σ_2, we could have just as well produced a union of languages as in (2). □

4 Transducers

The proof of the Factorization Forests Theorem also shows that factorization forests can be computed in linear time. In this section we strengthen that statement by showing that factorization forests can be produced by transducers.

A tree can be written as a word with matched parentheses. This idea can be applied to factorizations, as shown by the following picture:



To aid reading, we have removed the parentheses around individual letters (which correspond to factorization forests of height 1).

We can therefore define the word encoding of a factorization as a word over an extended alphabet $A \cup \{(,)\}$ that also contains an opening parenthesis, and a closing one. We write w_f for the word encoding of a factorization f. The following lemma shows that factorizations can be calculated by a transducer.

Lemma 3. *Fix a morphism $\alpha : A^* \to S$ and a height $k \in \mathbb{N}$. There is a nondeterministic transducer $\mathcal{T}_k : A^* \to (A \cup \{(,)\})^*$, which produces on input $w \in A^*$ the word encodings of all α-factorizations of w of height at most k.*

Proof
Induction on k. □

There are two problems with the transducer $\mathcal{T}_k$.

The first is nondeterminism. For instance, we might want to use the transducer to find a factorization forest, and nondeterminism seems to gets in the way. This particular problem with nondeterminism can be dealt with: as for any nondeterministic transducer, one can compute (some) output in $\mathcal{T}_k(w)$ in time proportional to the length of w times the number of states in $\mathcal{T}_k$. (In particular, assuming that the morphism α is fixed, we get a linear time algorithm for computing an α-factorization.) However, nondeterminism turns out to be a serious problem for applications to tree languages, as we will see later.

A second problem is that $\mathcal{T}_k$ has a lot of states. This is because the construction of $\mathcal{T}_k$, at least the easy inductive construction suggested above, gives a state space that is exponential in k.

A nice solution to this problem was proposed by Thomas Colcombet. He shows that if the conditions on a factorization forest are relaxed slightly, then the factorization can be output by a deterministic transducer with $O(|S|)$ states.

What is the relaxation on factorizations? Recall the idempotent rule, which allowed to split a word w into $w = w_1 \cdots w_n$ as long as all the factors $w_1, \ldots, w_n$ had the same idempotent type. This requirement could be stated as

$$\alpha(w_i) \cdot \alpha(w_j) = \alpha(w_j) \cdot \alpha(w_i) = \alpha(w_i) \qquad \text{for all } i, j \in \{1, \ldots, n\}.$$

In other words, the type of any word w_i absorbs the type of any other word w_j, both on the left and on the right. In [5,6] Colcombet proposed a relaxed version of this rule, where the type only absorbs to the right:

$$\alpha(w_i) \cdot \alpha(w_j) = \alpha(w_i) \qquad \text{for all } i, j \in \{2, \ldots, n-1\}.$$

We will use the term *forward Ramseyan rule* for a rule that allows a split $w = w_1 \cdots w_n$ under the above condition. A factorization that uses the forward Ramseyan rule instead of the idempotent rule is called a *forward Ramseyan factorization.* Every factorization that uses the idempotent rule is a forward Ramseyan factorization (since the condition in the forward Ramseyan rule is weaker than the condition in the idempotent rule), but not vice versa.

Despite being more relaxed, in most cases the forward Ramseyan rules gives the same results as the idempotent rule. Consider, for example, the infix problem from Theorem 2. Suppose we have a word split $w = w_1 \cdots w_n$ according the forward Ramseyan rule, and that we know the values $\alpha(w_1), \ldots, \alpha(w_n)$. Suppose that we want to calculate the type $\alpha(w_i \cdots w_j)$ for some $i \leq j \in \{2, \ldots, n-1\}$. Thanks to the forward Ramseyan rule, this type is

$$\alpha(w_i \cdots w_j) = \alpha(w_i)\alpha(w_{i+1})\alpha(w_{i+2} \cdots w_j) = \alpha(w_i)\alpha(w_{i+2} \cdots w_j) = \cdots = \alpha(w_i)\ .$$

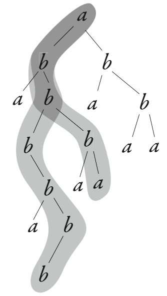
If we are interested in the case of $i = 1$ (a similar argument works for $j = n$), then we first find the type $\alpha(w_2 \cdots w_j)$ and then prepend the type of $\alpha(w_1)$.

The reason why Colcombet introduced forward Ramseyan factorizations is that they can be produced by a deterministic transducer (we use the same encoding of factorizations as words over the alphabet A_S).

Theorem 7 (Colcombet). *Fix a morphism $\alpha : A^* \to S$. There is a deterministic transducer $\mathcal{T}_k : A^* \to (A \cup \{(,)\})^*$, which produces, on input $w \in A^*$, the word encoding of a forward Ramseyan factorization of w of height at most $|S|$.*

We cite below two applications of this result. The first concerns trees, and the second concerns infinite words.

Trees. Suppose we have a tree, and we want to calculate factorizations for words that label paths in the tree. There are two difficulties, both related to the fact that paths have common prefixes, as in the picture below:



.

The first difficulty is that the combined length of all paths in the tree can be quadratic in the number of nodes. The second difficulty is that the factorizations for two different paths may be inconsistent on their common prefixes. Both of these difficulties are solved by using the deterministic transducer from Theorem 7, and running it on each path, from root to leaf. Along these lines, Theorem 7 was used in [4] to provide a linear time algorithm for evaluating XPath queries on XML documents.

Infinite words. The transducer in Theorem 7 can also be used on an infinite word $w = a_1 a_2 \cdots$. It also produces a forward Ramseyan factorization. The only difference is that after some point, we will start to see an infinite sequence of matched parentheses $(..)(..)(..)\cdots$ at the same nesting level (some of the initial parentheses might remain open forever). This construction has been used in [6] to determinize automata on infinite words (that is, convert a Büchi automaton into an equivalent Muller automaton).

5 Limitedness

In this last section, we talk about limitedness of automata. This is the original setting in which the Factorization Forests Theorem were used, so a discussion of the theorem would be incomplete without mentioning limitedness. On the other hand, the subject is quite technical (but fascinating), so we only sketch the problem, and point the reader to the literature.

A *distance automaton* is a nondeterministic automaton where a subset of the transitions is declared costly. The *cost* of a run ρ is the number of times it uses the costly transitions. The cost of a word $w \in A^*$ is the minimal cost of a run (from an initial to a finite state) over this word. If there is no run, the minimal cost is ∞. The automaton is called *limited* if there is a finite bound on the cost of all words.

We want an algorithm that decides if a distance automaton is limited. In other words, we want to decide if the expression

$$\max_{w \in A^*} \quad \min_{\rho \in \mathrm{runs}(w)} \quad \mathrm{cost}(\rho)$$

has a finite value. The difficulty of the problem comes from the alternation between max and min. If the expression had been max max, the problem could be decided by simply searching for a loop in the automaton that uses a costly state. (In particular, the limitedness problem is straightforward for deterministic automata.) If the expression had been min min or min max, the problem would trivialize, since the value would necessarily be finite.

The limitedness problem is closely related to star height. The star height of a regular expression is the nesting depth of the Kleene star. For instance, the expression $a^* + b^*$ has star height 1, while the expression $((a + b)^*aa)^*$ has star height 2, although it is equivalent to $\epsilon + (a + b)^*aa$, which has star height 1. Complementation and intersection are not allowed in the expressions (when complementation is allowed, we are talking about generalized star height). The star height problem is to decide, given a regular language L and a number k, if there exists an expression of star height k that defines L. This famous problem was posed by Eggan [7], and was open for 25 years, until it was solved by Hashiguchi [9]. An important technique in the star height problem is limitedness of distance automata. Distance automata have been introduced by Hashiguchi in [8], and the limitedness problem was also studied by Leung [13] and Simon [19]. The latter paper is the first important application of the Factorization Forests Theorem.

The current state of the art in the star height problem is the approach of Daniel Kirsten [10], who uses an extension of distance automata. The extended model is called a distance desert automaton, and it extends a distance automaton in two ways. First, a distance desert automaton keeps track of several costs (i.e. if the cost is seen as the value of a counter, then there are several counters). Second, the cost can be reset, and the cost of a run is the maximal cost seen at any point during the run. The star height problem can be reduced to limitedness of distance desert automata: for each regular language L and number k, one can write a distance desert automaton that is limited if and only if the language L admits an expression of star height k. In [10], Daniel Kirsten shows how to decide limitedness for distance desert automata, and thus provides another decidability proof for the star height problem.

Determinization is another important, and still open, problem for distance automata: decide if a distance automaton can be determinized or made unambiguous (not always possible). Recently, Kirsten and Lombardy used factorization forests to prove a special case, for polynomially ambiguous automata [11].

A related line of work was pursued in [3]. This paper considered a type of distance desert automaton (under the name BS-automaton), which would be executed on an infinite word. (The same type of automata was also considered in [1], this time under the name of R-automata.) The acceptance condition in a BS-automaton talks about the asymptotic values of the cost in the run, e.g. one can write an automaton that accepts infinite words where the cost is unbounded. The main contribution in [3] is a complementation result. This complementation result depends crucially on the Factorization Forests Theorem.

References

1. Abdulla, P.A., Krcal, P., Yi, W.: R-automata. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 67–81. Springer, Heidelberg (2008)
2. Bojańczyk, M.: The common fragment of ACTL and LTL. In: Foundations of Software Science and Computation Structures, pp. 172–185 (2008)
3. Bojańczyk, M., Colcombet, T.: Omega-regular expressions with bounds. In: Logic in Computer Science, pp. 285–296 (2006)
4. Bojanczyk, M., Parys, P.: XPath evaluation in linear time. In: PODS, pp. 241–250 (2008)
5. Colcombet, T.: A combinatorial theorem for trees. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 901–912. Springer, Heidelberg (2007)
6. Colcombet, T.: Factorisation forests for infinite words. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) FCT 2007. LNCS, vol. 4639, pp. 226–237. Springer, Heidelberg (2007)
7. Eggan, L.C.: Transition graphs and the star height of regular events. Michigan Math. J. 10, 385–397 (1963)
8. Hashiguchi, K.: Limitedness theorem on finite automata with distance functions. Journal of Computer and System Sciences 24, 233–244 (1982)
9. Hashiguchi, K.: Algorithms for determining relative star height and star height. Inf. Comput. 78(2), 124–169 (1988)
10. Kirsten, D.: Distance desert automata and the star height problem. Theoretical Informatics and Applications 39(3), 455–511 (2005)
11. Kirsten, D., Lombardy, S.: Deciding unambiguity and sequentiality of polynomially ambiguous min-plus automata. In: STACS 2009 Proceedings, pp. 589–600 (2009)
12. Kufleitner, M.: The height of factorization forests. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 443–454. Springer, Heidelberg (2008)
13. Leung, H.: The topological approach to the limitedness problem on distance automata. In: Idempotency, pp. 88–111 (1998)
14. McNaughton, R., Papert, S.: Counter-Free Automata. MIT Press, Cambridge (1971)
15. Pin, J.-É., Weil, P.: Polynomial closure and unambiguous product. Theory Comput. Systems 30, 1–30 (1997)
16. Schützenberger, M.P.: On finite monoids having only trivial subgroups. Information and Control 8, 190–194 (1965)
17. Simon, I.: Piecewise testable events. In: Brakhage, H. (ed.) GI-Fachtagung 1975. LNCS, vol. 33. Springer, Heidelberg (1975)
18. Simon, I.: Factorization forests of finite height. Theoretical Computer Science 72, 65–94 (1990)
19. Simon, I.: On semigroups of matrices over the tropical semiring. ITA 28(3-4), 277–294 (1994)

Weighted versus Probabilistic Logics[*]

Benedikt Bollig and Paul Gastin

LSV, ENS Cachan, CNRS, INRIA Saclay, France
{bollig,gastin}@lsv.ens-cachan.fr

Abstract. While a mature theory around logics such as MSO, LTL, and CTL has been developed in the pure boolean setting of finite automata, weighted automata lack such a natural connection with (temporal) logic and related verification algorithms. In this paper, we will identify weighted versions of MSO and CTL that generalize the classical logics and even other quantitative extensions such as probabilistic CTL. We establish expressiveness results on our logics giving translations from weighted and probabilistic CTL into weighted MSO.

1 Introduction

Connections between logic and classical automata theory have become indispensable tools in the modeling and verification of computer systems. Usually, a logical formula φ appears as a specification, a property that a system has to fulfill, whereas an automaton $\mathcal{A}$ represents a finite-state abstraction of the system itself. Prominent examples of specification formalisms are monadic second-order (MSO) logic [35], the μ-calculus [26], and the temporal logics LTL [31] and CTL [11]. Two questions that naturally arise in this context are the *satisfiability problem* (does there exist any model of φ?) and the *model-checking problem* (do all behaviors of $\mathcal{A}$ satisfy φ?) [12].

Both logic and automata semantics give rise to a formal language that separates accepted from non-accepted behaviors. This corresponds to assigning a truth value, taken from the boolean semiring, to a behavior. When it comes to modeling and verifying quantitative systems, however, the value of a behavior is not necessarily boolean but might, e.g., be a probability of acceptance or represent a reward. Classical automata theory and logic is not suited to account for such subtleties. This led to various specialized extensions of finite automata (over finite or infinite behaviors) such as probabilistic automata [36,33], timed automata [1], or automata with energy constraints [6], each coming with dedicated specification formalisms and approaches to related model-checking problems. In the particular case of stochastic systems, the temporal logics PCTL [24] and PCTL* [15] and corresponding model-checking techniques have been developed to reason about probabilities of events.

A generic concept of adding weights to qualitative systems is provided by the theory of weighted automata [27,28]. Unlike finite automata, which are based on

[*] Partially supported by projects ARCUS Île de France-Inde and ANR-06-SETIN-003 DOTS.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 18–38, 2009.

the boolean semiring, weighted automata build on more general structures such as the natural or real numbers (equipped with the usual addition and multiplication) or the probabilistic semiring. Hence, a weighted automaton associates with any possible behavior a weight beyond the usual boolean classification of "acceptance" or "non-acceptance". Automata with weights have produced a well-established theory and come, e.g., with a characterization in terms of rational expressions, which generalizes a famous counterpart in the unweighted setting. Equipped with a solid theoretical basis, weighted automata finally found their way into numerous application areas such as natural language processing and speech recognition [30], or digital image compression [14].

What is still missing in the theory of weighted automata is a satisfactory connection with logic that could lead to a general approach to related satisfiability and model-checking problems. A first step towards a logical characterization of weighted automata has been made in terms of a weighted MSO logic capturing the recognizable formal power series (i.e., the behaviors of finite weighted automata) [16,17]. This generalizes the classical equivalence of MSO logic and finite automata [8,21]. While, however, in the qualitative setting, temporal logics such as LTL and CTL appear as fragments of MSO logic and the μ-calculus, a natural transfer of such an embedding to weighted automata is beyond the state of the art. Let us mention here some promising works that deal with this issue. In [7], Buchholz and Kemper propose valued computation-tree logic (CTL$) and corresponding model-checking algorithms for weighted Kripke structures, but do not address satisfiability and expressiveness issues. A weighted linear μ-calculus on words was defined by Meinecke, who establishes its expressive equivalence to certain ω-rational formal power series [29]. An extension towards branching structures, the identification of temporal-logic fragments, and the definition of a corresponding model-checking problem are left for future work.

Actually, only very few efforts have been made to establish a smooth connection of weighted automata with MSO and, in particular, temporal logics. We do not aim at giving final solutions to these largely open questions, but will propose a precise description of missing concepts. It is the aim of this paper to identify a weighted MSO logic as well as linear-time and branching-time logics that subsume, in a natural manner, existing quantitative logics. We will actually study the relation between our new logics and the branching-time logics PCTL and PCTL*, thus putting an emphasis on probabilistic systems.

Outline. In Section 2, we settle some notation and introduce semirings and weighted automata. Towards the end of that section, we identify probabilistic automata as a special case, which can be embedded in our framework by using a specific semiring. Sections 3 and 4 present an extended weighted MSO logic and, respectively, weighted versions of the temporal logics CTL and CTL*. They are all interpreted over unfoldings of weighted automata as introduced in Section 2 and include as special cases PCTL and PCTL*. In Section 5, we establish that our weighted temporal logic is expressible in our extended weighted MSO, transferring the well-known qualitative counterpart to the weighted case. It is also shown that the probabilistic logic PCTL can be embedded in weighted

MSO. We conclude with Section 6, in which we suggest several directions for future work.

2 Preliminaries

Words. Let Σ be an alphabet, i.e., a nonempty finite set. The set of finite words over Σ is denoted by Σ^*, the set of infinite words by Σ^ω. Moreover, we let $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$, ε denoting the empty word, and $\Sigma^\infty = \Sigma^* \cup \Sigma^\omega$. For $w \in \Sigma^\infty$, the length of w is denoted by $|w| \in \mathbb{N} \cup \{\omega\}$. In particular, $|\varepsilon| = 0$, and $|w| = \omega$ iff $w \in \Sigma^\omega$. Let $w = a_1 a_2 \ldots \in \Sigma^\infty$. For $i \leq |w|$, we denote $w[i] = a_1 \ldots a_i$ the prefix of w of length i, in particular, $w[0] = \varepsilon$. We denote by $\mathrm{Pref}(w)$ the set of *finite prefixes* of w. Instead of $u \in \mathrm{Pref}(v)$, we also write $u \leq v$. We write $u < v$ if, in addition, $u \neq v$. The mapping Pref is extended to sets $L \subseteq \Sigma^\infty$ in the expected manner: $\mathrm{Pref}(L) = \bigcup_{w \in L} \mathrm{Pref}(w)$. We say that $L \subseteq \Sigma^\infty$ is *prefix-closed* if $\mathrm{Pref}(L) \subseteq L$.

Semirings. A semiring is a structure $\mathbb{K} = (K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ where K is a set, $\mathbf{0}$ and $\mathbf{1}$ are constants, and $\oplus : K \times K \to K$ and $\otimes : K \times K \to K$ are binary operations, called addition and, respectively, multiplication such that $(K, \oplus, \mathbf{0})$ is a commutative monoid, $(K, \otimes, \mathbf{1})$ is a monoid, multiplication distributes over addition, and $\mathbf{0} \otimes k = k \otimes \mathbf{0} = \mathbf{0}$ for every $k \in K$. We say that $\mathbb{K}$ is *commutative* if $\otimes$ is commutative. Some popular semirings are the semiring of natural numbers $(\mathbb{N}, +, \cdot, 0, 1)$ (with the usual addition and multiplication on natural numbers), the 2-valued Boolean algebra $\mathbb{B} = (\{\mathbf{0}, \mathbf{1}\}, \vee, \wedge, \mathbf{0}, \mathbf{1})$, and the tropical semiring $(\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$. In this paper, we will focus on $\mathbb{P}\mathrm{rob} = (\mathbb{R}_{\geq 0}, +, \cdot, 0, 1)$, the *probabilistic semiring*, which will allow us to model probabilistic systems.[1]

The classical semirings work fine for finite trees. However, the trees that we consider might in general be infinite. We will therefore deal with infinite sums and products wrt. $\oplus$ and $\otimes$, respectively. Unfortunately, unlike finite sums and products, they do not always have a value in the semiring at hand. However, we can identify examples of infinite sums and products that are essential for our purposes and that are always defined. For arbitrary semirings $(K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$, a (possibly uncountable) index set I, and $k_i \in K$ for $i \in I$, the sum $\bigoplus_{i \in I} k_i$ is defined whenever $k_i \neq \mathbf{0}$ for only finitely many i. Similarly, $\bigotimes_{i \in I} k_i$ is defined if $k_i \neq \mathbf{1}$ for finitely many i, assuming the semiring commutative or using some total order on the index set I. Considering concrete semirings such as the real numbers, a prominent infinite sum is the geometric series $\sum_{n \in \mathbb{N}} \frac{1}{2^n}$. We let its value be the limit $\lim_{n \to \infty} \sum_{n \in \mathbb{N}} \frac{1}{2^n} = 2$ of its partial sums, which is therefore defined. An example of an undefined infinite sum over the real numbers is $\sum_{n \in \mathbb{N}} \frac{1}{n}$, whose value is not in $\mathbb{R}$. We refer to the textbook [25] for a comprehensive introduction into infinite series.

If not otherwise stated, $\mathbb{K}$ will, in the following, be an arbitrary semiring $(K, \oplus, \otimes, \mathbf{0}, \mathbf{1})$ and Σ will be an alphabet.

[1] Note that $([0,1], \max, \cdot, 0, 1)$ is sometimes considered as the probabilistic semiring as its universe restricts to probabilities. It is, however, not suitable for our purposes, as it neglects addition and, thus, does not allow one to model non-determinism.

Weighted Trees. The behavior of a non-quantitative finite-state system is often described as a (possibly infinite) tree-unfolding, whose paths constitute all possible execution sequences of the system. When we move to the quantitative setting where transitions come with weights from a semiring, then this unfolding is equipped with weights as well, which gives rise to the following definition.

Definition 1. *Let D be a nonempty finite set of* directions*. A* weighted tree *(over D, $\mathbb{K}$, and Σ) is a partial mapping $t : D^* \rightharpoonup K \times \Sigma$ such that* dom(t) *is prefix-closed*[2] *and $t(\varepsilon) = (\mathbf{1}, a)$ for some distinguished element a from Σ.*[3]

The set of trees over D, $\mathbb{K}$, and Σ is denoted by *Trees*$(D, \mathbb{K}, \Sigma)$. We will, however, simply write *Trees* if the parameters are understood. Let $t \in$ *Trees* be a weighted tree. It is convenient to split t into two partial mappings $\kappa_t : D^* \rightharpoonup K$ and $\ell_t : D^* \rightharpoonup \Sigma$ to extract from $t(u) = (k, a)$ the values $\kappa_t(u) = k$ and $\ell_t(u) = a$. Elements from $\mathrm{dom}(t)$ are called *nodes* of t. The empty word $\varepsilon \in \mathrm{dom}(t)$ is the *root*. A node u is a *leaf* if it is maximal in $\mathrm{dom}(t)$ for the prefix ordering, i.e., if $uD \cap \mathrm{dom}(t) = \emptyset$. If u is not a leaf, then it has some *successors*, which are nodes of the form ud with $d \in D$. The set of leaves of t is denoted by $\mathrm{Leaves}(t)$. A *branch* of t is a leaf or an infinite word whose finite prefixes are in $\mathrm{dom}(t)$. We thus define $\mathrm{Branches}(t)$ to be $\mathrm{Leaves}(t) \cup \{u \in D^\omega \mid \mathrm{Pref}(u) \subseteq \mathrm{dom}(t)\}$. Tree t is called *finite* if $\mathrm{dom}(t)$ is finite. Otherwise, it is called *infinite*. Note that we deal with unordered trees of bounded degree: we do not fix a particular order on D, and every node has at most $|D|$ successors.

We sometimes manipulate subtrees or restrictions of trees that we define now. For $u \in \mathrm{dom}(t)$, the *subtree* of t rooted at u is denoted by t_u and given by $t_u(w) = t(uw)$ for all $w \in D^+$ (and indeed $t_u(\varepsilon) = (\mathbf{1}, a)$). Given a language $L \subseteq D^\infty$, the tree $t_{|L}$ is the restriction of t to $\mathrm{dom}(t) \cap \mathrm{Pref}(L)$: $t_{|L}(u) = t(u)$ if $u \in \mathrm{dom}(t) \cap \mathrm{Pref}(L)$, and $t_{|L}(u)$ is undefined otherwise. Alternatively, one may extract a tree based on a language $L \subseteq \Sigma^\infty$ by keeping only those branches whose labeling wrt. ℓ is in $\mathrm{Pref}(L)$. Formally, we define

$$\widetilde{L} = \{u \in D^* \mid \ell_t(u[1])\ell_t(u[2]) \cdots \ell_t(u) \in \mathrm{Pref}(L)\}$$

and we are interested in $t_{|\widetilde{L}}$. When we further restrict the tree to branches that end in nodes located at directions from a set $D' \subseteq D$, then we obtain trees $t_{|L \cap D^* D'}(u)$ and $t_{|\widetilde{L} \cap D^* D'}(u)$ respectively.

Let us define a partial mapping $\widehat{\kappa} : \mathit{Trees} \rightharpoonup K$, which associates with a tree its *measure*, a weight in the semiring $\mathbb{K}$, if it exists. Intuitively, we sum over the weights of every branch. The weight of a branch, in turn, is the product of weights that are assigned to its nodes. So let, for $t \in$ *Trees*,

$$\widehat{\kappa}(t) = \bigoplus_{u \in \mathrm{Branches}(t)} \bigotimes_{v \in \mathrm{Pref}(u)} \kappa_t(v) = \bigoplus_{d \in D \cap \mathrm{dom}(t)} \kappa_t(d) \otimes \widehat{\kappa}(t_d)\ .$$

[2] Let $\mathrm{dom}(t)$ be the set of words $u \in D^*$ such that $t(u)$ is defined.

[3] The value of ε will actually not be relevant so that we assume a unique value $(\mathbf{1}, a)$.

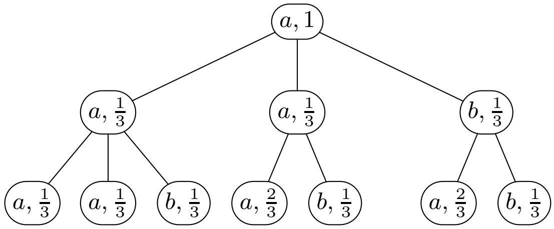


Fig. 1. A finite weighted tree over $\mathbb{P}\mathrm{rob}$, and $\{a, b\}$

Example 1. Figure 1 depicts a finite weighted tree t over $\mathbb{P}\mathrm{rob}$, and $\Sigma = \{a, b\}$. The branches of t are its leaves. We have

$$\widehat{\kappa}(t_{|\widetilde{\{aa\}}}) = \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{2}{3} = \frac{4}{9} \ .$$

Weighted Automata. In a weighted automaton, the values of a semiring that are collected along a run of the automaton are multiplied, while values of runs are summed-up.

Definition 2. *A* weighted automaton *over $\mathbb{K}$ and Σ is a quadruple $(Q, \lambda, \mu, \gamma)$ where Q is the nonempty finite set of* states*, $\mu : \Sigma \to K^{Q \times Q}$ is the* transition weight function*, and $\lambda, \gamma \in Q \to K$ provide weights for entering and leaving a state, respectively.*

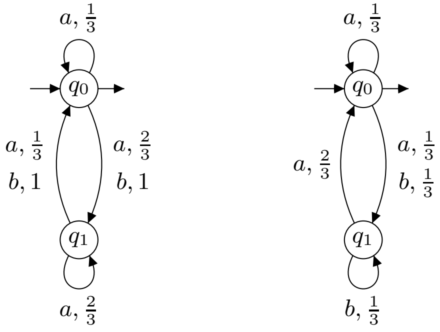
Let $\mathcal{A} = (Q, \lambda, \mu, \gamma)$ be a weighted automaton over $\mathbb{K}$ and Σ. For $a \in \Sigma$, $\mu(a)$ is a $(Q \times Q)$-matrix, and we let $\mu(a)_{p,q}$ or also $\mu(p, a, q)$ refer to its (p, q)-entry. The mapping μ uniquely extends to a monoid homomorphism $\Sigma^* \to (K^{Q \times Q}, \cdot, \mathsf{id})$ with unit matrix id where $\mathsf{id}_{p,p} = \mathbf{1}$ and $\mathsf{id}_{p,q} = \mathbf{0}$ if $p \neq q$. The semantics of $\mathcal{A}$ is given as a mapping $[\![\mathcal{A}]\!] : \Sigma^* \to K$ called a *formal power series*. Namely, for $w \in \Sigma^*$, one lets $[\![\mathcal{A}]\!](w) = \lambda \cdot \mu(w) \cdot \gamma$ with the usual matrix multiplication, considering λ as a row and γ as a column vector.

In the following, we will make two assumptions on initial and final weights:

(1) there is $q \in Q$ such that $\lambda(q) = \mathbf{1}$ and $\lambda(q') = \mathbf{0}$ for all $q' \in Q \setminus \{q\}$, and
(2) for all $q \in Q$, $\gamma(q) \in \{\mathbf{0}, \mathbf{1}\}$.

It is folklore that assuming (1) and (2) does not restrict generality, as any weighted automaton $\mathcal{A} = (Q, \lambda, \mu, \gamma)$ can be transformed into a weighted automaton $\mathcal{A}'$ that satisfies (1) and (2) and such that $[\![\mathcal{A}]\!](w) = [\![\mathcal{A}']\!](w)$ for all $w \in \Sigma^+$. Note that the transformation does not necessarily preserve the weight originally assigned to the empty word.

As we will, in the following, restrict to weighted automata that satisfy (1) and (2), we can represent $\mathcal{A}$ as the tuple (Q, q_0, μ, F) where the *initial state* $q_0 \in Q$ is the unique state q with $\lambda(q) = \mathbf{1}$, and $F = \{q \in Q \mid \gamma(q) = \mathbf{1}\}$ is the set of *final states*.



rPFA $\mathcal{A}_1$

gPFA $\mathcal{A}_2$

Fig. 2. Weighted automata over $\mathbb{P}\mathrm{rob}$

We will now define an alternative semantics and associate with a weighted automaton $\mathcal{A} = (Q, q_0, \mu, F)$ its unfolding in terms of an infinite weighted tree over $D = \Sigma \times Q$, $\mathbb{K}$, and Σ. Formally, the *unfolding* of $\mathcal{A}$, denoted by $t^{\mathcal{A}}$, is defined to be the tree $t \in \mathit{Trees}(D, \mathbb{K}, \Sigma)$ such that, for all $u \in D^*$ and $(a, q), (a', q') \in D$, the following hold: $\kappa_t((a,q)) = \mu(q_0, a, q)$, $\kappa_t(u(a,q)(a',q')) = \mu(q, a', q')$, and $\ell_t(u(a,q)) = a$. For every $w \in \Sigma^+$, we have

$$[\![\mathcal{A}]\!](w) = \widehat{\kappa}\Big(t^{\mathcal{A}}_{|\widetilde{\{w\}}\cap D^*(\Sigma\times F)}\Big)\ .$$

Example 2. Consider Figure 2 depicting weighted automata $\mathcal{A}_1$ and $\mathcal{A}_2$ over $\mathbb{P}\mathrm{rob}$ and Σ. In all cases, q_0 is both the initial state and the only final state. Missing values in $\mathcal{A}_1$ and $\mathcal{A}_2$ are supposed to be 0. For $n \in \mathbb{N}$, we have $[\![\mathcal{A}_1]\!](ab^n) = \frac{1}{3}$ if n is even, and $[\![\mathcal{A}_1]\!](ab^n) = \frac{2}{3}$ if n is odd. The set $[\![\mathcal{A}_1]\!](\Sigma^*) = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$ is actually finite. This does not apply to $[\![\mathcal{A}_2]\!]$. We have, e.g., $[\![\mathcal{A}_2]\!](a) = [\![\mathcal{A}_2]\!](aa) = \frac{1}{3}$ and $[\![\mathcal{A}_2]\!](aaa) = \frac{5}{27}$. Note that $[\![\mathcal{A}_2]\!](w) = 0$ whenever w ends with the letter b. Figure 1 depicts the unfolding of $\mathcal{A}_2$ up to depth 2, i.e., $t^{\mathcal{A}_2}_{|D^2}$.

When we consider our examples to be probabilistic automata (see Definitions 3, 4), it will be evident to which extent all these values can be interpreted as probabilities of acceptance.

Probabilistic Finite Automata. There is a wide range of automata models that incorporate probabilities. We refer the reader to [36, 34] for an overview. Our generic framework of weighted automata allows us to treat many of them in a unified manner.

One basically distinguishes between *reactive* and *generative* probabilistic automata. Reactive models are input-driven: an action (from our alphabet Σ) determines a probability distribution on the set of states. The next state of an execution is then randomly drawn according to this distribution. In a generative model, the next state *and* the action are chosen according to a probability distribution so that we might call the model probability-driven.

Definition 3. *A* reactive probabilistic finite automaton (rPFA) *over Σ is a weighted automaton (Q, q_0, μ, F) over* $\mathbb{P}$rob *and Σ such that, for every $q \in Q$ and $a \in \Sigma$, we have $\sum_{q' \in Q} \mu(q, a, q') \in \{0, 1\}$.*[4] *In other words, we require that $\mu(a)$ is a stochastic matrix for every action $a \in \Sigma$.*

The semantics of an rPFA $\mathcal{A} = (Q, q_0, \mu, F)$ computes, for each word $w \in \Sigma^*$, a probability of acceptance as defined directly in [33]. In the following sections, we will consider mechanisms that allow us to extract from the formal power series $[\![\mathcal{A}]\!]$ a boolean language. We may, for example, be interested in the set $L = \{w \in \Sigma^* \mid [\![\mathcal{A}]\!](w) > p\}$ of words whose probabilities exceed a given threshold $p \in [0, 1]$. In general, L can be non-regular, unless p is an *isolated cut-point* of $[\![\mathcal{A}]\!]$ [33]. Moreover, it is in general undecidable if L is empty:

Theorem 1 (Rabin [33]). *The following problem is undecidable:*
Input*: Alphabet Σ;* rPFA *$\mathcal{A}$ over Σ; $p \in [0, 1]$.*
Question*: Is there $w \in \Sigma^*$ such that $[\![\mathcal{A}]\!](w) > p$?*

Definition 4. *A* generative probabilistic finite automaton (gPFA) *over Σ is a weighted automaton (Q, q_0, μ, F) over* $\mathbb{P}$rob *and Σ such that, for every $q \in Q$, we have $\sum_{(a,q') \in \Sigma \times Q} \mu(q, a, q') \in \{0, 1\}$.*[5]

Example 3. Let us reconsider our sample automata from Figure 2 (cf. Example 2) and let $w \in \Sigma^*$. As $\mathcal{A}_1$ is an rPFA, the weight $[\![\mathcal{A}_1]\!](w)$ can be interpreted as the probability of reaching a final state when w is used as a *scheduling policy*. E.g., $[\![\mathcal{A}_1]\!](aab) = \frac{2}{3}$. On the other hand, $[\![\mathcal{A}_2]\!](w)$ is the probability of executing w *and* ending in a final state, under the precondition that we perform $|w|$ steps. Remember that, e.g., $[\![\mathcal{A}_2]\!](a) = [\![\mathcal{A}_2]\!](aa) = \frac{1}{3}$.

3 Extended Weighted MSO

A weighted MSO logic was proposed by Droste and Gastin [16, 17] in order to extend Büchi's and Elgot's fundamental theorems [8, 21, 9] from the boolean setting to the quantitative (weighted) one. This logic was designed in order to obtain an equivalence between weighted languages (or formal power series) generated by weighted automata and those definable in this weighted MSO logic.

Other quantitative logics have been introduced and studied, e.g., PCTL [24] or PCTL* [15, 10] which are probabilistic versions of the computation tree logic. These logics are evaluated on probabilistic transition systems, which are nothing

[4] Another common term for this model is simply *probabilistic finite automaton* [33]. When we neglect final states and consider the unfolding semantics rather than formal power series, then rPFA essentially correspond to the classical model of a *Markov decision process* (MDP) [32].

[5] If Σ is a singleton set, then a gPFA can be understood as a discrete-time Markov chain (DTMC). Otherwise, elements from Σ can be considered as sets of propositions that hold in the target state of a corresponding transitions. Then, we actually deal with a labeled DTMC [24].

but special instances of weighted automata as seen in Section 2. Hence, comparing weighted MSO and these logics is a natural question. It is easy to observe that these logics are *uncomparable*. Though formally correct, this answer is not very satisfactory.

Our aim is to slightly extend the weighted MSO logic in order to obtain classical quantitative logics such as PCTL and PCTL* as fragments. The crucial quantitative aspect of these logics is the probability of the set of infinite paths satisfying some linear time (LTL) property. We find it convenient to collect the set of paths in the weighted tree which is the unfolding of the probabilistic transition system. Hence, the models of our extended weighted MSO will be weighted finite or infinite trees.

The original weighted MSO logic on finite words [16] has been extended to various settings and in particular to finite trees [19] or infinite words [18] still with the aim of obtaining weighted versions of Büchi's and Elgot's fundamental theorems. These logics are also uncomparable with PCTL or PCTL*.

The key construction which is missing from all above mentioned weighted MSO logics is the possibility to transform a weighted formula into a boolean one, e.g., by using some threshold mechanism. Hence, this will be the main feature of our extension.

Our weighted MSO logic is based on a (finite) vocabulary $\mathcal{C}$ of symbols $\bowtie \in \mathcal{C}$ with $\text{arity}(\bowtie) \in \mathbb{N}$. We always include negation $\neg$, disjunction $\vee$ and conjunction $\wedge$ in the vocabulary. We may also include the equality predicate $=$ and if the semiring $\mathbb{K}$ is ordered we may use the *less than* predicate $<$. Each symbol $\bowtie \in \mathcal{C}$ is given a semantics $[\![\bowtie]\!] : K^{\text{arity}(\bowtie)} \to K$. To comply with the original weighted MSO, we interpret disjunction as addition $[\![\vee]\!] = \oplus$ and conjunction as multiplication $[\![\wedge]\!] = \otimes$. Depending on the semiring, the semantics of negation may be only partially defined. In any case, it is at least defined on $\mathbf{0}$ and $\mathbf{1}$ and exchanges these two values: $[\![\neg]\!](\mathbf{0}) = \mathbf{1}$ and $[\![\neg]\!](\mathbf{1}) = \mathbf{0}$. For the probabilistic semiring, we may define negation on the interval $[0,1]$ by $[\![\neg]\!](k) = 1 - k$ or we can even make it totally defined with $[\![\neg]\!](k) = \max(0, 1 - k)$.

Definition 5. *The syntax of our weighted MSO logic is given by the grammar*

$$\begin{aligned}\varphi ::= k \mid \kappa(x) \mid P_a(x) \mid x \leq y \mid x \in X \mid \\ \mid \bowtie(\varphi_1, \ldots, \varphi_{\text{arity}(\bowtie)}) \mid \exists x.\varphi \mid \exists X.\varphi \mid \forall x.\varphi \mid \forall X.\varphi\end{aligned}$$

where $k \in K$, $a \in \Sigma$, x, y are first-order variables, X is a set variable and $\bowtie \in \mathcal{C}$. We denote by $\text{MSO}(\mathbb{K}, \Sigma, \mathcal{C})$ the collection of all such formulas.

The original weighted MSO introduced in [16] is the fragment with $\mathcal{C} = \{\vee, \wedge\}$ and which does not use $\kappa(x)$. In our extension, the semantics of existential and universal quantifications will also be sums and products. In addition to the symbols $\bowtie \in \mathcal{C}$ whose semantics was already discussed, there is a new unary operator $\kappa(x)$ which gives the *weight* of the corresponding node in our models which are *weighted* trees.

Formally, we fix $t : D^* \rightharpoonup K \times \Sigma$ a weighted tree. Let $\mathcal{V}$ be a finite set of first-order and second-order variables. A $(\mathcal{V}, t)$-assignment σ is a function mapping

Table 1. Semantics of wMSO($\mathbb{K}, \Sigma, \mathcal{C}$)

$$
\begin{aligned}
[\![k]\!]_{\mathcal{V}}(t,\sigma) &= k \\
[\![\kappa(x)]\!]_{\mathcal{V}}(t,\sigma) &= \kappa_t(\sigma(x)) \\
[\![P_a(x)]\!]_{\mathcal{V}}(t,\sigma) &= \begin{cases} \mathbf{1} & \text{if } \ell_t(\sigma(x)) = a \\ \mathbf{0} & \text{otherwise} \end{cases} \\
[\![x \leq y]\!]_{\mathcal{V}}(t,\sigma) &= \begin{cases} \mathbf{1} & \text{if } \sigma(x) \leq \sigma(y) \\ \mathbf{0} & \text{otherwise} \end{cases} \qquad \begin{array}{l} \leq \text{ is the prefix} \\ \text{ordering on } \mathrm{dom}(t) \end{array} \\
[\![x \in X]\!]_{\mathcal{V}}(t,\sigma) &= \begin{cases} \mathbf{1} & \text{if } \sigma(x) \in \sigma(X) \\ \mathbf{0} & \text{otherwise} \end{cases} \\
[\![\bowtie(\varphi_1,\ldots,\varphi_r)]\!]_{\mathcal{V}}(t,\sigma) &= [\![\bowtie]\!]([\![\varphi_1]\!]_{\mathcal{V}}(t,\sigma),\ldots,[\![\varphi_r]\!]_{\mathcal{V}}(t,\sigma)) \qquad \text{if arity}(\bowtie) = r \\
[\![\exists x.\varphi]\!]_{\mathcal{V}}(t,\sigma) &= \bigoplus_{u \in \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(t,\sigma[x \to u]) \\
[\![\exists X.\varphi]\!]_{\mathcal{V}}(t,\sigma) &= \bigoplus_{U \subseteq \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(t,\sigma[X \to U]) \\
[\![\forall x.\varphi]\!]_{\mathcal{V}}(t,\sigma) &= \bigotimes_{u \in \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{x\}}(t,\sigma[x \to u]) \\
[\![\forall X.\varphi]\!]_{\mathcal{V}}(t,\sigma) &= \bigotimes_{U \subseteq \mathrm{dom}(t)} [\![\varphi]\!]_{\mathcal{V}\cup\{X\}}(t,\sigma[X \to U])
\end{aligned}
$$

first-order variables in $\mathcal{V}$ to elements of $\mathrm{dom}(t)$ and second-order variables in $\mathcal{V}$ to subsets of $\mathrm{dom}(t)$. If x is a first-order variable and $u \in \mathrm{dom}(t)$ then $\sigma[x \to u]$ is the $(\mathcal{V} \cup \{x\}, t)$-assignment which assigns x to u and acts like σ on all other variables. Similarly, $\sigma[X \to U]$ is defined for $U \subseteq \mathrm{dom}(t)$.

As usual, a pair (t, σ) where σ is a $(\mathcal{V}, t)$-assignment will be encoded using an extended alphabet $\Sigma_{\mathcal{V}} = \Sigma \times \{0,1\}^{\mathcal{V}}$. More precisely, we will write a weighted tree over $\Sigma_{\mathcal{V}}$ as a pair $(t,\sigma) : D^* \rightharpoonup K \times \Sigma_{\mathcal{V}}$ where t is the projection over $K \times \Sigma$ and σ is the projection over $\{0,1\}^{\mathcal{V}}$. Note that $\mathrm{dom}(t) = \mathrm{dom}(\sigma)$. Now, σ represents a *valid* assignment over $\mathcal{V}$ if for each first-order variable $x \in \mathcal{V}$, there is exactly one node $u \in \mathrm{dom}(\sigma)$ such that $\sigma(u)(x) = 1$.

Let now $\varphi \in$ wMSO($\mathbb{K}, \Sigma, \mathcal{C}$). We denote as usual by Free(φ) the set of *free* variables in φ. When Free(φ) $\subseteq \mathcal{V}$, we give in Table 1 the inductive definition of the semantics as a (partial) formal power series $[\![\varphi]\!]_{\mathcal{V}} : \mathit{Trees}(D, \mathbb{K}, \Sigma_{\mathcal{V}}) \rightharpoonup K$. We simply write $[\![\varphi]\!]$ for $[\![\varphi]\!]_{\mathrm{Free}(\varphi)}$.

Note that the semantics of a formula may be only partially defined. This may arise in particular if the semantics of some symbol in $\mathcal{C}$ is partially defined, e.g., for negation. The other difficulty is with the semantics of existential and universal quantifications.

First, if the semiring is not commutative we have to fix some order for the products of the universal quantifications. In the sequel, we will only use commutative semirings so this is not a problem. But it is also possible to deal with non commutative products. For instance, we may use the hierarchical total ordering $\prec$ on the nodes $u \in \mathrm{dom}(t)$ for the definition of $[\![\forall x.\varphi]\!]$. With this linear order, $(\mathrm{dom}(t), \prec)$ is isomorphic to an initial segment of $(\mathbb{N}, \leq)$. Hence, the characteristic function of a subset $U \subseteq \mathrm{dom}(t)$ can be identified with a word in $\{0,1\}^{\mathrm{dom}(t)}$. So the lexicographic order on words induces a total order on the powerset of $\mathrm{dom}(t)$ which can be used to compute the product over $U \subseteq \mathrm{dom}(t)$.

Second, if the tree t is infinite, we are faced with infinite sums and infinite products. We refer to Section 2 for a discussion on when this is well-defined.

A formula is *boolean* if it only takes values in $\{\mathbf{0}, \mathbf{1}\} \subseteq K$. We call bMSO the *boolean* fragment of wMSO which consists of formulas using only constants $k \in \{\mathbf{0}, \mathbf{1}\}$ and symbols $\bowtie \in \{\neg, \wedge\}$, and which does not use $\kappa(x)$ or existential quantifications. It is easy to see that each formula $\varphi \in$ bMSO takes only values in $\{\mathbf{0}, \mathbf{1}\}$ and for these formulas, the weighted semantics in $\mathbb{K}$ corresponds to the classical boolean semantics in $\mathbb{B}$. For convenience, we introduce macros for the boolean versions of disjunction and existential quantifications:

$$\varphi_1 \underline{\vee} \varphi_2 \stackrel{\mathrm{def}}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2) \qquad \underline{\exists} x.\varphi \stackrel{\mathrm{def}}{=} \neg\forall x.\neg\varphi \qquad \underline{\exists} X.\varphi \stackrel{\mathrm{def}}{=} \neg\forall X.\neg\varphi$$

These *boolean* formulas are a convenient alternative to the *unambiguous* formulas introduced in [16]. We also use a boolean version of implication which is simply defined by

$$\varphi_1 \xrightarrow{+} \varphi_2 \stackrel{\mathrm{def}}{=} \neg\varphi_1 \vee (\varphi_1 \wedge \varphi_2) \ .$$

This formula is also useful when φ_1 is boolean but not necessarily φ_2. Within a universal quantification, it allows us to compute the product of weights given by φ_2 provided φ_1 is satisfied (see Example 4 below). If φ_1 is boolean, we have

$$[\![\varphi_1 \xrightarrow{+} \varphi_2]\!]_{\mathcal{V}}(t,\sigma) = \begin{cases} [\![\varphi_2]\!]_{\mathcal{V}}(t,\sigma) & \text{if } [\![\varphi_1]\!]_{\mathcal{V}}(t,\sigma) = \mathbf{1} \\ \mathbf{1} & \text{otherwise.} \end{cases}$$

Note that the restricted form $(x \in X) \xrightarrow{+} k$ for $k \in K$ was introduced in [16, 17] for the same purpose.

Example 4. To exemplify weighted MSO, we study, as models, unfoldings of the weighted automaton $\mathcal{A}_2$ over $\mathbb{P}\mathrm{rob}$ and $\Sigma = \{a, b\}$ from Figure 2 with set of states $Q = \{q_0, q_1\}$ and set of directions $D = \Sigma \times Q$. We will, thus, define trees from $\mathit{Trees}(D, \mathbb{P}\mathrm{rob}, \Sigma)$ and formulas from $\mathrm{MSO}(\mathbb{P}\mathrm{rob}, \Sigma, \{\vee, \wedge, \neg, \leq\})$. Consider the infinite weighted tree $t_1 = t^{\mathcal{A}_2}$ as well as the finite tree $t_2 = t^{\mathcal{A}_2}_{|D^2}$ (see Figure 1). We assume that roots are always labeled with $(1, a)$.

For a start, let $\varphi_1 = \exists x.(P_b(x) \wedge (\kappa(x) > 0))$. The semantics of φ_1 is the number of nodes that carry b in their labeling and have a positive weight. Though we refer to the probabilistic semiring, formula φ_1 has therefore nothing to do with a probability. The value $[\![\varphi_1]\!](t_1)$ is not even defined, as it constitutes a non-convergent infinite sum. On the other hand, $[\![\varphi_1]\!](t_2) = 4$.

Now look at $\varphi_2 = \forall x.((P_a(x) \wedge (\kappa(x) > 0)) \xrightarrow{+} \kappa(x))$ which multiplies the positive values of all a-labeled nodes. We have $[\![\varphi_2]\!](t_1) = 0$ (it is actually an infinite product which converges to 0) and $[\![\varphi_2]\!](t_2) = \frac{4}{3^6}$. Though the semantics of φ_2 is always in the range $[0,1]$, it can hardly be interpreted as a probability. In Section 5, we will identify a syntactical fragment of MSO that is suited to speak about probabilities and accounts for the probability space of branches (paths) of a given tree. The next property will be expressible in this fragment.

Let us first assume a boolean macro formula pathto(X, x) stating that X forms a finite branch in the tree starting at the root and ending in x. We omit the precise definition, which is similar to the formula path(x, X) given in Section 5. Now consider

$$\varphi_3 = \exists X \exists x.(\mathrm{pathto}(X,x) \wedge P_b(x) \wedge \forall y.(y < x \xrightarrow{+} P_a(x)) \wedge \forall y.(y \in X \xrightarrow{+} \kappa(y)))$$

Indeed, $[\![\varphi_3]\!]$ computes the probability of the set of branches that contain at least one b. We have $[\![\varphi_3]\!](t_1) = 1$ (note that $[\![\varphi_3]\!](t_1)$ is an infinite sum whose partial sums converge to 1). Moreover, $[\![\varphi_3]\!](t_2) = \frac{5}{9}$. In Section 4, we will define a logic, called weighted CTL*, whose specialization to $\mathbb{P}\mathrm{rob}$ allows us to reason about probabilities. We will show that MSO covers this fragment, by giving corresponding formulas, which actually resemble the formula φ_3: one considers the sum over the value of paths that satisfy a given boolean property.

We show now that our weighted MSO is undecidable in general. This is obtained for the probabilistic semiring $\mathbb{P}\mathrm{rob}$ using the binary predicate *less than* even if we use unweighted words instead of weighted trees as models.

The (general) satisfiability problem is defined as follows: given a sentence $\varphi \in \mathrm{wMSO}(\mathbb{K}, \Sigma, \mathcal{C})$, does there exist a weighted tree $t \in \mathit{Trees}(D, \mathbb{K}, \Sigma)$ such that $[\![\varphi]\!](t) \neq \mathbf{0}$.

Proposition 1. *The satisfiability problem for* $\mathrm{wMSO}(\mathbb{P}\mathrm{rob}, \Sigma, \{\vee, \wedge, \neg, \leq\})$ *is undecidable.*

This result is obtained with a reduction of the emptiness problem for reactive probabilistic finite automata (rPFA). Hence, it also holds if we restrict to unweighted (finite) trees or to unweighted (finite) words.

Proof. Let $\mathcal{C} = \{\vee, \wedge, \neg, \leq\}$ and let $\mathcal{A} = (Q, q_0, \mu, F)$ be a rPFA over Σ. By [16] there is a sentence $\varphi \in \mathrm{wMSO}(\mathbb{P}\mathrm{rob}, \Sigma, \{\vee, \wedge, \neg\})$ which does not use the unary operator $\kappa(x)$ such that for all unweighted words $w \in \Sigma^*$ we have $[\![\varphi]\!](w) = [\![\mathcal{A}]\!](w)$. We may even assume that the formula φ is existential (i.e., of the form $\exists X_1 \ldots \exists X_n.\psi$) and is syntactically restricted (see [17]).

Now, let $p \in [0,1]$ and consider the weighted formula $p < \varphi$ using the binary predicate $\leq$. Then, for all unweighted words $w \in \Sigma^*$ we have $[\![p < \varphi]\!](w) \neq 0$ iff $[\![p < \varphi]\!](w) = 1$ iff $[\![\mathcal{A}]\!](w) > p$ iff the automaton $\mathcal{A}$ with threshold p accepts a nonempty language. By Theorem 1 we conclude that the satisfiability problem wrt. (unweighted) words is undecidable for $\mathrm{wMSO}(\mathbb{P}\mathrm{rob}, \Sigma, \{\vee, \wedge, \neg, \leq\})$. Since the formula φ does not use $\kappa(x)$, whether we consider weighted or unweighted words or trees does not make any difference. □

4 Weighted CTL*

We fix an *ordered* semiring $\mathbb{K}$ and a finite set *Prop* of atomic propositions. The corresponding alphabet is $\Sigma = 2^{Prop}$. As for wMSO we use a vocabulary $\mathcal{C}$ of symbols that includes $\{\neg, \vee, \wedge, \leq\}$ with the semantics given in Section 3. In our weighted CTL*, we distinguish as usual state formulas and path formulas. The path formulas are not quantitative so we call them *boolean path* formulas. The state formulas are quantitative so we use the terminology *weighted state* (or *node*) formulas. When a state formula only takes values in $\{\mathbf{0}, \mathbf{1}\}$ we call it a *boolean state* formula.

Definition 6. *The syntax of* wCTL$^*(\mathbb{K}, Prop, \mathcal{C})$ *is given by the grammar*

$$\begin{aligned} \varphi &::= k \mid \kappa \mid p \mid \bowtie(\varphi_1, \ldots, \varphi_{\mathrm{arity}(\bowtie)}) \mid \mu(\psi) \\ \psi &::= \varphi \mid \psi \wedge \psi \mid \neg\psi \mid \psi \, \mathsf{SU} \, \psi \end{aligned}$$

where $p \in Prop$, $k \in K$, $\bowtie \in \mathcal{C}$, φ *is a* weighted *state formula and* ψ *is a* boolean *path formula.*

Models for wCTL$^*(\mathbb{K}, Prop, \mathcal{C})$ are weighted trees $t \in \mathit{Trees}(D, \mathbb{K}, \Sigma)$. For weighted state formulas φ we also have to fix a node $u \in \mathrm{dom}(t)$, and the semantics $[\![\varphi]\!](t, u) \in K$ is a value in the semiring. For boolean path formulas ψ we fix both a path $w \in \mathrm{Branches}(t)$ and a node $u \leq w$ on this path, and the semantics defines whether the formula holds at node u on path w, denoted $t, w, u \models \psi$. The semantics are defined by induction on the formula: see Table 2 for weighted state formulas and Table 3 for boolean path formulas. We may also define the semantics of wCTL$^*(\mathbb{K}, Prop, \mathcal{C})$ formulas on weighted automata by using the associated unfolding which is a weighted tree.

The semantics of $\mu(\psi)$, the *measure* of ψ, is always well-defined for finite trees. But if we consider infinite trees, it may involve infinite sums or products which are not always defined. We discuss below the special case of the probabilistic semiring for which the natural semantics of $\mu(\psi)$ on infinite trees is given by the probability measure on the sequence space. In this way, we will obtain the probabilistic logics PCTL and PCTL* as framents of wCTL*.

Table 2. Semantics of weighted state formulas in wCTL$^*(\mathbb{K}, Prop, \mathcal{C})$

$$\begin{aligned} [\![k]\!](t,u) &= k \\ [\![\kappa]\!](t,u) &= \kappa_t(u) \\ [\![p]\!](t,u) &= \begin{cases} \mathbf{1} & \text{if } p \in \ell_t(u) \\ \mathbf{0} & \text{otherwise} \end{cases} \\ [\![\bowtie(\varphi_1,\ldots,\varphi_r)]\!](t,u) &= [\![\bowtie]\!]([\![\varphi_1]\!](t,u),\ldots,[\![\varphi_r]\!](t,u)) \qquad \text{if arity}(\bowtie) = r \\ [\![\mu(\psi)]\!](t,u) &= \bigoplus_{w \in \mathrm{Branches}(t) \,\mid\, t,w,u \models \psi} \quad \bigotimes_{v \,\mid\, u < v \leq w} \kappa_t(v) \end{aligned}$$

Table 3. Semantics of boolean path formulas in wCTL$^*(\mathbb{K}, \mathit{Prop}, \mathcal{C})$

$t, w, u \models \varphi$	if	$[\![\varphi]\!](t, u) \neq \mathbf{0}$
$t, w, u \models \psi_1 \wedge \psi_2$	if	$t, w, u \models \psi_1$ and $t, w, u \models \psi_2$
$t, w, u \models \neg\psi$	if	$t, w, u \not\models \psi$
$t, w, u \models \psi_1 \,\mathsf{SU}\, \psi_2$	if	$\exists u < v \leq w : (t, w, v \models \psi_2$ and $\forall u < v' < v : t, w, v' \models \psi_1)$

But first we derive some useful LTL modalities. As usual, the classical *next* and *until* modalities can be obtained from the strict until:

$$\mathsf{X}\,\psi \overset{\text{def}}{=} \mathbf{0}\,\mathsf{SU}\,\psi \qquad \psi_1\,\mathsf{U}\,\psi_2 \overset{\text{def}}{=} \psi_2 \,\underline{\vee}\, (\psi_1 \wedge (\psi_1\,\mathsf{SU}\,\psi_2))$$

When dealing with probabilistic systems such as Markov chains, it is also convenient to have a *bounded* version of until. To this purpose, logics like PCTL or PCTL* use formulas of the form $\psi_1\,\mathsf{U}^{\leq n}\,\psi_2$ for $n \in \mathbb{N}$. The semantics is that ψ_2 must hold within n time units and until then ψ_1 should hold. We may view $\mathsf{U}^{\leq n}$ or $\mathsf{SU}^{\leq n}$ as macros which are easily expressible using the next modality X.

The fragment wCTL of wCTL* consists only of state formulas and the arbitrary $\mu(\psi)$ construct of wCTL* is restricted to $\mu(\varphi_1\,\mathsf{SU}^{\leq n}\,\varphi_2)$ where φ_1 and φ_2 are (boolean) state formulas and $n \in \mathbb{N} \cup \{\infty\}$ (here $\mathsf{SU}^{\leq\infty}$ is the usual unbounded strict until SU).

Example 5. Figure 3 depicts a GPFA $\mathcal{A} = (Q, q_0, \mu)$ over $\Sigma = 2^{\mathit{Prop}}$ with $\mathit{Prop} = \{p, r\}$, and the initial part of its (infinite) unfolding $t = t^{\mathcal{A}}$. In both pictures, transitions and, respectively, nodes that carry the weight 0 are omitted. Moreover, inside every node $u \in D^*$ of t we have written the state reached by the corresponding path.

Consider the quantitative state formula $\varphi = \mu(1\,\mathsf{SU}\,r) \in$ wCTL. Table 2 allows us to compute the semantics of φ for finite trees. For instance, $[\![\varphi]\!](t_{|D^3}) = \frac{19}{27}$. Note that the boolean formula $\varphi > \frac{4}{9}$ is contained in the fragment PCTL defined below.

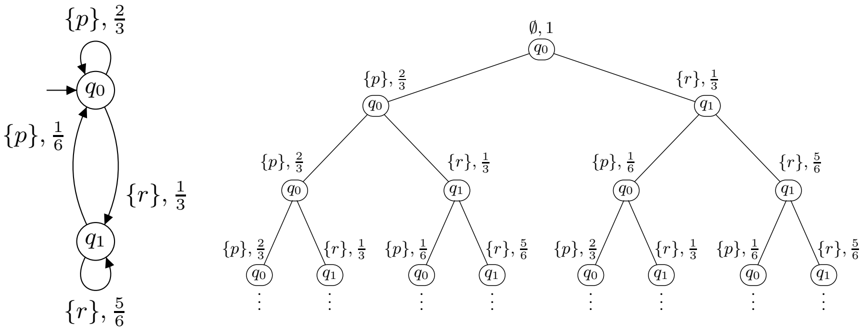


Fig. 3. GPFA $\mathcal{A} = (Q, q_0, \mu)$ over $\Sigma = 2^{\{p,r\}}$ and its unfolding $t^{\mathcal{A},q_0}$

For $n \geq 1$, consider the formula $\psi^n = \mu(\mathsf{X}^n(\mu(\mathsf{X}\,p) < \mu(\mathsf{X}\,r))) > \frac{4}{9}$ from wCTL$^*(\mathbb{P}\text{rob}, \mathit{Prop}, \{\neg, \wedge, \leq\})$. Formula ψ^n is neither in wCTL nor in PCTL* (defined below). Again, Tables (2,3) allow us to compute the semantics on finite trees. For instance, the boolean formula $\mu(\mathsf{X}\,p) < \mu(\mathsf{X}\,r)$ holds precisely in state q_1. We can check that for $n < m$ we have $[\![\psi^n]\!](t_{|D^m}) = 1$ iff $n \geq 2$.

Finally, using the semantics from Table 2, it is not clear how to compute $[\![\varphi]\!](t)$ since we have to deal with an infinite sum of infinite products. A possibility is to set $[\![\varphi]\!](t) = 0$ since the infinite products all converge to 0. But this is not the desired semantics which should measure the probability that r eventually holds. Hence, we should obtain $[\![\varphi]\!](t) = 1$.

Therefore, we extend below the semantics to infinite trees in a suitable way. For finite trees, however, both semantics coincide.

We restrict to the probabilistic semiring $\mathbb{P}$rob and to trees that are unfoldings of generative probabilistic finite automata (gPFA). More precisely, we consider a gPFA $\mathcal{A} = (Q, \mu)$ over $\Sigma = 2^{\mathit{Prop}}$ (the initial state will be fixed later and final states are irrelevant in the following). Usually, atomic propositions are associated with states. Here they are associated with transitions which is a minor difference as already noticed in Definition 4. We also assume that there are no deadlock, i.e., $\sum_{(a,q') \in \Sigma \times Q} \mu(q, a, q') = 1$ for all $q \in Q$. This is not a restriction since we may always add a sink state.

Let $t^{\mathcal{A},q} \in \mathit{Trees}(D, \mathbb{P}\text{rob}, \Sigma)$ be the full tree over $D = \Sigma \times Q$ obtained by unfolding the gPFA $\mathcal{A}$ with initial state $q \in Q$ (see definition in Section 2). Since $\mathcal{A}$ is fixed, we simply write $t^{\mathcal{A},q} = t^q$. As they arise from a *finite state* probabilistic system $\mathcal{A}$ the trees t^q are regular. More precisely, for $q \in Q$ and $u \in D^*$, we define $\text{last}(q, u) = q$ if $u = \varepsilon$ and $\text{last}(q, u) = q'$ if $u \in D^*(\Sigma \times \{q'\})$. It is easy to check that the subtree t^q_u of t^q rooted at u is in fact $t^{\text{last}(q,u)}$.

The *sequence probability space* associated with $\mathcal{A}$ and initial state $q \in Q$ is $(D^\omega, \mathfrak{B}, \text{prob}^q)$ where the Borel field $\mathfrak{B}$ is generated by the basic cylinders sets uD^ω with $u \in D^*$ and prob^q is the unique probability measure such that for a basic cylinder uD^ω we obtain the probability of the finite path described by u: if $u = (a_1, q_1)(a_2, q_2) \cdots (a_n, q_n)$ and with $q_0 = q$ then

$$\text{prob}^q(uD^\omega) = \prod_{i=1}^{n} \mu(q_{i-1}, a_i, q_i) = \prod_{v \in \text{Pref}(u)} \kappa_{t^q}(u) \ .$$

Any ω-regular set $L \subseteq D^\omega$ is measurable [37]. More precisely, any ω-regular language is a finite boolean combination of languages at the second level of the Borel hierarchy. This is a consequence of McNaughton's theorem showing that ω-regular languages can be accepted by *deterministic* Muller automata. Hence, they are finite boolean combinations of languages accepted by *deterministic* Büchi automata. Let F be the accepting set of states of a *deterministic* Büchi automaton $\mathcal{B}$ and for $n \in \mathbb{N}$, let L_n be the set of *finite* words whose run on $\mathcal{B}$ visits F at least n times. Then the language accepted by $\mathcal{B}$ is

$$\bigcap_{n\geq 0} \bigcup_{w\in L_n} wD^\omega$$

By the very definition of *strict until*, one sees that every LTL formula ψ is first-order definable, hence defines an ω-regular language $\mathcal{L}(\psi)$ by [9]. Antoher argument is to use classical translations from LTL formulas to Büchi automata. It follows that $\mathcal{L}(\psi)$ is measurable and $\mathrm{prob}^q(\mathcal{L}(\psi))$ is well-defined for LTL formulas ψ. Hence, we will be able to define the semantics of $\mu(\psi)$ using the probability measure of the sequence space.

Formally, let $q \in Q$, $u \in D^*$ and ψ be a boolean path formula. We define

$$\mathcal{L}^q_u(\psi) = \{w \in uD^\omega \mid t^q, w, u \models \psi\}$$

and

$$[\![\mu(\psi)]\!](t^q, u) = \mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L}^q_u(\psi))$$

where $u^{-1}L = \{v \in D^\infty \mid uv \in L\}$ is the left quotient of $L \subseteq D^\infty$ by $u \in D^*$. Recall that $\mathrm{last}(q, \varepsilon) = q$ and $\mathrm{last}(q, u) = q'$ if $u \in D^*(\Sigma \times \{q'\})$.

Using the remarks above, we can show by induction on the state formulas φ and the boolean path formulas ψ in wCTL*($\mathbb{P}$rob, *Prop*, $\mathcal{C}$) that for all $q \in Q$ and $u \in D^*$, $[\![\varphi]\!](t^q, u)$ only depends on $\mathrm{last}(q, u)$, and $\mathcal{L}^q_u(\psi)$ is measurable. Hence, the semantics of $\mu(\psi)$ is well-defined.

Example 6. Let us continue the discussion started in Example 5. Using the probabilistic semantics defined above for infinite trees, we obtain now

$$[\![\mu(1 \mathsf{SU}\, r)]\!](t, \varepsilon) = \mathrm{prob}^{q_0}(D^*(\{\{r\}, \{p, r\}\} \times Q)D^\omega) = 1$$

which is the probability that r eventually holds.

The probabilistic computation tree logic PCTL* [15] is a *boolean* fragment of wCTL*($\mathbb{P}$rob, *Prop*, $\{\neg, \wedge, \leq\}$) using the semantics defined above for $\mu(\psi)$. The restriction is on *state* formulas which

- use only constants $k \in \{\mathbf{0}, \mathbf{1}\}$ and symbols $\bowtie \in \{\neg, \wedge\}$,
- which do not use κ,
- and use $\mu(\psi)$ only with comparisons of the form $(\mu(\psi) \bowtie p)$ with $\bowtie \in \{\geq, >\}$, $p \in [0, 1]$, and ψ a path formula.

A further restriction is PCTL introduced in [24] where only state formulas are considered. Here the path formulas ψ used in $(\mu(\psi) \bowtie p)$ are restricted to be of the form $\varphi_1 \mathsf{SU}^{\leq n} \varphi_2$ where φ_1, φ_2 are boolean state formulas and $n \in \mathbb{N} \cup \{\infty\}$. Hence, PCTL is also a fragment of wCTL. Note that, if φ_1, φ_2 are boolean state formulas and $n \in \mathbb{N} \cup \{\infty\}$ then

$$[\![\mu(\varphi_1 \mathsf{U}^{\leq n} \varphi_2)]\!] = [\![\varphi_2 \vee (\varphi_1 \wedge \neg\varphi_2 \wedge \mu(\varphi_1 \mathsf{SU}^{\leq n} \varphi_2))]\!] \ .$$

Table 4. Translation in bMSO of boolean path formulas in wCTL*

$$\underline{\varphi}(x,X) = (\overline{\varphi}(x) \neq \mathbf{0})$$

$$\underline{\psi_1 \wedge \psi_2}(x,X) = \underline{\psi_1}(x,X) \wedge \underline{\psi_2}(x,X)$$

$$\underline{\neg\psi}(x,X) = \neg\underline{\psi}(x,X)$$

$$\underline{\psi_1 \,\mathsf{SU}\, \psi_2}(x,X) = \underline{\exists}\, z.(z \in X \wedge x < z \wedge \underline{\psi_2}(z,X) \wedge \forall y.((x < y < z) \xrightarrow{+} \underline{\psi_1}(y,X)))$$

5 wCTL* Is a Fragment of wMSO

In this section, we will give a translation from wCTL* formulas to weighted MSO formulas.

We start with path formulas ψ in wCTL*. Implicitely, such a formula has two free variables, the path (branch) on which the formula is evaluated and the current node on this path. Naturally, the current node is a first-order variable and the path in the tree can be described by the set variable consisting of all nodes on this branch. So we associate with each path formula $\psi \in \mathrm{wCTL}^*(\mathbb{K}, \mathit{Prop}, \mathcal{C})$ a boolean MSO formula $\underline{\psi}(x,X) \in \mathrm{bMSO}(\mathbb{K}, \Sigma, \mathcal{C})$. The definition by induction on the formula is given in Table 4. In this definition, we assume that the interpretation of X is indeed a path. We make sure to define *boolean* formulas by using $\underline{\exists}$, $\underline{\vee}$ and $\xrightarrow{+}$ which were defined in Section 3.

Next, we turn to (weighted) state formulas $\varphi \in \mathrm{wCTL}^*$. Here, the only implicit free variable is the current node. Hence, we associate with each state formula $\varphi \in \mathrm{wCTL}^*(\mathbb{K}, \mathit{Prop}, \mathcal{C})$ a weighted MSO formula $\overline{\varphi}(x) \in \mathrm{bMSO}(\mathbb{K}, \Sigma, \mathcal{C})$. The translation, which is indeed by induction on the formula, is given in Table 5. The *boolean* formula $\mathrm{path}(x,X)$ states that X is a *maximal* path in the tree starting from node x. To this aim, we use the boolean formula $y \lessdot z$ which holds if z is a successor (son) of y in the tree:

$$y \lessdot z \overset{\mathrm{def}}{=} y < z \wedge \forall z'.\neg(y < z' < z) \ .$$

The translation of $\mu(\psi)$ given in Table 5 is valid when the models are *finite* trees. For infinite trees, the set of paths is usually infinite and the semantics of $\overline{\mu(\psi)}$ would involve infinite products and sums that are not necessarily defined.

As explained in Section 4, it is crucial for applications to probabilistic systems to be able to deal with infinite trees that arise as unfoldings of gPFA's. So we give below a translation of $\mu(\psi)$ to wMSO such that the infinite sums and products involved in the semantics are always well-defined.

We deal with the fragment wCTL where the path formulas are restricted to be of the form $\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2$ where φ_1 and φ_2 are boolean state formulas and $n \in \mathbb{N} \cup \{\infty\}$. The translation in wMSO of $\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)$ is given in Table 6. The boolean formula $\mathrm{path}^{\leq\infty}(x,X)$ states that X is a path starting from x but do not impose that X is maximal (contrary to the definition in Table 5). When $n \in \mathbb{N}$, the boolean formula $\mathrm{path}^{\leq n}(x,X)$ requires in addition that the path X is of length at most n. Assuming that X is a path starting from x, the boolean

Table 5. Translation in bMSO of boolean path formulas in wCTL*

$$\begin{aligned}
\overline{k}(x) &= k \\
\overline{\kappa}(x) &= \kappa(x) \\
\overline{p}(x) &= \neg \bigwedge_{a \in \Sigma \mid p \in a} \neg P_a(x) \\
\overline{\bowtie(\varphi_1, \ldots, \varphi_r)}(x) &= \bowtie(\overline{\varphi_1}(x), \ldots, \overline{\varphi_r}(x)) \qquad \text{if } \mathrm{arity}(\bowtie) = r \\
\overline{\mu(\psi)}(x) &= \exists X.(\mathrm{path}(x, X) \wedge \underline{\psi}(x, X) \wedge \xi(x, X)) \\
\mathrm{path}(x, X) &= x \in X \\
&\quad \wedge \forall z.(z \in X \xrightarrow{+} (z = x \,\underline{\vee}\, \underline{\exists} y.(y \in X \wedge y \lessdot z))) \\
&\quad \wedge \neg \underline{\exists} y, z, z' \in X.(y \lessdot z \wedge y \lessdot z' \wedge z \neq z') \\
&\quad \wedge \forall y.(\ (y \in X \wedge \underline{\exists} z.(y < z)) \xrightarrow{+} \underline{\exists} z.(z \in X \wedge y < z)\ ) \\
\xi(x, X) &= \forall y.((y \in X \wedge x < y) \xrightarrow{+} \kappa(y))
\end{aligned}$$

formula $\underline{\psi}(x, X)$ states that the path satisfies $\varphi_1 \,\mathsf{SU}\, \varphi_2$ and is minimal with this property. In particular, such a path must be finite, even if $n = \infty$, since the formula $\neg(\mathbf{0}\,\mathsf{SU}\,\mathbf{1})$ means that there is no next state. Finally, the formula $\xi(x, X)$ computes the probability of the path.

Proposition 2. *Let $\mathcal{A} = (Q, \mu)$ be a* GPFA*, $q \in Q$. Let t^q be the tree unfolding of $\mathcal{A}$ starting from state q and let $u \in D^* = \mathrm{dom}(t^q)$ be a node. Let $\varphi_1, \varphi_2 \in$* wCTL($\mathbb{P}$rob, *Prop*, $\mathcal{C}$) *be boolean state formulas and $n \in \mathbb{N} \cup \{\infty\}$. Then,*

$$\begin{aligned}
\left[\!\left[\overline{\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)} \right]\!\right](t^q, u) &= \mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L}_u^q(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)) \\
&= [\![\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)]\!](t^q, u)
\end{aligned}$$

The tree t^q is infinite. Hence to prove this proposition we have to make sense of the infinite products and sums that arise from the semantics of the wMSO formula $\overline{\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)}$ given in Table 6. So let $X \subseteq D^* = \mathrm{dom}(t^q)$. By definition of $\underline{\exists}$, $\underline{\vee}$ and $\xrightarrow{+}$, the semantics of the boolean formula $\mathrm{path}^{\leq n}(u, X) \wedge \underline{\psi}(u, X)$ uses (infinite) products of boolean values $\mathbf{0}$ and $\mathbf{1}$. Naturally, we define such (infinite) products to be $\mathbf{0}$ if at least one factor is $\mathbf{0}$ and to be $\mathbf{1}$ otherwise. Hence, the difficulty is only to make sense of the (infinite) sum associated with $\exists X$ and of the (infinite) product used for the semantics of ξ.

Proof (of Prop. 2). We use the notation introduced above. We simply denote by $\mathcal{L}$ the language $\mathcal{L}_u^q(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)$ introduced in Section 4:

$$\mathcal{L} = \{w \in uD^\omega \mid t^q, w, u \models \varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2\} \ .$$

In this special case, the probability measure $\mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L})$ is easy to compute. To this aim, we introduce the language

$$\mathcal{K} = \{w \in uD^+ \mid t^q, w, u \models (\varphi_1 \wedge \neg\varphi_2) \,\mathsf{SU}^{\leq n}\, (\varphi_2 \wedge \neg(\mathbf{0} \,\mathsf{SU}\, \mathbf{1}))\}$$

We can easily check that $\mathcal{K}$ is prefix-free: $w, w' \in \mathcal{K}$ and $w \leq w'$ implies $w = w'$. Moreover,

$$\mathcal{L} = \biguplus_{w \in \mathcal{K}} wD^\omega$$

and this countable union is disjoint so that

$$\mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L}) = \sum_{w \in \mathcal{K}} \mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}wD^\omega) = \sum_{w \in \mathcal{K}} \prod_{u < v \leq w} \kappa_{t^q}(v) \ .$$

For each $w \in uD^+$, let $X_w = \{v \in D^* \mid u \leq v \leq w\}$. Next, define the set $\mathfrak{X} = \{X_w \subseteq D^* \mid w \in \mathcal{K}\}$. One can check that $\mathfrak{X}$ is precisely the set of subsets $X \subseteq D^* = \mathrm{dom}(t^q)$ such that the formula $\mathrm{path}^{\leq n}(u, X) \wedge \underline{\psi}(u, X)$ holds:

$$[\![\mathrm{path}^{\leq n} \wedge \underline{\psi}]\!](t^q, u, X) = \begin{cases} \mathbf{1} & \text{if } X \in \mathfrak{X} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Note that the infinite product used in the semantics of $[\![\xi]\!](u, X)$ is always well-defined. Either it has only finitely many factors different from $\mathbf{1}$ (which is in particular the case when X is finite) or it converges to $\mathbf{0}$. For each $w \in \mathcal{K}$, $[\![\xi]\!](u, X_w)$ computes the probability of path X_w:

$$[\![\xi]\!](t^q, u, X_w) = \prod_{u < v \leq w} \kappa_{t^q}(v) \ .$$

For sets $X \notin \mathfrak{X}$, we have $[\![\mathrm{path}^{\leq n} \wedge \underline{\psi} \wedge \xi]\!](t^q, u, X) = \mathbf{0}$ and we obtain

$$[\![\mathrm{path}^{\leq n} \wedge \underline{\psi} \wedge \xi]\!](t^q, u, X) = \begin{cases} [\![\xi]\!](t^q, u, X) & \text{if } X \in \mathfrak{X} \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Removing $\mathbf{0}$ terms in an infinite sum, we obtain

$$\begin{aligned} \left[\!\!\left[\overline{\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)} \right]\!\!\right](t^q, u) &= [\![\exists X.(\mathrm{path}^{\leq n} \wedge \underline{\psi} \wedge \xi)]\!](t^q, u) \\ &= \sum_{X \in \mathfrak{X}} [\![\xi]\!](t^q, u, X) \\ &= \sum_{w \in \mathcal{K}} \prod_{u < v \leq w} \kappa_{t^q}(v) \\ &= \mathrm{prob}^{\mathrm{last}(q,u)}(u^{-1}\mathcal{L}) \ . \end{aligned}$$

If $n \in \mathbb{N}$ then the sets $\mathcal{K}$ and $\mathfrak{X}$ are finite and the sums above are finite. When $n = \infty$, i.e., for the natural unbounded strict until, the sets $\mathcal{K}$ and $\mathfrak{X}$ may be infinite. For instance, this is the case with formula $p \,\mathsf{SU}\, r$ evaluated on the GPFA of Figure 3 starting from state q_0 where the sets $\mathcal{K}$ and $\mathfrak{X}$ corresponds to the infinite set of paths $q_0^* q_1$. But, as a consequence of the equalities above, the infinite sums over $\mathcal{K}$ and $\mathfrak{X}$ are well-defined with value in $[0, 1]$. □

Table 6. Translation in bMSO of $\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2) \in$ wCTL for $n \in \mathbb{N} \cup \{\infty\}$

$$
\begin{aligned}
\overline{\mu(\varphi_1 \,\mathsf{SU}^{\leq n}\, \varphi_2)}(x) &= \exists X.(\mathrm{path}^{\leq n}(x,X) \wedge \underline{\psi}(x,X) \wedge \xi(x,X)) \\
\mathrm{path}^{\leq \infty}(x,X) &= x \in X \\
&\quad \wedge \forall z.(z \in X \xrightarrow{+} (z = x \,\underline{\vee}\, \underline{\exists} y.(y \in X \wedge y \lessdot z))) \\
&\quad \wedge \neg \underline{\exists} y, z, z' \in X.(y \lessdot z \wedge y \lessdot z' \wedge z \neq z') \\
\text{if } n \in \mathbb{N}, \quad \mathrm{path}^{\leq n}(x,X) &= \mathrm{path}^{\leq \infty}(x,X) \wedge \neg \underline{\exists} x_0 \ldots \underline{\exists} x_n. \\
&\qquad (x_0 \in X \wedge \cdots \wedge x_n \in X \wedge x < x_0 < x_1 < \cdots < x_n) \\
\psi &= (\varphi_1 \wedge \neg\varphi_2) \,\mathsf{SU}\, (\varphi_2 \wedge \neg(\mathbf{0} \,\mathsf{SU}\, \mathbf{1})) \\
\xi(x,X) &= \forall y.((y \in X \wedge x < y) \xrightarrow{+} \kappa(y))
\end{aligned}
$$

6 Conclusion and Open Problems

In this paper, we have introduced wMSO, a weighted version of classical MSO logic. It is interpreted over weighted trees, which naturally appear as unfoldings of weighted automata. We showed that the satisfiability problem for wMSO is undecidable. We then defined wCTL and wCTL* over weighted trees and gave transformations of these logics into wMSO. For the probabilistic interpretation of the path-quantifier operator $\mu(\psi)$, we restricted to a transformation of wCTL into wMSO formulas.

Let us mention some directions for future work. We need to identify fragments of our logic that come with a decidable satisfiability problem. A natural further step is to tackle the model-checking problem: given a weighted formula φ and a weighted automaton $\mathcal{A}$, does $[\![\varphi]\!](t^{\mathcal{A}}) \neq \mathbf{0}$ hold? To find a solution, we might borrow techniques used in the probabilistic setting for PCTL*. Moreover, the translation of wCTL* into wMSO remains to be done. For the probabilistic semantics, such a translation might be based on techniques from [13] developed for checking linear-time properties of probabilistic systems (see also [5] for an overview).

It would be worthwhile to add the notion of a *scheduler* to wMSO to consider unfoldings of (partially observable) MDPs or probabilistic Büchi automata [3,4], which are essentially rPFA with a Büchi acceptance condition.

The *expectation semiring*, defined in [20], combines probabilities with expected rewards. A transfer of our logics to this specific structure (possibly extended by a discount operator) could provide a generic framework for reward models as considered, e.g., in [23,2].

A weighted μ-calculus has been defined in [22] to be interpreted over quantitative Kripke structures. For the design of a weighted μ-calculus over branching structures, one might benefit from ideas that led to a weighted μ-calculus over words [29].

References

1. Alur, R., Dill, D.L.: A theory of timed automata. Theoretical Computer Science 126(2), 183–235 (1994)
2. Andova, S., Hermanns, H., Katoen, J.P.: Discrete-time rewards model-checked. In: Larsen, K.G., Niebert, P. (eds.) FORMATS 2003. LNCS, vol. 2791, pp. 88–104. Springer, Heidelberg (2004)
3. Baier, C., Bertrand, N., Größer, M.: On decision problems for probabilistic Büchi automata. In: Amadio, R. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 287–301. Springer, Heidelberg (2008)
4. Baier, C., Größer, M.: Recognizing ω-regular languages with probabilistic automata. In: Proceedings of LICS 2005, pp. 137–146. IEEE Computer Society Press, Los Alamitos (2005)
5. Bollig, B., Leucker, M.: Verifying qualitative properties of probabilistic programs. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 124–146. Springer, Heidelberg (2004)
6. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: Cassez, F., Jard, C. (eds.) FORMATS 2008. LNCS, vol. 5215, pp. 33–47. Springer, Heidelberg (2008)
7. Buchholz, P., Kemper, P.: Model checking for a class of weighted automata. Discrete Event Dynamic Systems (to appear, 2009)
8. Büchi, J.R.: Weak second-order arithmetic and finite automata. Z. Math. Logik Grundlagen Math. 6, 66–92 (1960)
9. Büchi, J.R.: On a decision method in restricted second order arithmetic. In: Proc. of the International Congress on Logic, Methodology and Philosophy, pp. 1–11. Standford University Press (1962)
10. Ciesinski, F., Größer, M.: On probabilistic computation tree logic. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 147–188. Springer, Heidelberg (2004)
11. Clarke, E.M., Emerson, E.A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982)
12. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. The MIT Press, Cambridge (1999)
13. Courcoubetis, C., Yannakakis, M.: The complexity of probabilistic verification. Journal of the ACM 42(4), 857–907 (1995)
14. Culik, K., Kari, J.: Image compression using weighted finite automata. Computer and Graphics 17(3), 305–313 (1993)
15. de Alfaro, L.: Formal verification of probabilistic systems. Technical report, Stanford University, PhD thesis (1998)
16. Droste, M., Gastin, P.: Weighted automata and weighted logics. Theoretical Computer Science 380(1-2), 69–86 (2007); Special issue of ICALP 2005
17. Droste, M., Gastin, P.: Weighted automata and weighted logics. In: Kuich, W., Vogler, H., Droste, M. (eds.) Handbook of Weighted Automata. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg (to appear, 2009)
18. Droste, M., Rahonis, G.: Weighted automata and weighted logics with discounting. In: Holub, J., Žďárek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 73–84. Springer, Heidelberg (2007)

19. Droste, M., Vogler, H.: Weighted tree automata and weighted logics. Theoretical Computer Science 366(3), 228–247 (2006)
20. Eisner, J.: Expectation semirings: Flexible EM for learning finite-state transducers. In: Proceedings of the ESSLLI workshop on finite-state methods in NLP (2001)
21. Elgot, C.C.: Decision problems of finite automata design and related arithmetics. Trans. Amer. Math. Soc. 98, 21–52 (1961)
22. Fischer, D., Grädel, E., Kaiser, L.: Model checking games for the quantitative μ-calculus. Theory of Computing Systems (2009); Special Issue of STACS 2008
23. Größer, M., Norman, G., Baier, C., Ciesinski, F., Kwiatkowska, M., Parker, D.: On reduction criteria for probabilistic reward models. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 309–320. Springer, Heidelberg (2006)
24. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing 6(5), 512–535 (1994)
25. Knopp, K.: Theory and Application of Infinite Series. Dover Publications, New York (1990); Republication of the second English edn. (1951)
26. Kozen, D.: Results on the propositional μ-calculus. Theoretical Computer Science 27, 333–354 (1983)
27. Kuich, W., Salomaa, A.: Semirings, Automata and Languages. Springer, Heidelberg (1985)
28. Kuich, W., Vogler, H., Droste, M. (eds.): Handbook of Weighted Automata. EATCS Monographs in Theoretical Computer Science. Springer, Heidelberg (2009)
29. Meinecke, I.: A weighted μ-calculus on words. In: Diekert, V., Nowotka, D. (eds.) DLT 2009. LNCS, vol. 5583. Springer, Heidelberg (2009)
30. Mohri, M.: Finite-state transducers in language and speech processing. Computational Linguistics 23(2), 269–311 (1997)
31. Pnueli, A.: The temporal logic of programs. In: Proceedings of FOCS 1977, pp. 46–57. IEEE Computer Society Press, Los Alamitos (1977)
32. Puterman, M.L.: Markov Decision Processes. John Wiley & Sons, Inc., New York (1994)
33. Rabin, M.O.: Probabilistic automata. Information and Control 6, 230–245 (1963)
34. Segala, R.: Probability and nondeterminism in operational models of concurrency. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 64–78. Springer, Heidelberg (2006)
35. Thomas, W.: Languages, automata and logic. In: Salomaa, A., Rozenberg, G. (eds.) Handbook of Formal Languages. Beyond Words, vol. 3, pp. 389–455. Springer, Heidelberg (1997)
36. van Glabbeek, R.J., Smolka, S.A., Steffen, B.: Reactive, generative and stratified models of probabilistic processes. Information and Computation 121(1), 59–80 (1995)
37. Vardi, M.Y.: Automatic verification of probabilistic concurrent finite-state programs. In: Proceedings of FOCS 1985, pp. 327–338. IEEE, Los Alamitos (1985)

Post Correspondence Problem and Small Dimensional Matrices

Tero Harju

Department of Mathematics
University of Turku
Turku 20014, Finland
harju@utu.fi

Abstract. This is a survey on some undecidable problems on integer matrices. The proofs of these results employ special instances, called Claus instances, of the Post Correspondence Problem. The presentation is based on the article Halava et al. “Undecidability bounds for integer matrices using Claus instances” (Internat. J. Foundations of Comput. Sci. 18, 2007, 931–948).

1 Introduction

We give a short survey on undecidable decision problems on small dimensional matrices with integer entries. We have neglected to mention here many results from literature, for recent ones see, for instance, [1,2,3,10,11,13].

We express the Post Correspondence Problem (*PCP*, for short) in its morphical form. An *instance* of the PCP is a pair (g, h) of morphisms $g, h \colon A^* \to B^*$ for finite alphabets A and B. The set of the *solutions* of an instance (g, h),

$$E(g,h) = \{w \in A^+ \mid g(w) = h(w)\}\,,$$

is also called the *equality set* of the instance.

Problem 1 (PCP). Given an instance (g, h), does it have a solution, i.e., is the equality set $E(g, h)$ nonempty?

Example 1. Let $h, g \colon A^* \to A^*$ for $A = \{a, b, c\}$ be defined by

$$\begin{aligned} g(a) &= a\,, \qquad & g(b) &= bb\,, \qquad & g(c) &= bc\,, \\ h(a) &= ab\,, \qquad & h(b) &= bb\,, \qquad & h(c) &= c\,. \end{aligned}$$

In this example, we have $E(g, h) = (b^+ \cup ab^*c)^+$.

Let $e(g, h)$ denote set of all *minimal solutions* in $E(g, h)$, i.e., those solutions w that are not proper prefixes of other solutions. In general, the equality set $E(g, h)$ is a free semigroup, if it is nonempty. Indeed, if $v, vu \in E(g, h)$ with $u \neq \varepsilon$, then also $u \in E(g, h)$, and therefore every solution of (g, h) is a unique concatenation of minimal solutions. Then $E(g, h) = e(g, h)^+$.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 39–46, 2009.

2 Small Instances

The *Busy Beaver Problem for PCP* consists of instances $(g,h) \in \mathrm{PCP}(n,k)$, where n is the *size* (of the domain alphabet A) and $k = \max_{a \in A}\{\,|g(a)|,\ |h(a)|\,\}$ is the *length* of the instance. Up to isomorphism, the set $\mathrm{PCP}(n,k)$ of instances is finite, and thus solvability is decidable for these instances.

Problem 2. For small values of n and k, compute

$$\begin{aligned} M(n,k) = \max\{m \mid \exists (g,h) \in \mathrm{PCP}(n,k) \\ \text{with a shortest solution of length } m\}. \end{aligned}$$

Example 2. Waldmann and Lorentz gave the following hard instance in $\mathrm{PCP}(3,3)$ with the shortest solution w of length $|w| = 75$:

$$\begin{aligned} &g(0) = 0, \quad g(1) = 1, \qquad g(2) = 011, \\ &h(0) = 1, \quad h(1) = 011, \quad h(2) = 0. \end{aligned}$$

see, http://www.theory.informatik.uni-kassel.de/~stamer/pcp/pcpcontest_en.html.

In 1982 Ehrenfeucht, Karhumäki and Rozenberg [9] and in 1981 Pavlenko [17], independently, proved that if the size of an instance of the PCP is two, i.e., the domain alphabet A is binary, then the PCP is decidable. Let $\mathrm{PCP}(n)$ denote the Post Correspondence Problem for instances of size n.

Theorem 1 (Ehrenfeucht et al.; Pavlenko 1981–82). $\mathrm{PCP}(2)$ *is decidable.*

Theorem 2 (Matiyasevich, Sénizergues 2005). $\mathrm{PCP}(7)$ *is undecidable.*

Problem 3. The decidability status of $\mathrm{PCP}(n)$ is open for $3 \le n \le 6$.

For the length of the instances, we have

Theorem 3 (Halava et al. 2008). *The PCP is undecidable for instances of length at most 2.*

This result is based on the undecidability of the words problem for small semigroups. The *Ceĭtin semigroup* S_7 has a presentation in five generators $\Gamma = \{a,b,c,d,e\}$ and seven relations: $S_7 = \langle a,b,c,d,e \mid R\rangle$, where R consists of

$$\begin{aligned} &ac = ca, \quad ad = da, \quad bc = cb, \quad bd = db\,, \\ &eca = ce, \quad edb = de, \quad cca = ccae\,. \end{aligned}$$

The *word problem* for S_7 is the problem to determine whether two words $u, v \in \Gamma^*$ satisfy $u = v$ in S_7, that is, whether there exists a finite sequence $u = u_1, u_2, \ldots, u_n = v$ such that for $i = 0, 1, \ldots n-1$, $u_i = x_i\alpha_i y_i$ and $u_{i+1} = x_i\beta_i y_i$, where $\alpha_i = \beta_i$ is a relation in R for each i.

Theorem 4 (Ceĭtin 1958). *The word problem is undecidable for S_7.*

One then reduces the Ceĭtin semigroup to a new semigroup where both sides of the relations $u = v$ have length at most 2.

3 Claus Instances

A *semi–Thue system* is a pair $T = (\Sigma, R)$ where R is a set of relations, called the *rules*, on words: $R \subseteq \Sigma^* \times \Sigma^*$. We write $u \to_T v$, if there are words u_1 and u_2 such that

$$u = u_1 x u_2 \quad \text{and} \quad v = u_1 y u_2 \quad \text{where} \;\; (x, y) \in R.$$

Let $\to_T^*$ be the reflexive and transitive closure of the relation $\to$.

An instance of the *individual word problem* consists of a semi-Thue system T and a word w_0 and we ask, for input words w, whether or not $w \to_T^* w_0$ holds.

Theorem 5 (Matiyasevich, Sénizergues 2005). *There exists a 3-rule semi-Thue system with undecidable individual word problem.*

The reduction from semi-Thue systems to the PCP is due to Claus [8].

For simplicity, and without restriction, we concentrate on instances (g, h) where $g, h \colon \Sigma^* \to \Gamma^*$ for the binary alphabet $\Gamma = \{a, b\}$. Let also

$$d = aba \;\text{ and }\; A = ab^2b^*a.$$

An instance (g, h) is called a *Claus instance* if $g, h \colon \Sigma^* \to (abb^*a)^*$, where

$$\Sigma = \{b_1, b_2, \ldots, b_n\}$$

and

$$\begin{aligned} h(b_i) &\in (dA)^* && \text{with } h(b_n) = dd, \\ g(b_i) &\in (Ad)^* && \text{with } g(b_1) = d \text{ and } g(b_n) \in (Ad)^+ d. \end{aligned}$$

The following lemma is straightforward, see [8,13].

Lemma 1. *Let (g, h) be a Claus instance. Then $e(g, h) \subseteq b_1 \Sigma^* b_n$.*

Theorem 6 (Claus 1980). *Suppose there exists a k-rule semi-Thue system with an undecidable individual word problem. Then the PCP is undecidable for Claus instances of size $k + 4$.*

Claus's construction. Let $T = (\Gamma, R)$ be a semi–Thue system with $\Gamma = \{a, b\}$ and $R = \{t_1, t_2, \ldots, t_k\}$ where $t_i = (u_i, v_i)$ is encoded by φ, i.e., $u_i, v_i \in A^*$. Note that a^2 is not an image of φ. Let $w, w_0 \in \{a, b\}^*$.

Let $\ell_d, r_d \colon \{a, b\}^* \to (abb^*a)^*$ be the *desynchronizing morphisms* defined by

$$\ell_d(x) = dx \quad \text{and} \quad r_d(x) = xd \qquad (x \in \{a, b\})$$

and for new letters c an e, let $\Delta = \{a, b, c, e\}$, and let

$$h, g \colon (\Delta \cup R)^* \to \{a, b\}^*$$

be defined in Table 1.

Table 1. The instance (g, h) for the semi-Thue system T

$g(x) = r_d(x)$	$h(x) = \ell_d(x)$	for both $x \in \{a, b\}$
$g(t_i) = r_d(u_i)$	$h(t_i) = \ell_d(v_i)$	for $t_i \in R$
$g(c) = d$	$h(c) = \ell_d(wa^2)$	
$g(e) = r_d(a^2 w_0)d$	$h(e) = dd$	

The instance (g, h) is a Claus instance with the prefix and suffix markers $b_1 = c$ and $b_n = e$, for $n = k + 4$. By Lemma 1, $e(g, h) \subseteq c(\Gamma \cup R)^* e$.

Now, see details in [8,13], a minimal solution of (g, h) is necessarily of the form

$$cw_1a^2w_2a^2 \cdots a^2w_me \quad \text{where } w_i \in (\Delta^* R)^* \text{ and} \qquad (1)$$
$$w_i \to_T^* w_{i+1} \text{ for } i = 1, 2, \ldots, m-1.$$

When Theorem 6 is applied to Theorem 5, we have

Theorem 7. *The PCP is undecidable for Claus instances of size* $n = 7$.

We also obtain the following stronger result [11], where we have one variable word as an input. In the presentation of this result, we use PCP in the "word format" instead of morphisms.

Theorem 8 (HHH). *Let* $\Gamma = \{a, b\}$. *There exist a word* $v_1 \in \Gamma^*$ *and six pairs*

$$(u_2, v_2), (u_3, v_3), \ldots, (u_7, v_7) \in \Gamma^* \times \Gamma^*$$

such that it is undecidable whether for a given word $u_1 \in \Gamma^*$,

$$u_1u_{i_2}u_{i_3} \ldots u_{i_m}u_7 = v_1v_{i_2}v_{i_3} \ldots v_{i_m}v_7$$

for some indices $2 \leq i_2, i_3, \ldots, i_m \leq 6$. *Moreover, the instance* (g, h), *where* $h(b_i) = u_i$ *and* $g(b_i) = v_i$ *for* $i = 1, 2, \ldots, 7$, *is a Claus instance.*

The following theorem gives a morphic form of the undecidability of the *modified PCP* on seven letters.

Theorem 9. *It is undecidable whether an instance* (g, h) *of the PCP, where* $h, g \colon \{b_1, b_2, \ldots, b_7\}^* \to \Gamma^*$, *has a solution* b_1wb_7 *with* $w \in \{b_2, b_3, \ldots, b_6\}^*$.

4 Matrix Problems

There are many simply formulated undecidable problems for sets of 3×3 integer matrices. The first of these results was proved by Paterson [16] using a clever coding techniques. In this section we shall state several problems on finitely generated matrix semigroups the proofs of undecidability of which employ Claus instances.

Let $M_1, M_2, \ldots, M_k \in \mathbb{Z}^{n \times n}$ be integer matrices, and denote by

$$\langle M_1, M_2, \ldots, M_k \rangle = \{M_{i_1}M_{i_2} \ldots M_{i_m} \mid m \geq 1 \text{ and } 1 \leq i_1, i_2, \ldots, i_m \leq k\}$$

the semigroup generated by them.

Zeros in corners. Let $\varDelta = \{a_1, a_2, a_3\}$ and $\varGamma = \{a_2, a_3\}$ be fixed alphabets. Also, let

$$\sigma(a_{i_1} a_{i_2} \cdots a_{i_k}) = \sum_{j=1}^{k} i_j 3^{k-j} \quad \text{and} \quad \sigma(\varepsilon) = 0. \tag{2}$$

Then $\sigma(uv) = 3^{|v|}\sigma(u) + \sigma(v)$. We use Paterson's morphism $\gamma \colon \varDelta^* \times \varDelta^* \to \mathbb{N}^{3\times 3}$ to represent pairs of words by nonnegative integer matrices:

$$\gamma(u, v) = \begin{pmatrix} 3^{|u|} & 0 & 0 \\ 0 & 3^{|v|} & 0 \\ \sigma(u) & \sigma(v) & 1 \end{pmatrix}. \tag{3}$$

Lemma 2. *The monoid morphism γ satisfies $\gamma(u_1 u_2, v_1 v_2) = \gamma(u_1, v_1)\gamma_1(u_2, v_2)$. It is doubly injective: if $\gamma(u_1, v_1)_{31} = \gamma(u_2, v_2)_{31}$, then $u_1 = u_2$, and if $\gamma(u_1, v_1)_{32} = \gamma(u_2, v_2)_{32}$, then $v_1 = v_2$.*

Consider then the following matrix A together with its inverse A^{-1},

$$A = \begin{pmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{and} \quad A^{-1} = \begin{pmatrix} 1 & 0 & -1 \\ -1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Define $\gamma' \colon \varDelta^* \times \varDelta^* \to \mathbb{N}^{3\times 3}$ by $\gamma'(u, v) = A\gamma(u, v)A^{-1}$. The matrices $\gamma'(u, v)$ and $\gamma(u, v)$ are similar and thus also γ' is injective. We have $\gamma'(u_1, v_1)\gamma'(u_2, v_2) = \gamma'(u_1 u_2, v_1 v_2)$. Furthermore, by the above, for all words $u, v \in \varDelta^*$, we have

$$(\gamma'(u, v))_{11} = 3^{|u|} + \sigma(u) - \sigma(v). \tag{4}$$

Using Theorem 8 on Clause instances, we have

Theorem 10 (HHH). *There is a semigroup $\mathbf{S}$ generated by six 3×3-integer matrices such that it is undecidable for matrices $N \in \mathbb{Z}^{3\times 3}$ whether there exists $M \in \mathbf{S}$ with $(NM)_{11} = 0$.*

Corollary 1. *It is undecidable for matrix semigroups $\mathbf{M}$ generated by seven 3×3-integer matrices whether $\mathbf{M}$ contains a matrix M with $M_{11} = 0$.*

Vector reachability. As an example, we give a construction in the next theorem.

Theorem 11 (HHH). *Given a semigroup $\mathbf{S} \subseteq \mathbb{Z}^{3\times 3}$ generated by 6 matrices and two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{Z}^3$. It is undecidable whether or not there exists a matrix $M \in \mathbf{S}$ such that $\mathbf{u} \cdot M = \mathbf{v}$.*

Proof (Sketch). We use the following special matrix

$$A = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

Let (g, h) be a Claus instance for $g, h\colon \{b_1, b_2, \ldots, b_7\}^* \to \Delta^*$ where $\Delta = \{a_1, a_2, a_3\}$ and let $\{a, b\} = \{a_2, a_3\}$. Let the first vector $\mathbf{u}$ be defined by

$$\mathbf{u} = (\sigma(h(b_1)), \sigma(g(b_1)), 1),$$

where σ is given in (2). Define $M_i = \gamma(h(b_i), g(b_i))$, for $i = 2, 3, \ldots, 7$, and let $\mathbf{S} = \langle M_2, \ldots, M_6, M_7 A\rangle$. Note that, for all words in Δ^*,

$$(\sigma(u_1), \sigma(v_1), 1)\gamma(u_2, v_2) = (\sigma(u_1 u_2), \sigma(v_1 v_2), 1).$$

Therefore for $w = b_{j_1} \cdots b_{j_k} \in \{b_2, \ldots, b_6\}^*$

$$\begin{aligned} &\mathbf{u} M_{j_1} \cdots M_{j_k} M_7 A = \\ &(\sigma(h(b_1 w b_7)) - \sigma(g(b_1 w b_7)), \sigma(g(b_1 w b_7)) - \sigma(h(b_1 w b_7)), 0). \end{aligned} \tag{5}$$

Now (g, h) has a solution if and only if there exists $M \in \mathbf{S}$ such that $\mathbf{u} \cdot M = (0, 0, 0)$. We omit the proof of this claim.

The mortality problem. It is a long standing open problem whether the mortality problem is decidable for finitely generated semigroups $\mathbf{S} \subseteq \mathbb{Z}^{2\times 2}$, see [19]. However, Paterson [16] showed that the case for dimension three is undecidable. From the above techniques

Theorem 12 (HHH). *There is a fixed semigroup $\mathbf{S} \subseteq \mathbb{Z}^{3\times 3}$ generated by 6 matrices such that it is undecidable for a matrix A there exists $M \in \mathbf{S}$ with $AM = 0$.*

Corollary 2. *In particular, the mortality problem (i.e., whether the zero matrix belongs to the semigroup) is undecidable for semigroups generated by seven 3×3 integer matrices.*

Using results of [6] and [4], we obtain by Theorem 12 has the following corollary.

Theorem 13. *The mortality problem is undecidable for two matrices of dimension 21.*

Freeness problem. Recall that a semigroup $\mathbf{S}$ is *free* if there exists a subset G of $\mathbf{S}$ that generates $\mathbf{S}$ freely: every element of $\mathbf{S}$ has a unique factorisation over G.

It was proved by Klarner, Birget and Satterfield [14] that the freeness problem is undecidable for finitely generated matrix semigroups $\mathbf{S} \subseteq \mathbb{N}^{3\times 3}$. The proof given in Cassaigne, Harju and Karhumäki [5] gives a better bound 18 on the number of matrices. The proof is based on instances of the *mixed PCP*.

Problem 4 (Mixed PCP). Given two morphisms $g, h\colon \Sigma^* \to \Delta^*$ determine whether there exists a word $w = a_1 \ldots a_k$ with $a_i \in \Sigma$ and $k \geq 1$, such that

$$g_1(a_1) g_2(a_2) \ldots g_k(a_k) = h_1(a_1) h_2(a_2) \ldots h_k(a_k), \tag{6}$$

where, for each i, h_i and g_i are in $\{g, h\}$ and, for some j, $h_j \neq g_j$.

Lemma 3. *The Mixed PCP is undecidable for Claus instances of size 7.*

The next proof is from [5] .

Theorem 14. *It is undecidable whether a semigroup* $\mathbf{S} \subseteq \mathbb{N}^{3\times 3}$ *generated by 14 matrices is free.*

Proof. Let (g, h) be an instance of the Mixed PCP where, without loss of generality, $g, h\colon \Sigma^* \to \Sigma^*$. Let $\mathbf{S} = \langle X\rangle$ for $X = \{(a, h(a)), \gamma(a, g(a)) \mid a \in \Sigma\}$. Let $M_1, \ldots, M_p$, $N_1, \ldots, N_q$ be in X, where $M_t = \gamma(a_{i_t}, h_{i_t}(a_{i_t}))$ and $N_s = \gamma(b_{j_s}, g_{j_s}(b_{j_s}))$ with $h_{i_t}, g_{j_s} \in \{h, g\}$ and $a_{i_t}, b_{j_s} \in \Sigma$, for $t = 1, 2, \ldots, p$ and $s = 1, 2, \ldots, q$. By the definition of γ,

$$M_1 \ldots M_p = N_1 \ldots N_q \text{ in } \mathbf{S}$$

iff $(M_1 \ldots M_p)_{3,1} = (N_1 \ldots N_q)_{3,1}$ and $(M_1 \ldots M_p)_{3,2} = (N_1 \ldots N_q)_{3,2}$, which, by injectivity of σ, is equivalent to

$$a_{i_1} \ldots a_{i_p} = b_{j_1} \ldots b_{j_q} \text{ and } h_{i_1}(a_{i_1}) \ldots h_{i_p}(a_{i_p}) = g_{j_1}(b_{j_1}) \ldots g_{j_q}(b_{j_q}).$$

Thus $\mathbf{S}$ is not free if and only if the instance (g, h) of the Mixed PCP has a solution.

Diagonal matrices. It was show by Bell and Potapov in [2] that it is undecidable whether a diagonal matrix belongs to a finitely generated semigroup $\mathbf{S} \subseteq \mathbb{Z}^{4\times 4}$. Using Claus instances the bound 30 on the number of generators was reduced to 14 in [11]. The proof employs free groups.

Finally, we mention the following result, where I_n denotes identity matrix of dimension n.

Theorem 15 (HHH). *Let* $k \in \mathbb{Z}$ *with* $|k| > 1$. *It is undecidable for the matrix* kI_4 *and a matrix semigroup* $\mathbf{R} \subseteq \mathbb{Z}^{4\times 4}$ *generated by 12 matrices whether or not* $kI_4 \in \mathbf{R}$.

The decidability status of following problems remain open.

Problem 5 (Identity matrix). Given a finitely generated semigroup $\mathbf{S} \subseteq \mathbb{Z}^{n\times n}$, does I_n belong to $\mathbf{S}$?

Problem 6 (Diagonal matrix). Given a finitely generated semigroup $\mathbf{S} \subseteq \mathbb{Z}^{n\times n}$, does there exists any diagonal matrix in $\mathbf{S}$?

References

1. Bell, P., Halava, V., Harju, T., Karhumäki, J., Potapov, I.: Matrix equations and Hilbert's tenth problem. Internat. J. Algebra and Computation 18, 1–11 (2008)
2. Bell, P., Potapov, I.: On the membership of invertible diagonal matrices. In: De Felice, C., Restivo, A. (eds.) DLT 2005. LNCS, vol. 3572, pp. 146–157. Springer, Heidelberg (2005)

3. Bell, P., Potapov, I.: On undecidability bounds for matrix decision problems. Theoret. Comput. Sci. 391, 3–13 (2008)
4. Blondel, V.D., Tsitsiklis, J.N.: When is a pair of matrices mortal? Information Processing Lett. 63, 283–286 (1997)
5. Cassaigne, J., Harju, T., Karhumäki, J.: On the undecidability of freeness of matrix semigroups, Internat. J. Algebra Comput. 9, 295–305 (1999)
6. Cassaigne, J., Karhumäki, J.: Examples of undecidable problems for 2-generator matrix semigroups. Theoret. Comput. Sci. 204, 29–34 (1998)
7. Ceitin, G.C.: Associative calculus with an unsolvable equivalence problem. Tr. Mat. Inst. Akad. Nauk 52, 172–189 (1958) (in Russian)
8. Claus, V.: Some remarks on PCP(k) and related problems. Bull. of the EATCS 12, 54–61 (1980)
9. Ehrenfeucht, A., Karhumäki, J., Rozenberg, G.: The (generalized) Post Correspondence Problem with lists consisting of two words is decidable. Theoret. Comput. Sci. 21, 119–144 (1982)
10. Halava, V., Harju, T.: On Markov's undecidability theorem for integer matrices. Semigroup Forum 75, 173–180 (2007)
11. Halava, V., Harju, T., Hirvensalo, M.: Undecidability bounds for integer matrices using Claus instances. Internat. J. Foundations of Comput. Sci. 18, 931–948 (2007)
12. Halava, V., Harju, T., Hirvensalo, M., Karhumäki, J.: Post Correspondence Problem for short words. Inform. Process. Lett. 108, 115–118 (2008)
13. Harju, T., Karhumäki, J.: Morphisms. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 439–510. Springer, Heidelberg (1997)
14. Klarner, D.A., Birget, J.-C., Satterfield, W.: On the undecidability of the freeness of integer matrix semigroups. Internat. J. Algebra Comp. 1, 223–226 (1991)
15. Matiyasevich, Y., Sénizergues, G.: Decision problems for semi-Thue systems with a few rules. Theoret. Comput. Sci. 330, 145–169 (2005)
16. Paterson, M.S.: Unsolvability in 3 × 3 matrices. Stud. Appl. Math. 49, 105–107 (1970)
17. Pavlenko, V.A.: Post combinatorial problem with two pairs of words. Dokl. Akad. Nauk. Ukr. SSR, 9–11 (1981)
18. Post, E.: A variant of a recursively unsolvable problem. Bull. Amer. Math. Soc. 52, 264–268 (1946)
19. Schultz, P.: Mortality of 2 × 2 matrices. Amer. Math. Monthly 84, 463–464 (1977)

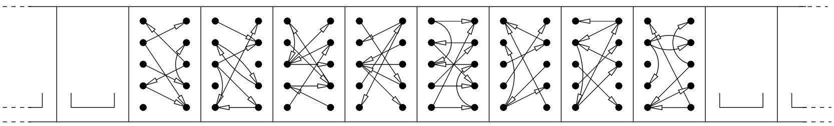
Size Complexity of Two-Way Finite Automata

Christos A. Kapoutsis

Department of Computer Science, University of Cyprus

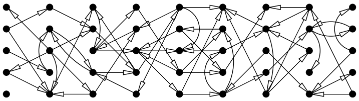
Abstract. This is a talk on the size complexity of two-way finite automata. We present the central open problem in the area, explain a motivation behind it, recall its early history, and introduce some of the concepts used in its study. We then sketch a possible future, describe a natural systematic way of pursuing it, and record some of the progress that has been achieved. We add little to what is already known—only exposition, terminology, and questions.

A problem. On the tape of a Turing machine (TM) lies an input of the form:



(1)

That is, each non-blank symbol is a two-column graph with the same, constant number of nodes per column. The question that the machine has to answer is the following: In the multi-column graph produced by identifying adjacent columns



does there exist a path from (a node of) *the leftmost column to* (a node of) *the rightmost one*? For example, in the above graph such paths exist (can you discover one?) and the answer should be "yes".

Perhaps it looks a bit strange that an entire graph fits in one tape cell. But this is not an issue. It is just that the input alphabet of this TM is a bit large: if each column has h nodes, then this alphabet consists of $2^{(2h)^2}$ symbols/graphs.

How hard is this problem? How much time/space will the TM need in order to solve it? Very little. Our problem is, in fact, regular. An easy way to prove this is to solve it with a *two-way nondeterministic finite automaton* (2NFA)—recall that 2NFAs solve exactly the regular problems [22,23,26]. Here is how:

> We start on the leftmost symbol. We nondeterministically guess a node in the left column of that symbol, and "focus" on it. From then on, we only remember which node of the current symbol we focus on, and repeat: nondeterministically guess one of the arrows out of the focused node, and make its destination the new focus. If we ever focus on a node in the rightmost column, we accept. (2)

To implement this simple algorithm, a 2NFA will need at most $2h$ states.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 47–66, 2009.

An alternative, more direct, but slightly more complicated way to prove the regularity of our problem is to solve it with a standard (*one-way*) *deterministic finite automaton* (1DFA). Here is how:

> We start on the leftmost symbol. We always move right. At each step, we remember for the nodes of the left column of the current symbol the following: (i) for each of them, whether it is reachable from the leftmost column via a path that lies entirely to our left, and (ii) for each pair (u, v) of them, whether v is reachable from u via a path that lies entirely to our left. On reaching a blank symbol, we accept iff the answer in (i) is "yes" for at least one node.

To implement this algorithm, a 1DFA will need at most 2^{h+h^2} states.

So, no matter which of the two algorithms it chooses to implement, our TM will solve the problem in linear time and zero space. Notice, however, the huge blow-up in the number of states that it is going to need if it decides not to use nondeterminism: instead of $2h$ states, it will need more than 2^{h^2}. And this is not due to lack of ingenuity in designing the deterministic algorithm: it can be proved that no other 1DFA can do with significantly fewer states. That is, the blow-up is unavoidable. So, if, e.g., each column has 16 nodes, then drawing the states of the 2NFA can be done on 1 page and in 1 minute, whereas drawing the states of the 1DFA would need more matter than we can see in the universe and would finish long after the sun has burnt out.[1] Nondeterminism wins.

But this comparison is unfair, one complains. The nondeterministic algorithm was allowed to use a two-way head, but the deterministic algorithm was not. For a fair comparison, the deterministic automaton should also be two-way; i.e., it should be a *two-way deterministic finite automaton* (2DFA). The more powerful head will probably help it solve the problem with much fewer states.

Good point. So, let's see. How would a 2DFA solve the problem? One's first attempt would probably be some kind of depth-first search inside the multi-column graph. But this doesn't work: it needs a stack of visited nodes which can grow arbitrarily large, and thus cannot fit in any finite number of states—let alone a small one. One would not give up so easily, though: sure, out-of-the-box depth-first search doesn't work, but certainly some other, cleverer version of graph exploration does. No it doesn't. To use significantly fewer states than the 1DFA, the 2DFA must do more than simply explore the graph [14]; it must use its bidirectionality both within the input [28] and at the two ends of the input [25]; and it must trace at least a linear (with respect to the input length) number of different trajectories [12].

In fact, nobody knows whether the minimum number of states in a 2DFA that solves our problem is closer to the $2h$ of the 2NFA or closer to the 2^{h^2} of the 1DFA. The best known lower bound is $\Omega(h^2)$ [4] and the best known upper bound is $2^{O(h^2)}$ [26]. At this "exponential" level of ignorance, the correct question is:

Can a 2DFA solve problem (1) with $p(h)$ states, for some polynomial p? (Q)

and is wide open. But, right now, it is probably some other question that mostly bothers you—a meta-question:

Who cares? $(\mathcal{Q})$

Determinism v Nondeterminism. A central theme in the theory of computation is the comparison between deterministic and nondeterministic computations. Most characteristic in this theme is, of course, the P v NP question, a special case of the following, more general question about the *time* used by deterministic and nondeterministic *Turing machines* (DTMs and NTMs):

Can DTMs always stay at most polynomially slower than NTMs? $(\mathrm{Q_t})$

Less prominent, but also very important, is the L v NL question, a special case of the following, more general question about the *space* used by *Turing machines*:

Can DTMs always use at most linearly more space than NTMs? $(\mathrm{Q_s})$

Despite the richness and sophistication of our theory around these questions, it is probably fair to say that our progress against their core has been slow. This has led some to suspect that *the same elusive idea may lie at the center of all problems of this kind*, little affected by the particulars of the underlying computational model and resource. If this view is correct, then a possibly advantageous approach is to study restricted models of computation.

For an extreme example, consider TMs whose heads neither turn nor write. Is nondeterminism essential there? Before pondering the question, we should specify the resource under consideration. Under these restrictions, TMs are just *one-way finite automata*. So, neither time nor space is interesting, as both 1DFAs and 1NFAs use linear time and zero space.[2] Instead, observation confirms that in this case it is the *size* of the machines, as expressed by the number of states, that reveals the nondeterministic advantage. So, the analogue to $(\mathrm{Q_t})$ and $(\mathrm{Q_s})$ is:

Can 1DFAs always stay at most polynomially larger than 1NFAs? $(\mathrm{Q_1})$

The answer is well-known to be "no" [20]. E.g., the promise problem

$$\big(\,\{\alpha i \mid \alpha\subseteq[h] \;\text{ and }\; i\in\alpha\}\,,\;\{\alpha i \mid \alpha\subseteq[h] \;\text{ and }\; i\in[h]-\alpha\}\,\big) \tag{3}$$

of checking whether a set α of numbers from 1 to h contains a number i (α and i given in this order), needs only h states on 1NFAs but at least 2^h states on 1DFAs.

Hence, in this first example, the restrictions were so strong that the resulting question was easy to answer. Backing up a bit, we may now consider TMs whose heads cannot write (but can turn). Such machines are essentially identical to *two-way finite automata*. As before, observation confirms *size* as the resource that reveals the nondeterministic advantage, and the question

Can 2DFAs always stay at most polynomially larger than 2NFAs? $(\mathrm{Q_2})$

is our new analogue to $(\mathrm{Q_t})$ and $(\mathrm{Q_s})$.

One might expect that $(\mathrm{Q_2})$ is as easy as $(\mathrm{Q_1})$. After all, it is again about finite automata. How hard can a question about finite automata be? Automata have been studied extensively since the 1950's and the answers to most interesting questions about them are already in the textbooks, right? Not really.

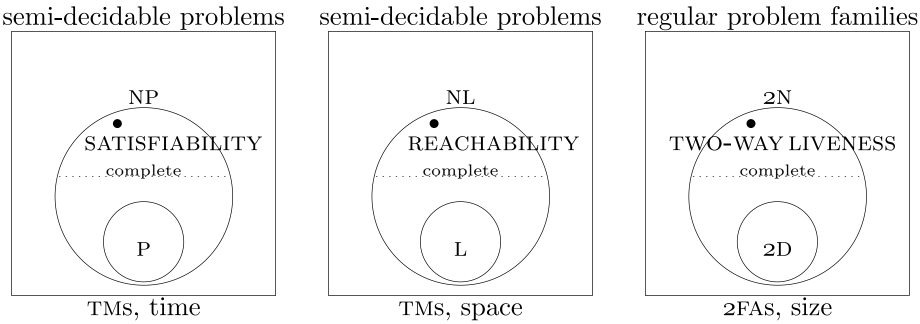


Fig. 1. An analogy between P v NP and L v NL and 2D v 2N

Such a claim may be fair only if it refers to *computability* questions about finite automata. In contrast, *complexity* questions about finite automata have been addressed only sporadically and by relatively few researchers. Many interesting and hard questions about them remain wide open. Question (Q$_2$) is one of them.

Research on (Q$_2$) is supported by an elegant theory that mirrors the theory of NP-completeness that was developed around (Q$_t$) and the theory of NL-completeness that was developed around (Q$_s$) (Fig. 1). Proposed by Sakoda and Sipser [24] in 1978, the theory starts with the class 2D of all families of regular problems that can be solved by 2DFAs of polynomially growing size

$$\text{2D} := \left\{ (L_h)_{h\geq 1} \;\middle|\; \begin{array}{l} \text{there exist 2DFAs } (M_h)_{h\geq 1} \text{ and polynomial } p \text{ such that} \\ M_h \text{ solves } L_h \text{ with at most } p(h) \text{ states, for all } h \end{array} \right\}, \quad (4)$$

and the class 2N, defined for 2NFAs in a similar manner. For example, if C_h is problem (1) when each column has h nodes, then our discussion in the previous section proves that the family $C := (C_h)_{h\geq 1}$ is in 2N, and (Q) is asking whether it is also in 2D. Moreover, the question

$$\text{2D} = \text{2N} \,? \qquad (\text{Q}')$$

is easily seen to be equivalent to (Q$_2$) in the special case of families of 2NFAs whose sizes grow polynomially.

Sakoda and Sipser went on to introduce appropriate reductions between problem families, the so-called *homomorphic reductions*, proved that 2D is closed under them, and identified a particular family in 2N that is complete with respect to them. That family was exactly C, the family whose 5th member is (1), and this is exactly how they named it—a pretty boring name, we'll call it TWO-WAY LIVENESS instead.^{3} Thus, (Q) is equivalent to (Q$'$); it is a concrete version of the 2D v 2N problem, in the same sense that the questions

Can a DTM solve SATISFIABILITY in $p(n)$ time, for some polynomial p?
Can a DTM solve REACHABILITY in $\lg\big(p(n)\big)$ space, for some polynomial p?

(where n is the input length) are concrete versions of P v NP and L v NL (Fig. 1).

So, to return to our meta-question ($\mathcal{Q}$): One reason why one may want to care about (Q) is that it can be seen as a "microscopic version" of our big questions on the power of nondeterminism, P v NP and L v NL, a question that is simultaneously complex enough to seem relevant and simple enough to seem tractable. Conceivably, by answering (Q) we might get to understand aspects of nondeterminism which are currently inaccessible through the big questions.

In addition, the connection to L v NL is more than simply conceptual. In 1977 Berman and Lingas [1] proved that, if L = NL then (in our terminology), for a polynomial p and all h, some $p(h)$-state 2DFA decides TWO-WAY LIVENESS$_h$ correctly on every $p(h)$-long input. Hence, if we can answer (Q) in the negative using only polynomially long instances, then we can also prove L $\neq$ NL—an exciting connection, which should nevertheless be received with reserve: establishing a negative answer via exponentially long strings appears to be hard already.

Much like P v NP and L v NL, most people believe that 2D $\neq$ 2N, as well.

A stronger conjecture. The possibility 2D $\neq$ 2N had actually been conjectured earlier than [24,1] and more strongly. In a 1973 manuscript [25], J. Seiferas had conjectured that sometimes a 2NFA can stay super-polynomially smaller than all 2DFAs *even without turning its head.* That is, he had conjectured that even 1NFAs can solve problems with super-polynomially fewer states than 2DFAs.

Seiferas went on to suggest a few such problems. In one of them, the input alphabet is all sets of numbers from 0 to $h-1$. E.g., if $h = 8$, then the string

$$\{1,2,4\}\emptyset\{4\}\{0,4\}\{2,4,6\}\{4\}\{4,6\}\emptyset\{3,6\}\emptyset\{2,4\}\{5,7\}\{0,3\}\{4,7\}\emptyset\{4\}\emptyset\{4\}\{0,1\}\{2,5,6\}\{1\} \tag{5}$$

is an input. A substring $\alpha_0\alpha_1\cdots\alpha_l$ of sets forms a *block* if the first set contains the number of sets after it, i.e., if $\alpha_0 \ni l$. The question is: *Can the input be separated into blocks*? E.g., the answer for (5) is "yes" because of the separation

$$\{1,\mathbf{2},4\}\emptyset\{4\}\quad\{0,\mathbf{4}\}\{2,4,6\}\{4\}\{4,6\}\emptyset\quad\{\mathbf{3},6\}\emptyset\{2,4\}\{5,7\}\quad\{\mathbf{0},3\}\quad\{4,\mathbf{7}\}\emptyset\{4\}\emptyset\{4\}\{0,1\}\{2,5,6\}\{1\}$$

where indeed the first set in each substring contains the number of sets after it in the substring, as indicated by boldface. In contrast, the answer for the string $\{1,2,7\}\{4\}\{5,6\}\emptyset\{3,6\}\{2,4,6\}$ is "no", as there is (easily) no way to break it into blocks. Seiferas called the set of all separable strings L_h—another boring name, we'll call it SEPARABILITY$_h$ instead, and let SEPARABILITY := (SEPARABILITY$_h$)$_{h\geq1}$.

Solving this problem nondeterministically is straightforward and cheap. A 1NFA can implement the following algorithm with only h states:

> We scan the input from left to right. At the start of each block, we read the first set. If it is empty, we just hang (in this nondeterministic branch). Otherwise, we nondeterministically select from it the correct number of remaining sets in the block. We then consume as many sets, counting from that number down. When the count reaches 0, we know the block is over and a new one will follow. In the end, we accept if the input and our last count-down finish together. (6)

Seiferas conjectured that, in contrast, no 2DFA can solve SEPARABILITY$_h$ with $p(h)$ states, for any polynomial p.[4]

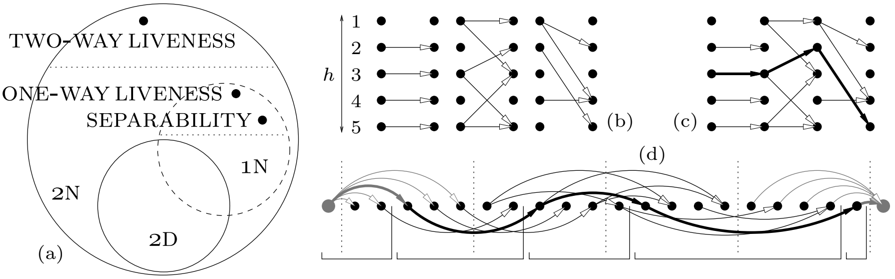


Fig. 2. (a) 1N in the map of 2D v 2N. (b) Symbols from the alphabet of ONE-WAY LIVENESS$_5$; (c) the multi-level graph they define; in bold: a live path; (d) the same graph, "flattened" for the purposes of the reduction to SEPARABILITY$_9$; dashed vertical lines distinguish the four columns; in grey: the extra nodes and arrows; in bold: a path connecting the extra nodes; also shown: the blocks defined by this path.

Sakoda and Sipser agreed with this stronger conjecture. In their terminology, this could be written as 2D $\not\supseteq$ 1N, where the class 1N is defined as in (4) but for 1NFAs (Fig. 2a). They also identified a problem family that is 1N-complete with respect to homomorphic reductions: the restriction of TWO-WAY LIVENESS to symbols/graphs with only left-to-right arrows. They called that restriction B—names were really boring in the 70's, we'll call it ONE-WAY LIVENESS (Fig. 2bc). Of course, completeness implied that 2D $\not\supseteq$ 1N $\iff$ ONE-WAY LIVENESS $\notin$ 2D. Hence, unlike Seiferas' witness, which was proposed based only on intuition, theirs was guaranteed to confirm the conjecture iff the conjecture was true.

Still, Seiferas' intuitively suggested candidate turned out to be 1N-complete, as well [24].[5] We already saw why it is in 1N (Alg. 6), so let us also see why it is 1N-hard. For this, it is enough to homomorphically reduce ONE-WAY LIVENESS to SEPARABILITY. This means (see [24] for the formal details) to provide a systematic way g of replacing each symbol a from the alphabet of ONE-WAY LIVENESS$_h$ and the endmarkers $\vdash$, $\dashv$ with a string $g(a)$ over the alphabet of SEPARABILITY$_{q(h)}$ so that, for each instance $w = a_1 \cdots a_l$ of ONE-WAY LIVENESS$_h$, performing all replacements preserves membership across the problems:

$$w \in \text{ONE-WAY LIVENESS}_h \iff g(\vdash)g(a_1)\cdots g(a_l)g(\dashv) \in \text{SEPARABILITY}_{q(h)}\,;$$

here, q must be a polynomial. What g should we use? Here is an idea.

> Consider any multi-level graph (Fig. 2c). Imagine "flattening" it by toppling each column to the left, so that its topmost node becomes leftmost (Fig. 2d). Then, an arrow from the ith node of a column to the jth node of the next column spans $2 + (h - i) + (j - 1)$ nodes: its source, its target, all nodes below/after its source, and all nodes above/before its target. We add to this graph two extra nodes, one on the left, pointing at all nodes of the leftmost column, and one on the right, pointed at by all nodes of the rightmost column.
>
> Clearly, the resulting graph contains a path connecting the two extra nodes iff the original graph contains a live path. Moreover, every such path naturally

> separates the nodes into groups: each arrow in the path defines the group of all nodes spanned by it, except for its target (Fig. 2d).
>
> We are now ready to produce an instance of separability. We replace each node u with a set of numbers that describe the arrows departing from u. Each arrow is described by the number of nodes between u and its target. E.g., in Fig. 2d, the bold arrow out of the left extra node is described by 2, the extra node itself is replaced by $\{0, 1, 2, 3, 4\}$, and the full graph gives rise to the instance
>
> $$\{0,1,\mathbf{2},3,4\}\emptyset\{4\}\{\mathbf{4}\}\{4\}\{4\}\{4,6\}\emptyset\{\mathbf{3},6\}\emptyset\{2,4\}\{5,7\}\{\mathbf{7}\}\emptyset\{4\}\emptyset\{4\}\{3\}\{2\}\{1\}\{\mathbf{0}\}\,.$$
>
> (No set describes the right extra node.) Notice the numbers in bold: they represent the bold arrows of Fig. 2d; and they separate the sets into blocks, in the same way that the bold arrows separate the nodes into groups!

Formally, for each symbol/graph a, we define $g(a) := \alpha_1\alpha_2 \cdots \alpha_h$ where

$$\alpha_i := \left\{(h-i)+(j-1) \;\middle|\; \begin{array}{l} a \text{ contains an arrow from the } i\text{th node of the} \\ \text{left column to the } j\text{th node of the right column} \end{array}\right\};$$

we also set $g(\vdash) := \{0, 1, \ldots, h-1\}$ and $g(\dashv) := \{h-1\}\cdots\{1\}\{0\}$. All numbers involved are from 0 to $2h-2$, so the result is an instance of SEPARABILITY$_{2h-1}$, so $q(h) = 2h-1$. A careful proof can easily be extracted from these observations.

Most attempts to prove 2D $\neq$ 2N have actually focused on confirming this stronger conjecture. The lower bounds mentioned in the introductory section for TWO-WAY LIVENESS [14,28,25,12] were actually proved for ONE-WAY LIVENESS.

A pause. The first goal of this talk so far has been to acquaint the reader with the study of the size complexity of two-way finite automata: the central open question in the area, a motivation behind it, some history, some terminology. Has this goal been achieved for you? Test yourself by answering the following questions: What is 1D? How does it compare to 1N? How does it relate to SEPARABILITY? How does it relate to TWO-WAY LIVENESS?[6] (for the answers)

The second goal of this talk so far has been to convince with examples that the important word in its title is not "Automata", but "Complexity". Automata theory is strongly associated with *computability questions* ("Can such-and-such an automaton recognize language such-and-such?") and with *formal language theory.* In contrast, this talk examines *complexity questions* ("I know such-and-such an automaton can solve problem such-and-such, but how efficient can it be?") and is closer to *computational complexity theory*: we discuss "algorithms" (that happen to run on automata) and "problems" and "reductions" and "complexity classes"and "completeness". Furhtermore, we don't necessarily care about automata or size; we just use this model and resource in the hope of improving our understanding of the properties of general computation.

At this point we are also ready to address a possible complaint: How come we call this "size complexity"? Since we are counting states, shouldn't we call it "state complexity"? The best measure of the size of an automaton, goes the complaint, is the number of bits needed to write down its transition function.

For a 2NFA with σ input alphabet symbols and s states, this is $2\sigma s^2$ bits.[7] So, whenever the symbols greatly outnumber the states, "size" (as number of bits) and "number of states" are hugely different. E.g., the 2NFA implementing Alg. 2 has $2h$ states, but its size is $2 \cdot 2^{(2h)^2} \cdot (2h)^2 = 2^{\Theta(h^2)}$, already close to that of the best known equivalent 2DFA; so, 2D v 2N is about "number of states", not "size"!

This argument is misleading. To see why, let BINARY TWO-WAY LIVENESS be defined over the binary alphabet and differ from TWO-WAY LIVENESS only in that each two-column graph is now encoded in $(2h)^2$ bits. The new problem is still in 2N, again by Alg. 2—it's just that the implementation will now need $O(h^4)$ states, so as to locate the start/end of each graph and the bit representing each arrow, by counting. The new problem is also 2N-complete, by a homomorphic reduction from TWO-WAY LIVENESS—just replace each graph with its binary encoding. So, 2D v 2N is also equivalent to questions of constant alphabet, where "number of states" and "size" are polynomially related. There, removing nondeterminism causes a super-polynomial blow-up either in both measures or in neither, making it safe to use either one. In short, large alphabets are "abbreviations" that allow us to focus on the combinatorial core of a problem. They can always be replaced by small alphabets, where "number of states" and "size" are interchangeable.

So, our use of the term "size complexity" is not wrong. Moreover, it seems advantageous to prefer this term whenever our question about the least upper bound for the blow-up in the number of states is only *whether it is polynomial or not*. The term "state complexity" may then be reserved for the finer part of our studies where, after having answered the polynomiality question, we go on to find the *asymptotic behavior* of the bound or, for even greater detail, its *exact value*; then, "number of states" may behave differently from "size" (as number of bits in description) and/or other measures (e.g., "number of transitions").

A theory to develop. In sharp contrast with TM time/space complexity, where a plethora of complexity classes have been introduced and studied since the 70's [21,29], the study of 2FA size complexity has been progressing very slowly and has stayed focused mostly on 2D v 2N. Figure 3a sketches a map of some TM time complexity classes for three primary modes of computation: *determinism*, *alternation*, and *randomization*.[8] Figure 3b shows what the analogous map should be for 2FAs and size. The key to the analogy is that the time bound $f(n)$ for TMs (where n is the input length) becomes the size bound $f(h)$ for 2FAs (where h is the family index). More specifically, if $\mathcal{X}$ is a mode of computation and $\mathcal{F}$ is a class of functions, then the TM time complexity class

$$\left\{ L \;\middle|\; \begin{array}{l} \text{there exist } \mathcal{X}\text{TM } M \text{ and } f \in \mathcal{F} \text{ such that } M \text{ solves } L \text{ using} \\ \text{at most } f(n) \text{ steps, for all } n \text{ and all } n\text{-long positive instances} \end{array} \right\} \tag{7}$$

corresponds to the 2FA size complexity class

$$\left\{ (L_h)_{h\geq 1} \;\middle|\; \begin{array}{l} \text{there exist } 2\mathcal{X}\text{FAs } (M_h)_{h\geq 1} \text{ and } f \in \mathcal{F} \text{ such that} \\ M_h \text{ solves } L_h \text{ using at most } f(h) \text{ states, for all } h \end{array} \right\}. \tag{8}$$

Let's explore the similarities and differences between these two maps.

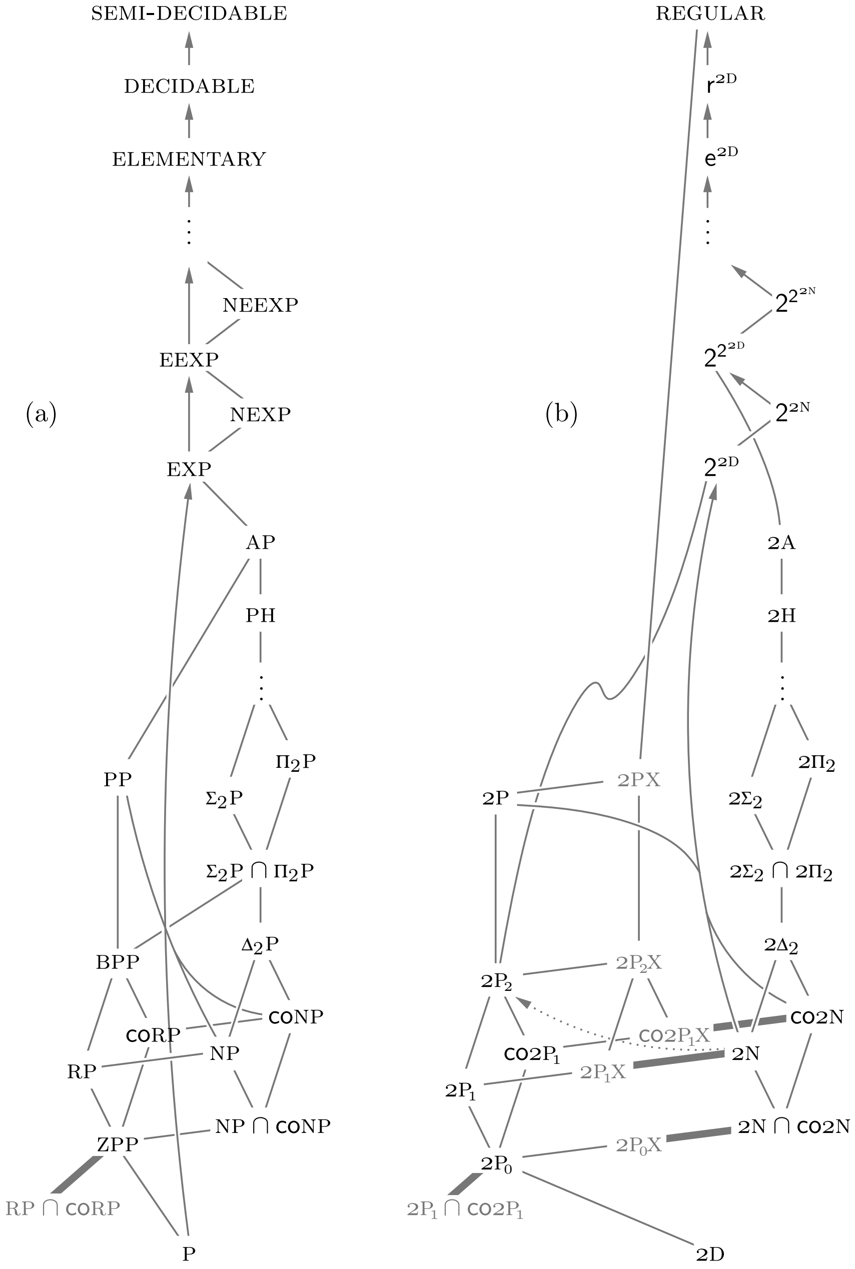


Fig. 3. (a) A map of some TM time complexity classes. (b) The corresponding 2FA size complexity map. A bold line $\mathcal{C}$—$\mathcal{C}'$ means $\mathcal{C} = \mathcal{C}'$; a simple line from $\mathcal{C}$ upwards to $\mathcal{C}'$ means $\mathcal{C} \subseteq \mathcal{C}'$; an arrow $\mathcal{C} \longrightarrow \mathcal{C}'$ means $\mathcal{C} \subsetneq \mathcal{C}'$; a dotted arrow $\mathcal{C} \dashrightarrow \mathcal{C}'$ means $\mathcal{C} \not\supseteq \mathcal{C}'$.

Determinism. If $\mathcal{X}$ is determinism and $\mathcal{F}$ is all *polynomial* functions, then (7) and (8) define P and 2D, respectively. If $\mathcal{F}$ is all *exponential* functions ($2^{p(n)}$, for polynomial p), then DTMs define EXP, while 2DFAs define the class of problem families that are solvable with at most exponentially many states—we propose the name $\mathsf{2}^{2D}$. Similarly, if $\mathcal{F}$ is all *doubly-exponential* functions, we get EEXP and $\mathsf{2}^{\mathsf{2}^{2D}}$ (again, a proposed name); and so on, for higher exponentials. When $\mathcal{F}$ becomes all *elementary* functions, we get the union of all these classes on each side; for DTMs, this is ELEMENTARY; for 2DFAs, we propose the name e^{2D}. Further up, decidable problems is the TM class when $\mathcal{F}$ is all *recursive* functions; for the corresponding 2FA class, we propose the name r^{2D}. Finally, if $\mathcal{F}$ is *all* functions, we get all semi-decidable problems and all families of regular problems.

The deterministic size complexity classes are all closed under complement—below the elementary bounds, this is non-trivial [27,9]. In addition, the well-known strict hierarchy

$$\mathrm{P} \subsetneq \mathrm{EXP} \subsetneq \mathrm{EEXP} \subsetneq \cdots \subsetneq \mathrm{ELEMENTARY} \subsetneq \mathrm{DECIDABLE} \subsetneq \mathrm{SEMI\text{-}DECIDABLE}$$

maps to a hierarchy that is also strict:

$$\mathrm{2D} \subsetneq \mathsf{2}^{2D} \subsetneq \mathsf{2}^{\mathsf{2}^{2D}} \subsetneq \cdots \subsetneq \mathsf{e}^{2D} \subsetneq \mathsf{r}^{2D} \subsetneq \mathrm{REGULAR}.$$

Inclusions are trivial.[9] Strictness follows from the fact that the minimum number of states on a 2DFA solving the unary singleton problem $\{0^x\}$ is $x+1$ [2].[10] So, for an appropriately selected f, the family $(\{0^{f(h)}\})_{h\geq 1}$ can witness any of the above differences; e.g., to show $\mathsf{e}^{2D} \subsetneq \mathsf{r}^{2D}$, just let f be recursive but non-elementary.

Alternation. If $\mathcal{X}$ is alternation and $\mathcal{F}$ is all polynomial functions, then we arrive at alternating TMs (ATMs) of polynomial time and the class AP. In studying this class, people have distinguished subclasses of problems by restricting the maximum number of runs of existential and universal steps that the ATM may perform throughout a computation. Fixing this number to a particular $i \geq 0$, we get the two classes Σ_iP and Π_iP, depending on whether the first run consists of existential or universal steps, respectively. This way, $\mathrm{P} = \Sigma_0\mathrm{P} = \Pi_0\mathrm{P}$, $\mathrm{NP} = \Sigma_1\mathrm{P}$, $\mathsf{co}\mathrm{NP} = \Pi_1\mathrm{P}$, and the infinite *polynomial-time hierarchy* rises above them, which may or may not be strict. Equivalently, one can think of Σ_iP as the problems that are solvable in polynomial time by a NTM with access to an oracle for a problem in Σ_{i-1}P, namely as $\mathrm{NP}^{\Sigma_{i-1}\mathrm{P}}$; and of Π_iP as all their complements, namely as $\mathsf{co}\mathrm{NP}^{\Sigma_{i-1}\mathrm{P}}$. Then, the class Δ_iP can also be considered, defined analogously but with a DTM, namely as $\mathrm{P}^{\Sigma_{i-1}\mathrm{P}}$. By the definitions and a few relatively easy observations, we end up with the well-known relationships:

$$\mathrm{P} \subseteq \mathrm{NP} \cap \mathsf{co}\mathrm{NP} \begin{array}{c} \subsetneq \\ \subsetneq \end{array} \begin{array}{c} \mathrm{NP} \\ \mathsf{co}\mathrm{NP} \end{array} \begin{array}{c} \subsetneq \\ \subsetneq \end{array} \Delta_2\mathrm{P} \subseteq \Sigma_2\mathrm{P} \cap \Pi_2\mathrm{P} \begin{array}{c} \subsetneq \\ \subsetneq \end{array} \begin{array}{c} \Sigma_2\mathrm{P} \\ \Pi_2\mathrm{P} \end{array} \begin{array}{c} \subsetneq \\ \subsetneq \end{array} \cdots \subseteq \mathrm{PH} \subseteq \mathrm{AP} \subseteq \mathrm{EXP} \tag{9}$$

where PH is the union of all restricted classes.

Analogous complexity classes can be considered for 2FAs and size—we propose[11] the names $2\Sigma_i$ and $2\Pi_i$ for the ith level of the hierarchy, 2H for the union of all levels, and 2A for all problem families solvable by alternating 2FAs (2AFAs) with polynomially many states. E.g., a problem family should be in $2\Sigma_i$ iff its hth member can be solved by a small ($p(h)$-state, for some polynomial p) 2AFA that performs $\leq i$ runs of existential and universal steps per computation, starting with existential ones. This way, $2\mathrm{D} = 2\Sigma_0 = 2\Pi_0$, $2\mathrm{N} = 2\Sigma_1$, and $\mathrm{co}2\mathrm{N} = 2\Pi_1$.

It should also be possible to work with the oracle-based definition. E.g., a problem family should be in the class $2\Delta_2 = 2\mathrm{D}^{2\mathrm{N}}$ if its hth member can be solved by a small 2DFA that has access to an oracle which responds to any question that can be answered by a small 2NFA executed on the same input. This way, the join TWO-WAY LIVENESS ⋈ $\overline{\text{TWO-WAY LIVENESS}}$, defined as[12]

> Given an instance w of TWO-WAY LIVENESS check that
> *either* w has even length and is live *or* it has odd length and is dead.

is in $2\Delta_2$ by the straightforward algorithm

> We scan the input once to check whether its length is even. We return to the leftmost symbol and call the oracle to check whether the input is live. If the two checks returned the same result, we accept; otherwise, we reject.

but is not known to be in $2\mathrm{N} \cup \mathrm{co}2\mathrm{N}$ (if it were, then we could disprove[13] the conjecture $2\mathrm{N} \neq \mathrm{co}2\mathrm{N}$). Similarly, one can define $2\mathrm{N}^{2\mathrm{N}}$, $\mathrm{co}2\mathrm{N}^{2\mathrm{N}}$, etc. However, some work is necessary in order to clarify these definitions: one should describe how exactly oracle calls work[14] and compare with the earlier definitions (is $2\Sigma_i = 2\mathrm{N}^{2\Sigma_{i-1}}$?). Such work is beyond the purposes of this exploratory exposition.

In the end, after appropriate fine-tuning, we should probably be able to produce a situation similar to the one in (9):

$$2\mathrm{D} \subseteq 2\mathrm{N} \cap \mathrm{co}2\mathrm{N} \begin{array}{c}\subsetneq\\ \subsetneq\end{array} \begin{array}{c}2\mathrm{N}\\ \mathrm{co}2\mathrm{N}\end{array} \begin{array}{c}\subsetneq\\ \subsetneq\end{array} 2\Delta_2 \subseteq 2\Sigma_2 \cap 2\Pi_2 \begin{array}{c}\subsetneq\\ \subsetneq\end{array} \begin{array}{c}2\Sigma_2\\ 2\Pi_2\end{array} \begin{array}{c}\subsetneq\\ \subsetneq\end{array} \cdots \subseteq 2\mathrm{H} \subseteq 2\mathrm{A} \subseteq 2^{2^{2\mathrm{D}}}$$

Note that, for a tight analogy with (9), the last inclusion should be $\mathrm{AP} \subseteq 2^{2\mathrm{D}}$. But this seems not to known. The listed inclusion follows from [18].[15]

Randomization. If $\mathcal{X}$ is randomization, we need to clarify what it means for a probabilistic TM (PTM) or probabilistic 2FA (2PFA) M to solve a problem L. Depending on what we want to model, this can be done in different ways:

two-sided error: To model all nontrivial probabilistic algorithms, we require that a *cut-point* $\lambda \in (0, 1)$ distinguishes between positive and negative instances: each $w \in L$ is accepted w.p. $> \lambda$ and each $w \notin L$ is accepted w.p. $< \lambda$. Then, the *completeness* $c(n)$ of M is the smallest acceptance probability over positive n-long instances; and the *soundness* $s(n)$ of M is the largest acceptance probability over negative n-long instances. Hence, the cut-point separates completeness and soundness: $s(n) < \lambda < c(n)$, for all n. The difference $c(n) - s(n)$ is the *isolation* of the cut-point.

two-sided error of bounded probability: To model all practical probabilistic algorithms (i.e., those allowing us to efficiently extract statistically reliable answers by sampling and majority vote), we further require that the isolation of the cut-point is significant in the length of the input: $c(n) - s(n) \geq \frac{1}{r(n)}$, for some polynomial r and all n.

one-sided error of bounded probability: To model all Monte Carlo algorithms, we further require that M never errs on negative instances, namely $s(n) = 0$.

(*"zero-sided"*) *error of zero probability*: To model all Las Vegas algorithms, we require that M always halts with either the correct answer or no answer at all and that, for all n and all n-long instances, the probability of it returning an answer is significant ($\geq \frac{1}{r(n)}$, for some polynomial r).

The TM time complexity classes that correspond to these requirements when $\mathcal{F}$ is all *polynomial* functions are PP, BPP, RP, and ZPP, respectively. For the 2FA size complexity analogues, we propose[16] the names 2P, 2P$_2$, 2P$_1$, and 2P$_0$—and call the corresponding automata 2PFAs, 2P$_2$FAs, 2P$_1$FAs, and 2P$_0$FAs.

Still, some further clarifications are necessary.

I. *Isolation*: In the bounded-error models, we require that the cut-point isolation be significant in the input length ($\geq \frac{1}{r(n)}$ for some polynomial r). This way, given the probabilistic machine and an n-long input w, one can extract from the machine a statistically reliable answer about w efficiently (in time polynomial in n). This is true irrespective of whether the machine is a standalone PTM M or a member M_h of a 2P$_2$FA-family. In the latter case, however, we must require that the isolation be significant in h as well (or else we may lose the connections to TM space complexity—via theorems à la Berman and Lingas [1]). Hence, for 2P$_2$ and 2P$_1$ we require that the cut-point isolation of M_h on n-long instances is $\geq \frac{1}{r(h,n)}$, for some polynomial r and all h, n. Similarly, for 2P$_0$ we require that M_h returns an answer w.p. $\geq \frac{1}{r(h,n)}$.

II. *Time complexity*: In contrast to deterministic and alternating 2FAs, where accepting computations are always at most linearly longer than the input, a probabilistic 2FA may very well run much slower: when finite, its expected running time may be exponential in the input length. Hence, to describe *efficient* computation, our complexity classes must also require that the expected time is polynomial in n—and also in h, for reasons similar as above. E.g., 2P$_2$ must be

$$\left\{ (L_h)_{h\geq 1} \;\middle|\; \begin{array}{l} \text{there exist 2P}_2\text{FAs } (M_h)_{h\geq 1} \text{ and polynomials } p, q \text{ such that} \\ M_h \text{ solves } L_h \text{ using at most } p(h) \text{ states and } q(h,n) \text{ steps} \\ \text{on average, for all } h \text{ and all } n \text{ and all } n\text{-long instances} \end{array} \right\},$$

and similarly for the other classes. Still, the case of polynomial size but exponential expected time is not uninteresting. To discuss such "small but slow" algorithms, we propose the names 2PX, 2P$_2$X, 2P$_1$X, and 2P$_0$X, respectively.

III. *Fineness of distributions*: A probabilistic 2FA can be *coin-flipping*, if the probability of each transition is either 0 or $\frac{1}{2}$ or 1; or *rational*, if rational transition probabilities are allowed; or *real*, if real transition probabilities are allowed. To describe *discrete* efficient computation, we must assume that our complexity classes have been defined based on automata of the first kind. Still, one can prove that every rational 2FA has an equivalent coin-flipping 2FA that is at most linearly larger and slower. Hence, redefining our classes on the basis of rational automata would not affect them. Finally, to discuss the variant classes that we get when we let all 2FAs be real, we propose the names $\mathsf{real}\text{-}2\mathrm{P}_0$, $\mathsf{real}\text{-}2\mathrm{P}_1\mathrm{X}$, etc.

IV. *Regularity*: It is easy to see that $2\mathrm{P}_0$FAs and $2\mathrm{P}_1$FAs can solve only regular problems. In contrast, $2\mathrm{P}_2$FAs can solve only regular problems iff we restrict their expected time to be polynomial [5,7], and 2PFAs can solve non-regular problems even with polynomial expected time [5]. Hence, in order to keep all members of every family in our classes regular, the definitions of $2\mathrm{P}_2\mathrm{X}$, 2P, 2PX must include the explicit requirement that "each L_h is regular".

With these clarifications, we are ready to list some known facts. First of all, the well-known relationships

$$\mathrm{P} \subseteq \mathrm{ZPP} = \mathrm{RP} \cap \mathsf{co}\mathrm{RP} \subseteq \mathrm{RP} \subseteq \mathrm{BPP} \subseteq \mathrm{PP}$$

translate directly (by the definitions and an easy fact) to the relationships

$$2\mathrm{D} \subseteq 2\mathrm{P}_0 = 2\mathrm{P}_1 \cap \mathsf{co}2\mathrm{P}_1 \subseteq 2\mathrm{P}_1 \subseteq 2\mathrm{P}_2 \subseteq 2\mathrm{P},$$

and similarly for the $*\mathrm{X}$ classes; also, we clearly have $2\mathrm{P}_0 \subseteq 2\mathrm{P}_0\mathrm{X}$, $2\mathrm{P}_1 \subseteq 2\mathrm{P}_1\mathrm{X}$, etc. Moreover, it can be proved (using the ideas of [19]) that the freedom to be slow allows Monte Carlo and Las Vegas automata to simulate nondeterminism:

$$2\mathrm{P}_1\mathrm{X} = 2\mathrm{N} \qquad \text{and thus} \qquad 2\mathrm{P}_0\mathrm{X} = 2\mathrm{P}_1\mathrm{X} \cap \mathsf{co}2\mathrm{P}_1\mathrm{X} = 2\mathrm{N} \cap \mathsf{co}2\mathrm{N}.$$

Finally, we also know (by the theorems of [5, Sect. 6]) that small & fast $2\mathrm{P}_2$FAs can be simulated by large 2DFAs, but not by small ones:[17]

$$2\mathrm{D} \subsetneq 2\mathrm{P}_2 \subseteq 2^{2\mathrm{D}}$$

but small & slow $2\mathrm{P}_2$FAs may even need non-recursively larger 2DFA simulators:

$$2\mathrm{P}_2\mathrm{X} \not\subseteq \mathsf{r}^{2\mathrm{D}}\,,$$

i.e., no recursive function can upper bound the size of the simulating 2DFAs.

Programmatic access. Although some of the open questions posed by the diagram of Fig. 3b are certainly hard, none seems to be hopeless. Moreover, each of them can be approached via three other questions of gradually decreasing difficulty: the corresponding questions for *sweeping*, *rotating*, and *one-way* automata (Fig. 4). A 2FA is sweeping (SFA: SDFA, SNFA, etc.) if its head can turn only on the end-markers, so that each computation is a series of one-way scans of alternate directions; it is rotating (RFA: RDFA, RNFA, etc.) if its head can only move right or jump from the right end-marker to the left one, so that each computation is a series of rightward scans; and it is one-way (1FA: 1DFA, 1NFA, etc.) if its head moves always right, in a single rightward scan.

So, e.g., if the full 2D v 2N problem seems hard, we can step back and study the relationship between determinism and nondetermism for SFAs first: Can SDFAs always stay at most polynomially larger than SNFAs? Or, introducing the classes SD and SN (as in (4) but for SDFAs and SNFAs), we can ask the restriction:

$$\mathrm{SD} = \mathrm{SN}\ ?$$

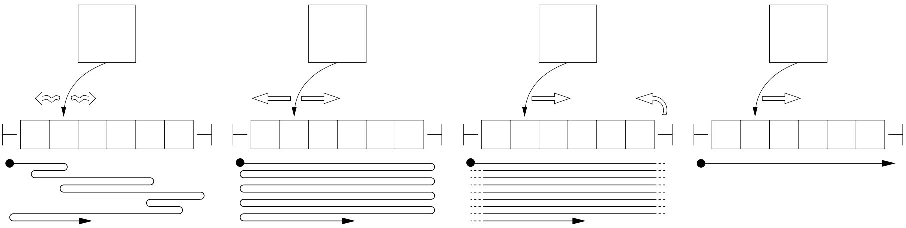


Fig. 4. The full range: *two-way*, *sweeping*, *rotating*, and *one-way* automata

If this is still hard, we can attack the even simpler question for RDFAs and RNFAs:

$$\textsc{rd} = \textsc{rn}\ ?$$

for RD and RN defined analogously. Finally, our last retreat is the one-way case:

$$\textsc{1d} = \textsc{1n}\ ?$$

These same simplifying steps can be made in the study of any relationship in Fig. 3b. Typically, solving the one-way case is indeed a lot easier than all other cases. Then, a serious boost of ideas is required for the rotating case; here, an indispensible lower-bound technique is Sipser's "generic strings" method [28], in which one studies the behavior of the automaton on inputs that are long enough to minimize a carefully chosen measure.[18] The sweeping case is then relatively easy; one just needs to carefully exploit symmetry. Finally, moving from the sweeping to the two-way case is currently beyond our reach, in general.

Another natural restriction one can focus on is that of *unary* automata. For each class $\mathcal{C}$ in Fig. 3b, one can consider the class unary-$\mathcal{C}$ that is defined identically to $\mathcal{C}$ but for unary automata. For example, one can ask:

$$\mathsf{unary}\text{-}\textsc{2d} = \mathsf{unary}\text{-}\textsc{2n}\ ?$$

Although a lot simpler, the unary case can still be highly demanding. Moving from it to the multi-symbol case is currently again beyond our reach, in general.

Some facts. When our questions are asked for the restricted models, as opposed to full-fledged 2FAs, the diagram of Fig. 3b changes as in Fig. 5. More specifically:

For SFAs (Fig. 5S), we have confirmed that nondeterminism beats determinism: $\textsc{sd} \subsetneq \textsc{sn}$ [28],[19] which directly implies $2^{\textsc{sd}} \subsetneq 2^{\textsc{sn}}$, $2^{2^{\textsc{sd}}} \subsetneq 2^{2^{\textsc{sn}}}$, etc. In fact, we even know that in the series of trivial inclusions

$$\textsc{sd} \overset{1}{\subseteq} \textsc{sp}_0 \overset{2}{\subseteq} \textsc{sp}_0\textsc{x} = \textsc{sp}_1\textsc{x} \cap \mathsf{co}\textsc{sp}_1\textsc{x} = \textsc{sn} \cap \mathsf{co}\textsc{sn} \overset{3}{\subseteq} \textsc{sn}\,,$$

both 3 and at least one of $1, 2$ are strict [15,13]. That one of $1, 2$ is strict follows from the fact that $\textsc{sd} \neq \textsc{sp}_0\textsc{x}$—i.e., slow Las Vegas behavior beats determinism [15].[20] Note that, although we can confirm the strictness of neither inclusion,

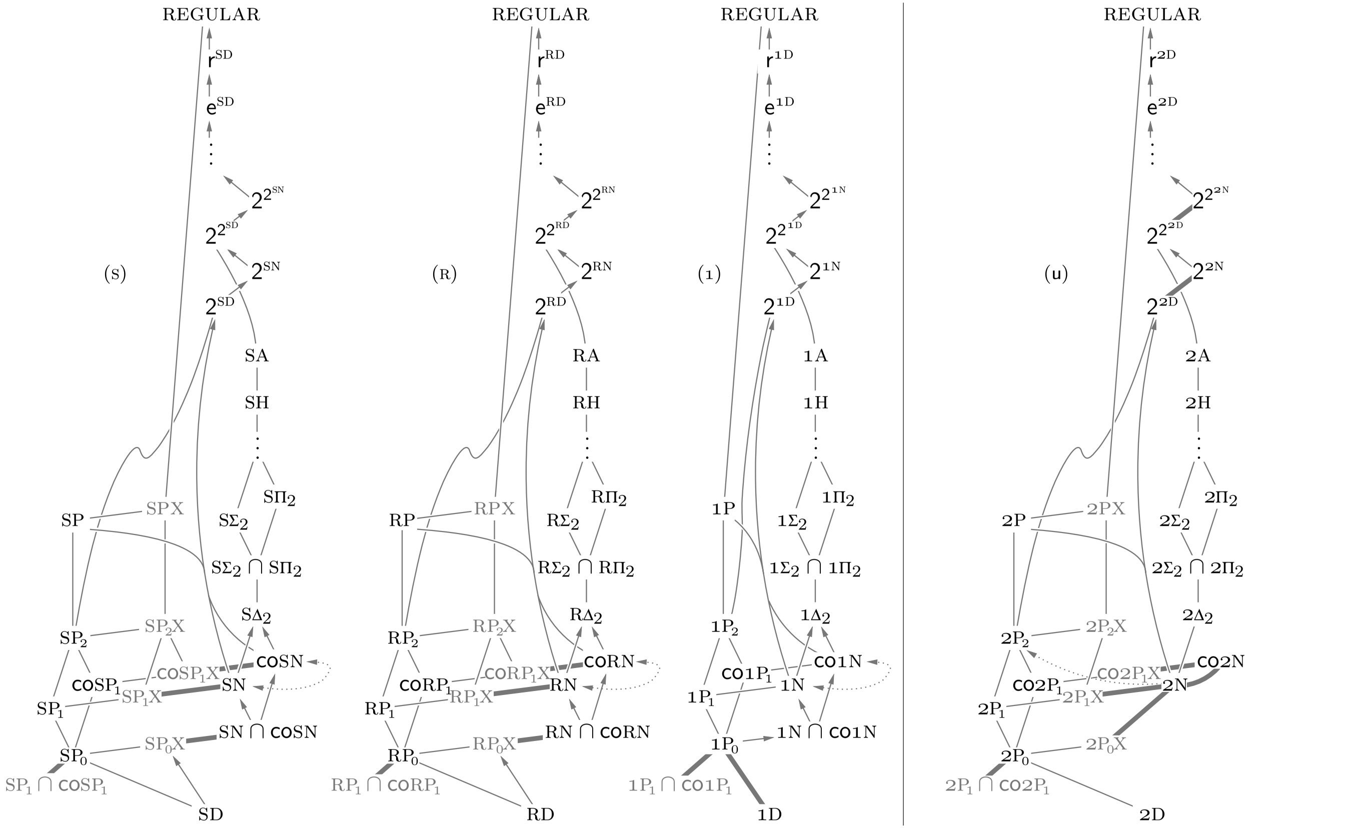


Fig. 5. The diagrams derived from that of Fig. 3b when all questions are asked not for full-fledged 2FAs, but (S) for SFAs; (R) for RFAs; (1) for 1FAs; (u) for unary 2FAs (for simplicity, we omit the unary- prefixes). The meaning of lines and arrows is as in Fig. 3.

we do know that 2 is strict in the special case where the fast SP_0FAs must run in linear expected time (as opposed to arbitrary polynomial) [17]. That inclusion 3 is strict follows from the fact that $\text{SN} \neq \text{coSN}$—i.e., nondeterminism is not closed under complement [13].[21] Note that this easily implies that $\text{SN} \cup \text{coSN} \subsetneq \text{S}\Delta_2$, as well.[22] The remaining relationships in Fig. 5s hold for the same reasons as for 2FAs. Note that there is no arrow to indicate $\text{SP}_2 \not\subseteq \text{SN}$, as the witness of [5, Thm 6.2.1] for $2\text{P}_2 \not\subseteq 2\text{N}$ needs the full bidirectionality of the $2\text{P}_2\text{FA}$.

The diagram for RFAs (Fig. 5r) is identical to that for SFAs, for essentially the same reasons. Typically, a theorem for the sweeping case comes with a proof that is stronger than the statement (as indicated in the Notes) and, in fact, implies the theorem for the rotating case. In addition, sometimes small RFAs already have all the power of small SFAs (e.g., $\text{RN} = \text{SN}$, $\text{RP}_2\text{X} = \text{SP}_2\text{X}$, $\text{RPX} = \text{SPX}$ [15]), and thus a theorem for either case implies the same for the other one.

The diagram for 1FAs (Fig. 5i) is not very different. Once again, we know that each one of the trivial inclusions $1\text{D} \subseteq 1\text{N} \cap \text{co}1\text{N} \subseteq 1\text{N}, \text{co}1\text{N} \subseteq 1\Delta_2$ is strict [24, §4.1].[23] But now, of course, there are no probabilistic classes for exponential expected time. In addition, we know that $1\text{D} = 1\text{P}_0$—i.e., Las Vegas behavior is no more powerful than determinism [11].

Finally, for unary 2FAs (Fig. 5u) important differences exist. First, a subexponential upper bound is known for the increase in size when removing nondeterminism [8]. So, starting at exponential size, nondeterminism is not essential:[24]

$$\text{unary-}2^{2\text{D}} = \text{unary-}2^{2\text{N}}, \quad \text{unary-}2^{2^{2\text{D}}} = \text{unary-}2^{2^{2\text{N}}}, \quad \text{etc.}$$

Second, nondeterminism is closed under complement [9]:

$$\text{unary-}2\text{N} = \text{unary-co}2\text{N}\,,$$

which implies that slow Las Vegas behavior is as powerful as nondeterminism:

$$\text{unary-}2\text{P}_0\text{X} = \text{unary-}2\text{P}_1\text{X} \cap \text{unary-co}2\text{P}_1\text{X} = \text{unary-}2\text{N} \cap \text{unary-co}2\text{N} = \text{unary-}2\text{N}\,,$$

Overall, the evidence in the unary case is that nondeterminism offers no significant advantage over determinism, which contrasts with what we know for the one-way, rotating, and sweeping cases and what we believe for the two-way case.

Conclusion. This has been a semi-formal talk on the size complexity of two-way finite automata. In the first half, we presented a central open problem and the main concepts in the area, explained a motivation, and recalled some early history. In the second half, we sketched where the area is heading for, if it is to mimic the development of Turing machine time/space complexity. We expressed all our statements in terms of size complexity classes (rather than the commoner "trade-off" vocabulary) and proposed names where necessary—all in continuation and in the style of the Sakoda-Sipser framework [24]. We then described how each open question may be approached via restrictions to the unary alphabet or to the sweeping, rotating, or one-way input head. Finally, we expressed in this framework some of the progress that has been achieved so far.

Our exposition has tried to be welcoming and informative, rather than rigorous or complete, and it represents this author's perspective on the subject.

The diagram of Fig. 3b remains, for the most part, unexplored: an open question lies behind any line that is not an arrow, and behind any pair of classes with no upward path between them. To a lesser but still great extent, the same is true of the diagrams of Fig. 5. A few of these questions may have already been answered—in which case this author offers his apologies for not knowing/realizing it. Other questions will be relatively easy, especially in cases where the corresponding question for TM complexity has been answered. Still, the (many) remaining questions will be hard, although certainly not impossible.

In studying these questions one will probably need to choose appropriate definitions where necessary (e.g., for oracle-2FAs), identify new complete problems (e.g., for $2\Sigma_i$, 2A), introduce new types of reductions (e.g., more powerful than the homomorphic ones), explore connections with time/space complexity (e.g., by extending the Berman-Lingas theorem [1]), add other modes of computation into the picture (e.g., interaction, the quantum mode), and more.

Much like the questions themselves, some of the ideas for answering them may come directly from answers that have already been given to corresponding questions in TM time/space complexity (e.g., inductive counting was borrowed from the proof of NL = coNL to help prove unary-2N = unary-co2N [9]). By testing these ideas in new settings, we can explore their limits and deepen our understanding of their power (e.g., inductive counting appears inadequate for showing 2N = co2N; and it will eventually prove so, if the conjecture 2N $\neq$ co2N is true). In turn, this may help us arrive at extensions or completely new techniques, hopefully advancing our understanding of TM complexity as well.

References

1. Berman, P., Lingas, A.: On complexity of regular languages in terms of finite automata. Report 304, Institute of Computer Science, Polish Academy of Sciences, Warsaw (1977)
2. Birget, J.-C.: Two-way automata and length-preserving homomorphisms. Report 109, Department of Computer Science, University of Nebraska (1990)
3. Birget, J.-C.: State-complexity of finite-state devices, state compressibility and incompressibility. Mathematical Systems Theory 26, 237–269 (1993)
4. Chrobak, M.: Finite automata and unary languages. Theoretical Computer Science 47, 149–158 (1986)
5. Dwork, C., Stockmeyer, L.J.: A time complexity gap for two-way probabilistic finite-state automata. SIAM Journal of Computing 19(6), 1011–1023 (1990)
6. Dwork, C., Stockmeyer, L.J.: Finite state verifiers I: The power of interaction. Journal of the ACM 39(4), 800–828 (1992)
7. Freivalds, R.: Probabilistic two-way machines. In: Proceedings of the International Symposium on Mathematical Foundations of Computer Science, pp. 33–45 (1981)
8. Geffert, V., Mereghetti, C., Pighizzini, G.: Converting two-way nondeterministic unary automata into simpler automata. Theoretical Computer Science 295, 189–203 (2003)

9. Geffert, V., Mereghetti, C., Pighizzini, G.: Complementing two-way finite automata. Information and Computation 205(8), 1173–1187 (2007)
10. Goldstine, J., Kappes, M., Kintala, C.M.R., Leung, H., Malcher, A., Wotschke, D.: Descriptional complexity of machines with limited resources. Journal of Universal Computer Science 8(2), 193–234 (2002)
11. Hromkovič, J., Schnitger, G.: On the power of Las Vegas for one-way communication complexity, OBDDs, and finite automata. Information and Computation 169, 284–296 (2001)
12. Hromkovič, J., Schnitger, G.: Nondeterminism versus determinism for two-way finite automata: generalizations of Sipser's separation. In: Proceedings of the International Colloquium on Automata, Languages, and Programming, pp. 439–451 (2003)
13. Kapoutsis, C.: Small sweeping 2NFAs are not closed under complement. In: Proceedings of the International Colloquium on Automata, Languages, and Programming, pp. 144–156 (2006)
14. Kapoutsis, C.: Deterministic moles cannot solve liveness. Journal of Automata, Languages and Combinatorics 12(1-2), 215–235 (2007)
15. Kapoutsis, C., Královič, R., Mömke, T.: An exponential gap between Las Vegas and deterministic sweeping finite automata. In: Proceedings of the International Symposium on Stochastic Algorithms: Foundations and Applications, pp. 130–141 (2007)
16. Kapoutsis, C., Královič, R., Mömke, T.: On the size complexity of rotating and sweeping automata. In: Proceedings of the International Conference on Developments in Language Theory, pp. 455–466 (2008)
17. Královič, R.: Infinite vs. finite space-bounded randomized computations. In: Proceedings of the IEEE Conference on Computational Complexity (to appear, 2009)
18. Ladner, R.E., Lipton, R.J., Stockmeyer, L.J.: Alternating pushdown and stack automata. SIAM Journal of Computing 13(1), 135–155 (1984)
19. Macarie, I.I., Seiferas, J.I.: Amplification of slight probabilistic advantage at absolutely no cost in space. Information Processing Letters 72(3–4), 113–118 (1999)
20. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: Proceedings of the Symposium on Switching and Automata Theory, pp. 188–191 (1971)
21. Papadimitriou, C.H.: Computational complexity. Addison-Wesley, Reading (1994)
22. Rabin, M.O.: Two-way finite automata. In: Proceedings of the Summer Institute of Symbolic Logic, Cornell, pp. 366–369 (1957)
23. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM Journal of Research and Development 3, 114–125 (1959)
24. Sakoda, W.J., Sipser, M.: Nondeterminism and the size of two-way finite automata. In: Proceedings of the Symposium on the Theory of Computing, pp. 275–286 (1978)
25. Seiferas, J.I.: Untitled manuscript. Communicated to M. Sipser (October 1973)
26. Shepherdson, J.C.: The reduction of two-way automata to one-way automata. IBM Journal of Research and Development 3, 198–200 (1959)
27. Sipser, M.: Halting space-bounded computations. Theoretical Computer Science 10, 335–338 (1980)
28. Sipser, M.: Lower bounds on the size of sweeping automata. Journal of Computer and System Sciences 21(2), 195–202 (1980)
29. Sipser, M.: Introduction to the theory of computation. PWS Publishing Company, Boston (1996)

Notes

[1] Assuming that the observable universe contains 10^{80} atoms, the sun runs out of fuel in 10 billion years, and drawing 1 state takes 1 atom and 1 picosecond.

[2] By now the reader has probably picked up our naming conventions. But, for one last time, let's just make sure: "1NFA" means *one-way nondeterministic finite automaton.*

[3] The "LIVENESS" part of the name hints at the behavior of problem instances under extension: A path from the leftmost to the rightmost column is called *live*; an instance that contains live paths is called *live*, as well; an instance that contains no live paths is called *dead.* Now think of what happens when we prepend or append extra symbols/graphs to an instance: if the instance is live, it may remain live or become dead; in contrast, if the instance is dead, it will remain dead—a most tight analogy. The "TWO-WAY" part hints at the fact that paths may grow in either direction.

[4] In fact, his conjecture was even stronger: that the minimum number of states in a 2DFA solving SEPARABILITY_h is exactly 2^h—as it is for 1DFAs.

[5] One can also prove the same for all other candidate witnesses that he proposed.

[6] By analogy to 2D (and 2N and 1N), 1D is the class of problem families that can be solved by families of 1DFAs of polynomially growing size. The formal definition is as in (4) but for 1DFAs. We have $1\text{D} \subseteq 1\text{N}$ (trivially) and $1\text{D} \neq 1\text{N}$ (e.g., the problem family implicit in (3) is in $1\text{N} - 1\text{D}$). Overall, $1\text{D} \subsetneq 1\text{N}$, and thus also $1\text{D} \subsetneq 2\text{N}$ (since $1\text{N} \subseteq 2\text{N}$). Since SEPARABILITY is 1N-complete, we know $\text{SEPARABILITY} \not\in 1\text{D}$. Since TWO-WAY LIVENESS is 2N-complete, we know $\text{TWO-WAY LIVENESS} \not\in 1\text{D}$. (Here we are using the fact that 1D is closed under homomorphic reductions.)

[7] For each direction d (left, right), input symbol a, and pair of states (p, q), we need 1 bit saying whether being at p and reading a causes the automaton to jump to q and move its head in the d direction.

[8] One could also include here one more map for TM space complexity and/or augment all maps with *interaction* [6], *parallelism*, the *quantum* mode, etc. But time complexity and the three primary modes are enough to make our point.

[9] In fact, [26] implies that even $2\text{N} \subseteq 2^{1\text{D}}$, $2^{2\text{N}} \subseteq 2^{2^{1\text{D}}}$, etc.

[10] In fact, $x + 1$ states are sufficient even for a 1DFA; and they are necessary even for a 2NFA [2, Fact 5.2].

[11] Here, we follow the Sakoda-Sipser two-symbol naming convention: one symbol for the head mode, one more for the transition function mode—as in "2D", "1N", etc.

[12] See [24, §4.1] for a similar join, witnessing that $2\text{D} \not\subseteq 1\text{N} \cup \text{co}1\text{N}$. Also, see Note 3 for what it means for an instance of TWO-WAY LIVENESS to be live/dead.

[13] *Proof*: Suppose the join is in $2\text{N} \cup \text{co}2\text{N}$. W.l.o.g., assume it is in 2N (if in co2N, work similarly but with even lengths). Let M be a small 2NFA solving the join. Using M, we can construct a small 2NFA M' solving $\overline{\text{TWO-WAY LIVENESS}}$. Here is how: We scan the input w once to check the parity of its length. If odd, we just simulate M on w—and thus end up accepting iff w is dead. If even, we simulate M on rw, where r is the two-column graph that contains all $(2h)^2$ arrows—since rw is of odd length, we end up acepting iff rw is dead, and thus iff w is dead. It should be clear that M' can indeed implement this algorithm, and thus solves $\overline{\text{TWO-WAY LIVENESS}}$ with roughly twice as many states as M. This implies $\overline{\text{TWO-WAY LIVENESS}} \in 2\text{N}$, and thus $\text{co}2\text{N} = 2\text{N}$.

[14] How does the oracle read a query? Is the query always the entire input of the 2FA, is it some portion of the input, or is it produced from the input by a small two-way transducer? How does the 2FA read the oracle's answer?

[15] In fact, [18, Thm 4.2.1] proves that even $2^{2^{1\text{D}}}$ contains 2A.

[16] Again, we follow the Sakoda-Sipser naming convention: “P” means “probabilistic” and the index counts the sides of bounded error or, if the error is unbounded, is absent.

[17] In fact, [5] proves much more: Thm 6.1 says that even real-$2P_2 \subseteq 2^{1D}$ (small & fast $2P_2$FAs can be simulated by large 2DFAs even when they are *real* and even when the 2DFAs are actually *one-way*) and Thm 6.2 says that even $2N \not\supseteq 2P_2$ (small 2DFAs cannot simulate every small & fast $2P_2$FA even if they are allowed to use nondeterminism).

[18] The method was first applied to deterministic (rotating/sweeping) automata [28], then also to nondeterministic ones [13] and to probabilistic ones [17]. For other applications to deterministic automata, see [14,15,16].

[19] In fact, [28] proves that even $SD \not\supseteq 1N$.

[20] In fact, [15] proves that even $SD \not\supseteq 1N \cap co1N$.

[21] In fact, [13] proves that even $coSN \not\supseteq 1N$.

[22] In [13], the witness for $SN \not\subseteq coSN$ is ONE-WAY LIVENESS. So, consider the join ONE-WAY LIVENESS $\bowtie$ $\overline{\text{ONE-WAY LIVENESS}}$. As in Note 13, we can easily prove that (i) the join is in SD^{SN} and (ii) if it were in $SN \cup coSN$, we would have ONE-WAY LIVENESS $\in coSN$.

[23] For the strictness of $1N \cup co1N \subseteq 1\Delta_2$, consider the join T_n used in [24, §4.1].

[24] In fact, [8] implies that even unary-$2^{SD} \supseteq$ unary-2^{2N}, unary-$2^{2^{SD}} \supseteq$ unary-$2^{2^{2N}}$, etc.

Matrix Mortality and the Černý-Pin Conjecture

Jorge Almeida[1,⋆] and Benjamin Steinberg[2,⋆⋆]

[1] Departamento de Matemática Pura, Faculdade de Ciências, Universidade do Porto,
Rua do Campo Alegre, 687, 4169-007 Porto, Portugal
[2] School of Mathematics and Statistics, Carleton University, 1125 Colonel By Drive,
Ottawa, Ontario K1S 5B6, Canada
jalmeida@fc.up.pt, bsteinbg@math.carleton.ca

Abstract. In this paper, we establish the Černý-Pin conjecture for automata with the property that their transition monoid cannot recognize the language $\{a,b\}^*ab\{a,b\}^*$. For the subclass of automata whose transition monoids have the property that each regular $\mathscr{J}$-class is a subsemigroup, we give a tight bound on lengths of reset words for synchronizing automata thereby answering a question of Volkov.

1 Introduction

In 1964 Černý conjectured that any n-state synchronizing automaton has a reset word of length at most $(n-1)^2$. Despite years of intensive work [2–6, 8, 12, 18, 19, 27, 28, 34–37, 39, 41–43], the best known upper bound is $\frac{n^3-n}{6}$, due to Pin [29] based on a non-trivial result of Frankl from extremal set theory; see also [20]. Černý, himself, showed that $(n-1)^2$ is the best one can hope for [8]. Pin generalized the conjecture as follows [28]. Suppose (Q,Σ) is an automaton such that some word $w\in\Sigma^*$ acts on Q as a transformation of rank r. Then he proposed that there should be a word of length at most $(n-r)^2$ acting as a rank r transformation. This generalized conjecture was disproved by Kari [18]. However, there is a reformulation of the Pin conjecture that is still open (and that was interpreted by Rystsov as being the Pin conjecture [35]). This conjecture is sometimes known as the Rank conjecture or the Černý-Pin conjecture. It states that if r is the minimal rank of a transformation in the transition monoid of an n-state automaton (in which case we say the automaton has rank r), then there is a word of length at most $(n-r)^2$ that acts as a transformation of rank r. The case $r=1$ is the Černý conjecture.

This paper is a contribution to this form of the Černý-Pin conjecture. To state our main result, we recall the notion of a variety of finite monoids [13]. A *variety of finite monoids* is a class of finite monoids closed under taking finite products, submonoids and homomorphic images [13]. There is a bijection

[⋆] The first author was supported, in part, by FCT through CMUP and also through the project PTDC/MAT/65481/2006, which is partly funded by the European Community Fund FEDER.

[⋆⋆] The second author was supported in part by NSERC and the DFG.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 67–80, 2009.

between varieties of finite monoids and varieties of languages [13]. Recall that the variety **DS** [1, 32], introduced by Schützenberger [38], consists of all finite monoids whose regular $\mathscr{J}$-classes are subsemigroups. The variety **EDS** consists of all monoids whose idempotents generate a submonoid belonging to **DS**. For example, this variety contains all monoids with commuting idempotents. It is known that **EDS** is the largest variety of finite monoids that does not contain the syntactic monoid of the language $\{a,b\}^*ab\{a,b\}^*$ and that a monoid belongs to **EDS** if and only if it cannot recognize the language $\{a,b\}^*ab\{a,b\}^*$ (cf. [32, Chapter 7]). We show that, for an n-state automaton of rank r whose transition monoid belongs to **EDS**, there is a word of length at most $(n-r)(n-r+1)/2$ which acts as a transformation of rank r. This bound is tight for this class since Rystsov gave an example of an n-state synchronizing automaton whose transition monoid has commuting idempotents and whose minimal length reset word has length $n(n-1)/2$ [34]. As most papers just focus on the original Černý conjecture, this result gives the widest class of monoids for which the more general Černý-Pin conjecture is known to hold.

We also give a tight bound of $n-1$ on the length of reset words for n-state synchronizing automata with transition monoid in the pseudovariety **DS** [1, 32], improving the result of [2] and answering a question of Volkov [43]. Our techniques are a continuation of the representation theoretic approach to the Černý conjecture initiated in [2, 6, 39], and also are an elaboration on an idea of Rystsov [35].

The key notion in this paper is that of a mortality function for a finite monoid S. A mortality function measures the lengths of zero words under matrix representations of S. We estimate mortality functions by reducing to the case of irreducible representations and using the theory of Munn, Ponizovsky, Rhodes and Zalcstein [9, 15, 23, 24, 33]. These results are then applied to a particular representation coming from an automaton. The paper ends with a universal mortality function that relies on the effective solution to the Burnside problem for matrix semigroups [7, 14, 17, 22, 25, 40].

A journal version of this paper is under preparation that extends the results to a much more general class of automata, which is a bit more technical to define.

2 Mortality Functions

In this paper, all monoids are assumed finite except free monoids and full matrix monoids. We use Σ^* to denote the free monoid on a set Σ. If Σ is a generating set for a monoid S, we will abuse notation and not distinguish between $w \in \Sigma^*$ and the element of S represented by w. All actions of monoids are on the right. Denote by $\mathbb{N}$ the set of positive integers. By a *representation* of a monoid S of *degree* n, we mean a monoid homomorphism $\varphi\colon S \to M_n(\mathbb{Q})$ where $M_n(\mathbb{Q})$ is the monoid of $n \times n$ matrices over the field of rational numbers $\mathbb{Q}$. If $v \in \mathbb{Q}^n$ and $s \in S$, then $v\varphi(s)$ will be abbreviated to vs.

Definition 2.1 (Mortality function). *Let S be a monoid. By a **mortality function** for S, we mean a function $f\colon \mathbb{N} \to \mathbb{N}$ such that, for all representations*

$\varphi\colon S \to M_n(\mathbb{Q})$ of degree n with $0 \in \varphi(S)$ and all generating sets Σ of S, there is a word $w \in \Sigma^*$ of length at most $f(n)$ so that $\varphi(w) = 0$.

The terminology mortality comes from [26]. The reader is referred to [10, 11] for basic notions and definitions from representation theory. Notice that a mortality function is non-decreasing since if $\varphi\colon S \to M_n(\mathbb{Q})$ is a representation with $0 \in \varphi(S)$ and $\psi\colon S \to \mathbb{Q}$ is the degree 1 representation sending the group of units of S to 1 and all other elements to 0, then $\varphi \oplus \psi$ has degree $n+1$ and contains 0, so from this it follows that $f(n) \leq f(n+1)$. Also note that $f(n) = |S| - 1$ is a mortality function for S and so we are really interested in mortality functions with "good" constants, rather than in asymptotics. Most of the time we are interested in degrees that are significantly smaller than $|S|$. Also, we want mortality functions that are valid for whole classes of monoids and not just for a single monoid. We remark that if $\varphi\colon S \to T$ is an onto homomorphism and f is a mortality function for S, then f is also a mortality function for T.

There is a connection between mortality under matrix representations and the Černý-Pin problem due to Rystsov [35]. To state the Černý-Pin conjecture, we need a few definitions. The *rank* of a transformation is the size of its image. An automaton (Q, Σ) has *rank* r if r is the minimal rank of an element of its transition monoid S. Notice that the set of elements of minimal rank in S is an ideal and hence contains the minimal ideal. We now state the Černý-Pin (or Rank) conjecture.

Conjecture 2.2 (Černý-Pin). An automaton of rank r has a word of length at most $(n-r)^2$ representing a transformation of rank r.

The Černý conjecture is the special case when $r = 1$; the general statement is a variation on a conjecture of Pin. An automaton of rank 1 is called *synchronizing* and a word representing a transformation of rank 1 is often termed a *reset word.*

A function $f\colon \mathbb{N} \to \mathbb{N}$ is called *superadditive* if, for all $m, n \in \mathbb{N}$, one has that $f(m) + f(n) \leq f(m+n)$. We are mostly interested in superadditive mortality functions. The following is a variant on a result of Rystsov [35].

Proposition 2.3. *Let (Q, Σ) be an n-state automaton of rank r with transition monoid S. Let f be a supperadditive mortality function for S. Then there is a word $w \in \Sigma^*$ of length at most $f(n-r)$ so that $|Qw| = r$.*

Proof. Without loss of generality, assume $Q = \{1, \ldots, n\}$. Linearize the action of S on Q to a matrix representation $\varphi\colon S \to M_n(\mathbb{Q})$ by setting $e_i s = e_{is}$ where $e_1, \ldots, e_n$ is the standard basis for $\mathbb{Q}^n$.

Assume first that S acts transitively on Q, that is, the automaton (Q, Σ) is strongly connected. If $X \subseteq Q$, let $\overline{X}$ denote the characteristic vector of X. Let $\mathscr{C}$ be the set of images of rank r elements of S. Notice that S acts on $\mathscr{C}$. Indeed, the elements of rank r in S form an ideal and so if $t \in S$ has rank r, then $|Qts| = r$ for all $s \in S$ and so $Qt \in \mathscr{C}$ implies $Qts \in \mathscr{C}$. Let V be the subspace of $\mathbb{Q}^n$ spanned by the elements $\overline{X} - \overline{Y}$ such that $X, Y \in \mathscr{C}$. It is easy to see that V is S-invariant. We claim that $Vs = 0$, for $s \in S$, if and only if s has rank r.

First note that if s has rank r, then for any $X \in \mathscr{C}$ one has $Xs = Qs$ since both sets have size r. Thus $(\overline{X} - \overline{Y})s = 0$. For the converse, suppose that s has rank greater than r. Let $X \in \mathscr{C}$ and choose $q \in Qs \setminus Xs$. Choose $p \in Q$ so that $ps = q$ and let $Y \in \mathscr{C}$ such that $p \in Y$. Such a Y exists as S acts transitively on Q. Then $(\overline{X} - \overline{Y})s \neq 0$ since $\overline{X}s$ has 0 in the q-coordinate while $\overline{Y}s$ has 1 in this coordinate.

Next we show that $\dim V \leq n - r$. First of all let $s \in S$ have rank r and let $P_1, \ldots, P_r$ be the equivalence classes of the kernel of s. Since these sets are disjoint, it is immediate that their characteristic vectors are linearly independent. Let W be the subspace spanned by the $\overline{P_i}$, $i = 1, \ldots, r$. Then $\dim W = r$ and hence $\dim W^{\perp} = n - r$. Suppose now that $X \in \mathscr{C}$. Since $|Xs| = |X| = r$, it follows that $|X \cap P_i| = 1$ all i. In other words, $\overline{X} \cdot \overline{P_i} = 1$ for all i. Thus if $X, Y \in \mathscr{C}$, then $\overline{X} - \overline{Y} \perp \overline{P_i}$, for all i. We conclude that $V \subseteq W^{\perp}$ and hence $\dim V \leq n - r$. The result now follows in the transitive case.

Now suppose that S does not act transitively on Q. The S-invariant subsets of Q are ordered by inclusion. Let $C_1, \ldots, C_\ell$ be the minimal S-invariant subsets of Q, that is, the minimal strongly connected components of the automaton (Q, Σ). Then the C_i are disjoint and S acts transitively on each C_i by minimality. First suppose that $Q = C_1 \cup \ldots \cup C_\ell$. Let $n_i = |C_i|$ and let r_i be the rank of (C_i, Σ). Plainly $n = n_1 + \cdots + n_\ell$ and, moreover, one easily verifies that $r = r_1 + \cdots + r_\ell$ (consider an element of the minimal ideal of S). Since the transition monoid of (C_i, Σ) is a quotient of S, it admits f as a mortality function. It now follows from the previous case that we have, for each i, a word w_i of length at most $f(n_i - r_i)$ with $|C_i w_i| = r_i$. Then $w = w_1 \cdots w_\ell$ represents a transformation of Q of rank r and the length of w is at most $\sum_{i=1}^{\ell} f(n_i - r_i) \leq f(n - r)$ by superadditivity.

Next suppose that $C = C_1 \cup \cdots \cup C_\ell \neq Q$. Since C contains all the minimal S-invariant subsets, it is easy to see that $qS \cap C \neq \emptyset$ for all $q \in Q \setminus C$. Consequently, the set $\{s \in S \mid Qs \subseteq C\}$ is a non-empty ideal of S and hence contains the minimal ideal. Thus r is also the rank of (C, Σ). Indeed, if $s \in S$ belongs to the minimal ideal, then $Qs \subseteq C$ and hence $Qs = Qs^2 \subseteq Cs$ (the equality $Qs = Qs^2$ follows from minimality of the rank of s).

Now S acts by partial transformations on $X = Q \setminus C$ by restriction; moreover, the elements of the minimal ideal of S act via the empty transformation by the above paragraph. Thus by linearizing this partial transformation action of S on X, we obtain a representation $\rho\colon S \to M_{|X|}(\mathbb{Q})$ with $0 \in \rho(S)$. Hence there is a word w of length at most $f(n - |C|)$ representing the empty transformation on X, i.e., so that $Qw \subseteq C$. Because the transition monoid of (C, Σ) is a quotient of S, it has f as a mortality function, so by the previous case there is a word u of length at most $f(|C| - r)$ so that $|Cu| = r$. Then $r \leq |Qwu| \leq |Cu| = r$ so wu represents a transformation of rank r. Since f is superadditive,

$$|wu| \leq f(n - |C|) + f(|C| - r) \leq f(n - r)$$

as required. □

It is natural to try to obtain a mortality function for S by reducing to the case of an irreducible representation: a representation of S is called *irreducible* if there are no proper, non-zero S-invariant subspaces. To do this, we need to deal with composition series.

Lemma 2.4. *Let $\varphi\colon S \to M_n(\mathbb{Q})$ be a representation and let*

$$0 = V_k \subseteq V_{k-1} \subseteq \cdots \subseteq V_0 = \mathbb{Q}^n$$

be a tower of S-invariant subspaces. Suppose that, for $i = 0, \ldots, k-1$, there are elements $s_i \in S$ with $V_i s_i \subseteq V_{i+1}$. Then $\varphi(s_0 s_1 \cdots s_{k-1}) = 0$.

Proof. Straightforward induction shows that $V_0 s_0 \cdots s_i \subseteq V_{i+1}$, from which the result follows as $V_k = 0$. □

The above lemma lets us enact our reduction to the case of irreducible representations.

Proposition 2.5. *Let $f\colon \mathbb{N} \to \mathbb{N}$ be a superadditive function and suppose that, for all irreducible representations $\rho\colon S \to M_d(\mathbb{Q})$ with $0 \in \rho(S)$ and for all generating sets Σ of S, there exists $w \in \Sigma^*$ so that $|w| \leq f(d)$ and $\rho(w) = 0$. Then f is a mortality function for S.*

Proof. Let $\varphi\colon S \to M_n(\mathbb{Q})$ be a representation so that $0 \in \varphi(S)$ and fix a generating set Σ for S. Let $0 = V_k \subseteq V_{k-1} \subseteq \cdots \subseteq V_0 = \mathbb{Q}^n$ be a composition series for $\mathbb{Q}^n$, that is, an unrefinable tower of S-invariant subspaces. Let $\rho_i\colon S \to \mathrm{End}(V_i/V_{i+1})$ be the associated irreducible representation. Since $0 \in \varphi(S)$, it follows immediately that $0 \in \rho_i(S)$. Thus, by assumption, we can find words w_i, for $0 \leq i \leq k-1$, with $V_i w_i \subseteq V_{i+1}$ and $|w_i| \leq f(d_i)$ where d_i is the dimension of V_i/V_{i+1}. Now $d_0 + \cdots + d_{k-1} = n$ and $\varphi(w_0 w_1 \cdots w_{k-1}) = 0$ by Lemma 2.4. Since f is superadditive, $|w_0 w_1 \cdots w_{k-1}| \leq f(d_0) + \cdots + f(d_{k-1}) \leq f(n)$. This completes the proof. □

It was proved in [2] that if $S \in \mathbf{DS}$, $\varphi\colon S \to M_n(\mathbb{Q})$ is an irreducible representation with $0 \in \varphi(S)$ and Σ is a generating set of S, then there is an element of Σ which is mapped to the zero matrix. Since the function $f(n) = n$ is superadditive, it now follows from Proposition 2.5 that $f(n) = n$ is a mortality function for S. Putting it all together, we obtain:

Theorem 2.6. *Let S be a monoid in $\mathbf{DS}$. Then $f(n) = n$ is a mortality function for S. Hence, if (Q, Σ) is a synchronizing automaton with transition monoid in $\mathbf{DS}$, then it has a reset word of length at most $n-1$ and moreover this bound is tight. More generally, if (Q, Σ) is an automaton of rank r with transition monoid in $\mathbf{DS}$, then there is a word $w \in \Sigma^*$ of length at most $n-r$ so that $|Qw| = r$.*

Proof. In light of Proposition 2.3, the argument before the theorem statements proves everything except the tightness. For tightness, just use an n-state counter-free automaton over a unary alphabet. □

The above theorem generalizes Rystsov's result for the case of commutative monoids [34] and answers a question raised by Volkov [43]. The following lemma is due to Rystsov [34] and will be used later to obtain our main result.

Lemma 2.7. *Let S be a monoid acting by partial transformations on an n element set and suppose that some element of S acts as the empty function. Let Σ be a generating set for S. Then there is a word $w \in \Sigma^*$ of length at most $n(n+1)/2$ acting as the empty function.*

Rystsov shows in [34] that the bound in the above lemma is tight. The monoid in his example is an inverse semgroup.

3 The Structure of Monoids in EDS

We briefly recall here some structural results concerning monoids in **EDS**. The reader is referred to [32, Appendix A] or [1, 9, 21] for the basic structure theory of finite monoids. Let us denote by $\mathrm{Reg}(S/\mathscr{J})$ the set of regular $\mathscr{J}$-classes of a monoid S. We write J_s for the $\mathscr{J}$-class of s and use similar notation for $\mathscr{L}$-classes and $\mathscr{R}$-classes.

Let J be a regular $\mathscr{J}$-class of a monoid S. Then there is an isomorphism $J^0 \cong \mathscr{M}^0(G, A, B, C)$ of the principal factor J^0 with a Rees matrix semigroup with sandwich matrix $C\colon B \times A \to G^0$ [9, 21, 32] where G is the maximal subgroup of J, A is the set of $\mathscr{R}$-classes of J and B is the set of $\mathscr{L}$-classes of J. It follows from Graham's Theorem [16] (cf. [32, Theorem 4.13.34]) that S belongs to **EDS** if and only if, for each regular $\mathscr{J}$-class J of S, we can always choose the sandwich matrix C to have a block diagonal form

$$C = \begin{bmatrix} C_1 & 0 & \cdots & 0 \\ 0 & C_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & C_r \end{bmatrix} \tag{3.1}$$

where each C_i is a matrix over G (with no zero entries). We define r to be the *rank* of J, which we denote by $\mathrm{rk}(J)$. It can be defined independently of the Rees matrix representation in the following way. Continuing to denote the set of $\mathscr{L}$-classes of J by B, define an equivalence relation on B by setting $L_1 \sim_L L_2$ if, for all $\mathscr{R}$-classses R of J, one has $R \cap L_1$ contains an idempotent if and only if $R \cap L_2$ contains an idempotent. Observing that $C_{ba} \neq 0$ if and only if the $\mathscr{H}$-class $b \cap a$ contains an idempotent, it follows that r is the number of blocks of the partition associated to $\sim_L$. (The journal version of this article will define the rank of a regular $\mathscr{J}$-class for arbitrary finite monoids.)

The monoid S acts by partial functions on the right of J by right multiplication, where rs is undefined if $r \in J$, $s \in S$, but $rs \notin J$. Moreover, it is easy to see that s acts as the empty function on J if and only if $J_s \ngeq_{\mathscr{J}} J$. Indeed, if $J_s \ngeq_{\mathscr{J}} J$, trivially s acts as the empty function. Conversely, if $usv \in J$

with $u, v \in S^1$, then since J is regular we can find an idempotent e so that $eusv = usv \in J$. Thus $eu, eus \in J$ and so the action of s on $eu \in J$ is defined. Define an equivalence relation $\equiv$ on J by putting $s \equiv t$ if $L_s \sim_L L_t$. Denote by $[r]$ the $\equiv$-class of r.

Proposition 3.1. *There is a well defined action of S on $J/{\equiv}$ by partial functions given by*

$$[r]s = \begin{cases} [rs] & rs \in J \\ \textit{undefined} & \textit{else.} \end{cases}$$

Moreover, $s \in S$ acts as the empty function on $J/{\equiv}$ if and only if $J_s \not\geq_{\mathscr{J}} J$.

Proof. Let us begin with the following claim.

Claim. Suppose $t_1 \equiv t_2$. Then, for all $x \in J$, one has $t_1x \in J$ if and only if $t_2x \in J$.

Proof. If $E(J)$ denotes the set of idempotents of J, then standard finite semigroup theory [1, 9, 21, 32] yields

$$\begin{aligned} t_1x \in J &\iff L_{t_1} \cap R_x \cap E(J) \neq \emptyset \\ &\iff L_{t_2} \cap R_x \cap E(J) \neq \emptyset \\ &\iff t_2x \in J. \end{aligned}$$

□

Suppose now that $t_1 \equiv t_2$ and let $s \in S$. We first establish that $t_1s \in J$ if and only if $t_2s \in J$. Indeed, if $t_1s \in J$, we can find an idempotent $e \in E(J)$ so that $t_1se = t_1s \in J$ by regularity of J. Thus $se \in J$ and so by the claim $t_2se \in J$, whence $t_2s \in J$. The reverse implication is proved in an identical manner.

Next assume that $t_1s, t_2s \in J$. We establish that if R' is an $\mathscr{R}$-class of J, then

$$R' \cap L_{t_1s} \cap E(J) \neq \emptyset \iff R' \cap L_{t_2s} \cap E(J) \neq \emptyset.$$

Suppose that $e \in R' \cap L_{t_1s}$ is an idempotent. Then $t_1se = t_1s \in J$ and $se \in J$. Thus the claim implies $t_2se \in J$. It follows that $R' \cap L_{t_2s}$ contains an idempotent. The reverse implication is proved in the same fashion. We conclude that the action of S on $J/\{\equiv\}$ is well defined. Verifying the axioms of an action is straightforward and left to the reader.

By the definition of the action, it is clear that $s \in S$ acts as the empty function on $J/{\equiv}$ if and only if it acts as the empty function on J. The final statement now follows from the discussion before the proposition. □

The above proposition is valid for regular $\mathscr{J}$-classes of any monoid, not just those in **EDS**. However, one can verify that $S \in$ **EDS** if and only if the above action is by partial injective functions for each regular $\mathscr{J}$-class J. Since we do not need this result, we do not prove it here.

The following proposition will be used to estimate mortality bounds for monoids in **EDS**.

Proposition 3.2. *Let $S \in \mathbf{EDS}$ and suppose that J is a regular $\mathscr{J}$-class of S other than the minimal ideal. Given a generating set Σ for S, there is a word $w \in \Sigma^*$ of length at most $\mathrm{rk}(J)(\mathrm{rk}(J)+1)/2$ so that $J_w \not\geq_{\mathscr{J}} J$.*

Proof. The result follows from applying Lemma 2.7 to the action of S on $J/{\equiv}$ by partial functions and using Proposition 3.1. □

4 A Mortality Function for EDS

We begin with some basic facts concerning the representation theory of monoids. We take a minimalist approach, stating exactly what we need in order to prove our main result. Details can be found in [9, 15, 24, 30, 33]. To fix notation, if S is a monoid, we use $\mathrm{Irr}(S)$ to denote the set of equivalence classes of irreducible representations of S. The reader should verify that every irreducible representation of a group is by invertible maps.

Let S be a monoid. Fix a maximal subgroup G_J for each regular $\mathscr{J}$-class J of S. Then the theory of Munn and Ponizovsky says that $\mathrm{Irr}(S)$ is in bijection with $\coprod_{J\in \mathrm{Reg}(S/\mathscr{J})} \mathrm{Irr}(G_J)$. Following Munn, if ρ^* is the irreducible representation of S corresponding to an irreducible representation ρ of G_J, then the $\mathscr{J}$-class J is called the *apex* of ρ^*. Suppose that d is the degree of ρ. Let $C\colon B\times A \to G_J^0$ be the sandwich matrix for J and denote by $\rho\otimes C$ the $d|B|\times d|A|$ matrix obtained by applying ρ entrywise to C (where we take $\rho(0)$ to be the $d\times d$ zero matrix). The following result can be extracted from [33] and [9, Chapter 5]; see also [30] and [31, Chapter 15] for a summary without proofs or [15, 23] for module-theoretic statements and proofs.

Theorem 4.1 (Munn, Ponizovsky). *Suppose that S is a finite monoid. Let $\rho^*\colon S \to M_n(\mathbb{Q})$ be an irreducible representation with apex J corresponding to an irreducible representation $\rho\colon G_J \to M_d(\mathbb{Q})$ of the maximal subgroup of J. Let $C\colon B \times A \to G_J^0$ be the sandwich matrix of J. Then:*

1. *The degree of ρ^* is the rank of the matrix $\rho\otimes C$;*
2. *For $s \in S$, one has $\rho^*(s) = 0$ if and only if $J_s \not\geq_{\mathscr{J}} J$.*

Now we are ready to prove that $f(n) = n(n+1)/2$ is a superadditive mortality function for monoids in **EDS**.

Theorem 4.2. *Let $S \in \mathbf{EDS}$. Then $f(n) = n(n+1)/2$ is a superadditive mortality function for S.*

Proof. It is routine to verify that f is superadditive. Thus it suffices to consider irreducible representations by Proposition 2.5. So let $\varphi\colon S \to M_n(\mathbb{Q})$ be an irreducible representation with $0 \in \varphi(S)$ and let Σ be a generating set for S. Let $J \in \mathrm{Reg}(S/\mathscr{J})$ be the apex of φ; note that J is not the minimal ideal of S. Suppose that $\varphi = \rho^*$ where $\rho\colon G_J \to M_d(\mathbb{Q})$ is an irreducible representation of the maximal subgroup G_J of J. Putting $r = \mathrm{rk}(J)$, we can find by Proposition 3.2 a word w of length at most $r(r+1)/2$ with $J_w \not\geq_{\mathscr{J}} J$ and hence with $\varphi(w) = 0$

by Theorem 4.1. It thus suffices to prove that $r \leq n$. Since $S \in \mathbf{EDS}$, we can place C in the block form (3.1) where the C_i are matrices over G_J (with no zero entries). Then evidently,

$$\rho \otimes C = \begin{bmatrix} \rho \otimes C_1 & 0 & \cdots & 0 \\ 0 & \rho \otimes C_2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \rho \otimes C_r \end{bmatrix}.$$

Since $\rho(g)$ is invertible for all $g \in G$, it now follows that the rank of $\rho \otimes C$ is at least r. But this rank is the degree n of φ by Theorem 4.1. This completes the proof of the theorem. □

We can now resolve the Černý-Pin conjecture for automata with transition monoid in **EDS**.

Corollary 4.3. *Every synchronizing automaton with transition monoid in* **EDS** *has a reset word of length at most $n(n-1)/2$ and this bound is sharp. More generally, if (Q, Σ) is an automaton of rank r whose transition monoid is in* **EDS**, *then there is a word of length at most $(n-r)(n-r+1)/2$ representing a transformation of rank r.*

Proof. The upper bound is a direct consequence of Proposition 2.3 and Theorem 4.2. The sharpness follows from an example of Rystsov [34] of an n-state synchronizing automaton whose transition monoid has commuting idempotents with shortest reset word of length $n(n-1)/2$. □

5 A Universal Mortality Function

It is natural to ask whether there is a single function that is a mortality function for every finite monoid.

Definition 5.1 (Universal mortality function). *A* ***universal mortality function*** *is a function $f\colon \mathbb{N} \to \mathbb{N}$ which is a mortality function for all finite monoids.*

It is not *a priori* clear that there exist universal mortality functions. In fact, a famous result of Paterson [26] asserts that it is undecidable whether the monoid generated by a finite collection of 3×3 integer matrices contains the zero matrix and so there can be no 'universal' mortality function if one lifts the restriction on finiteness. A result proved independently by Mandel and Simon [22] and by Jacob [17] (see also [7, Chapter IX]) easily implies that one can find a simultaneous mortality function for all finitely generated monoids with at most k generators for any given k. But this is not good enough for our purposes.

We use the methods from the solution of the Burnside problem for matrix semigroups [7, 14, 25, 40] to establish the existence of a universal mortality function. More precisely, we show that the function $f\colon \mathbb{N} \to \mathbb{N}$ given by

$$f(n) = \begin{cases} 1 & n = 1 \\ (2n-1)^{n^2} - 1 & n > 1 \end{cases} \tag{5.1}$$

is a superadditive universal mortality function. The journal version of the paper will contain a slightly better bound.

The proof of the following elementary proposition is left to the reader.

Proposition 5.2. *The function f from (5.1) is superadditive.*

So to prove that f is a universal mortality function, it suffices to consider irreducible representations. Let us say that a submonoid S of $M_n(\mathbb{Q})$ is *irreducible* if the inclusion map $S \hookrightarrow M_n(\mathbb{Q})$ is an irreducible representation.

Recall that a subalgebra $\mathfrak{A} \subseteq M_n(\mathbb{Q})$ is said to be *irreducible* if the only $\mathfrak{A}$-invariant subspaces of $\mathbb{Q}^n$ are $\{0\}$ and $\mathbb{Q}^n$. An algebra is *simple* if it has no ideals. We shall need the following well-known result (cf. [9, Theorem 5.7]) going back to Burnside.

Theorem 5.3. *An irreducible subalgebra of $M_n(\mathbb{Q})$ is simple.*

If $a \in M_n(\mathbb{Q})$, then $\operatorname{tr} a$ denotes the trace of a. Our next lemma relies on a little bit of algebraic number theory.

Lemma 5.4. *Let $a \in M_n(\mathbb{Q})$ have finite order, that is, $|\langle a\rangle| < \infty$. Then $\operatorname{tr} a \in \mathbb{Z}$ and $|\operatorname{tr} a| \leq n$. Moreover, if $|\operatorname{tr} a| = n$, then a is invertible.*

Proof. By assumption, $a^m = a^{m+k}$ for some $m, k > 0$. Thus the minimal polynomial of a divides $x^m(x^k - 1)$ and so each non-zero eigenvalue is a k^{th}-root of unity. Thus $\operatorname{tr} a$ is an algebraic integer, being a sum of algebraic integers. But $\operatorname{tr} a \in \mathbb{Q}$ and hence $\operatorname{tr} a \in \mathbb{Z}$ as the rational algebraic integers are precisely the integers. Suppose $\lambda_1, \ldots, \lambda_n$ are the complex eigenvalues of a listed with multiplicity. Then

$$|\operatorname{tr} a| = \left|\sum_{i=1}^{n} \lambda_i\right| \leq \sum_{i=1}^{n} |\lambda_i| \leq n$$

since each λ_i is zero or a root of unity. Moreover, if $|\operatorname{tr} a| = n$, then no $\lambda_i = 0$ and so a is invertible. □

The following lemma uses traces to bound mortality. The essential idea goes back to Burnside [10]. We use the well-known and easy to prove fact that if S is a monoid with n elements generated as a monoid by a set Σ, then each element of S can be represented by a word of length at most $n - 1$.

Lemma 5.5. *Let $\Sigma \subseteq M_n(\mathbb{Q})$ be such that $S = \langle \Sigma \rangle$ is a finite irreducible submonoid, $0 \in S$ and $S \setminus \{0\}$ contains a singular matrix. Let J be the apex of the irreducible representation $S \hookrightarrow M_n(\mathbb{Q})$ and let G be a maximal subgroup of J. Suppose that $|\{\operatorname{tr} g \mid g \in G\} \cup \{0\}| = m$. Then there is a word $w \in \Sigma^*$ of length at most $m^{n^2} - 1$ mapping to the zero matrix in S.*

Proof. Let $\mathfrak{A}$ be the subalgebra of $M_n(\mathbb{Q})$ spanned by S; then $\mathfrak{A}$ is irreducible and hence simple by Theorem 5.3. Let $I = J \cup \{0\}$; it is the unique 0-minimal ideal of S as a consequence of Theorem 4.1. The span of I is a non-zero ideal of $\mathfrak{A}$ and hence $\mathfrak{A}$, being simple, is spanned by I. Thus there exists a basis $\{s_1, \ldots, s_d\}$ for $\mathfrak{A}$ consisting of elements of J; note that $d = \dim \mathfrak{A} \leq n^2$.

Consider now the trace form $(a, b) \mapsto \mathrm{tr}(ab)$ on $\mathfrak{A}$. The associativity of multiplication in $\mathfrak{A}$ and the linearity of the trace functional immediately yield that the trace form is a (symmetric) bilinear form on $\mathfrak{A}$. Since the identity matrix I_n is in $S \subseteq \mathfrak{A}$ and $\mathrm{tr}\, I_n = n$, it follows that the trace form is not identically 0 on $\mathfrak{A}$. Thus the radical of the trace form is a proper ideal of $\mathfrak{A}$, and hence zero by the simplicity of $\mathfrak{A}$. Thus the trace form is non-degenerate on $\mathfrak{A}$. Consequently, if $a \in \mathfrak{A}$, then a is determined by the d rational numbers $\mathrm{tr}(as_i)$, for $i = 1, \ldots, d$.

In particular, consider $s \in S$. Then $ss_i \in I$, for $i = 1, \ldots, d$. Let

$$A = \{\mathrm{tr}\, g \mid g \in G\} \cup \{0\}.$$

We claim that $\mathrm{tr}(ss_i) \in A$. This is evident if $ss_i = 0$. If $ss_i \in J$, but not in a maximal subgroup, then $(ss_i)^2 = 0$ and hence $\mathrm{tr}(ss_i) = 0$ (since it has only 0 as an eigenvalue). Finally, suppose ss_i belongs to some maximal subgroup of J. Then we can find by Green-Rees Theory [32, Appendix A] elements $x, x' \in J$ so that $xx'x = x, x'xx' = x'$, $x'xss_i = ss_i$ and $xss_ix' \in G$. Then

$$\mathrm{tr}(ss_i) = \mathrm{tr}(x'xss_i) = \mathrm{tr}(xss_ix') \in A$$

establishing the claim. As a consequence, $\mathrm{tr}(ss_i)$ takes on at most $m = |A|$ values for $s \in S$ and so S has at most m^d elements. Thus there is a word $w \in \Sigma^*$ of length at most $m^d - 1$ representing 0 in S. As $d \leq n^2$, this provides the desired result. □

We are now ready to prove the main theorem of this section.

Theorem 5.6. *The function f from* (5.1) *is a universal mortality function.*

Proof. Because f is superadditive, it suffices by Proposition 2.5 to show that if $\Sigma \subseteq M_n(\mathbb{Q})$ is such that $S = \langle \Sigma \rangle$ is a finite irreducible submonoid and $0 \in S$, then there exists a word $w \in \Sigma^*$ with $|w| \leq f(n)$ and $w = 0$ in S. If $S \setminus \{0\}$ contains only invertible elements, then $0 \in \Sigma$ and there is nothing to prove. So assume that S contains non-zero singular matrices (and hence $n > 1$).

Let J be the apex of $S \hookrightarrow M_n(\mathbb{Q})$ and let G be a maximal subgroup of J. Then since S is finite and G consists of singular matrices, it follows from Lemma 5.4 that $\{|\,\mathrm{tr}(g)| \mid g \in G\} \cup \{0\} \subseteq \{0, \ldots, n-1\}$ and hence has at most $2n - 1$ elements. Thus Lemma 5.5 yields the desired result. □

This leaves open an obvious question:

Question 5.7. Is there a polynomial universal mortality function? How about an exponential one?

Rystsov [35] conjectured that n^2 would be a universal mortality function, but he also conjectured this bound should hold over all finite fields, which is impossible given the undecidability of matrix mortality for integer matrices [26]. However, the best known lower bound to our knowledge is n^2 coming from the lower bound for the Černý problem.

Let us prove that for aperiodic monoids, we can find a better mortality function than (5.1). Recall that a monoid is *aperiodic* if all its maximal subgroups are trivial. The journal version of this paper deals with further classes of monoids.

Theorem 5.8. *The superadditive function $k(n) = 2^{n^2} - 1$ is a mortality function for all aperiodic monoids.*

Proof. Routine computation shows that k is superadditive. So it suffices by Proposition 2.5 to deal with irreducible representations $\rho\colon S \to M_n(\mathbb{Q})$. Without loss of generality we may assume that S is an irreducible aperiodic submonoid of $M_n(\mathbb{Q})$ and ρ is the inclusion. If $S \setminus \{0\}$ contains only invertible elements, then $0 \in \Sigma$ and there is nothing to prove. So assume that S contains non-zero singular matrices (and hence $n > 1$).

Let J be the apex of ρ and let G be a maximal subgroup of J; then $G = \{e\}$ where e is an idempotent. Set $V = \mathbb{Q}^n$. Then $\operatorname{tr} e = \dim Ve$. But the theory of Munn and Ponizovsky implies that the restriction of the action of G to Ve gives an irreducible representation of G [9, 15, 30, 33]. Since G is the trivial group, this implies $\dim Ve = 1$ and so $\operatorname{tr} e = 1$. Thus $|\{\operatorname{tr} g \mid g \in G\} \cup \{0\}| = 2$ from which Lemma 5.5 yields the desired result. □

References

1. Almeida, J.: Finite semigroups and universal algebra. Series in Algebra, vol. 3. World Scientific Publishing Co. Inc, River Edge (1994); Translated from the 1992 Portuguese original and revised by the author
2. Almeida, J., Margolis, S., Steinberg, B., Volkov, M.: Representation theory of finite semigroups, semigroup radicals and formal language theory. Trans. Amer. Math. Soc. 361(3), 1429–1461 (2009)
3. Ananichev, D.S., Volkov, M.V.: Some results on Černý type problems for transformation semigroups. In: Semigroups and languages, pp. 23–42. World Sci. Publ., River Edge (2004)
4. Ananichev, D.S., Volkov, M.V.: Synchronizing generalized monotonic automata. Theoret. Comput. Sci. 330(1), 3–13 (2005)
5. Ananichev, D.S., Volkov, M.V., Zaks, Y.I.: Synchronizing automata with a letter of deficiency 2. Theoret. Comput. Sci. 376(1-2), 30–41 (2007)
6. Arnold, F., Steinberg, B.: Synchronizing groups and automata. Theoret. Comput. Sci. 359(1-3), 101–110 (2006)
7. Berstel, J., Reutenauer, C.: Rational series and their languages. EATCS Monographs on Theoretical Computer Science, vol. 12. Springer, Berlin (1988)
8. Černý, J.: A remark on homogeneous experiments with finite automata. Mat.-Fyz. Časopis Sloven. Akad. Vied 14, 208–216 (1964)
9. Clifford, A.H., Preston, G.B.: The algebraic theory of semigroups. Mathematical Surveys, vol. I(7). American Mathematical Society, Providence (1961)

10. Curtis, C.W., Reiner, I.: Representation theory of finite groups and associative algebras. Wiley Classics Library. John Wiley & Sons Inc, New York (1988); Reprint of the 1962 original, A Wiley-Interscience Publication
11. Dornhoff, L.: Group representation theory. In: Part A: Ordinary representation theory, Marcel Dekker Inc., New York (1971); Pure and Applied Mathematics, 7
12. Dubuc, L.: Sur les automates circulaires et la conjecture de Černý. RAIRO Inform. Théor. Appl. 32(1-3), 21–34 (1998)
13. Eilenberg, S.: Automata, languages, and machines, vol. B. Academic Press, New York (1976); With two chapters "Depth decomposition theorem" and "Complexity of semigroups and morphisms by Bret Tilson", Pure and Applied Mathematics, vol. 59
14. Freedman, A., Gupta, R.N., Guralnick, R.M.: Shirshov's theorem and representations of semigroups. Pacific J. Math (Special Issue), 159–176 (1997); Olga Taussky-Todd: in memoriam
15. Ganyushkin, O., Mazorchuk, V., Steinberg, B.: On the irreducible representations of a finite semigroup. Proc. Amer. Math. Soc. (to appear)
16. Graham, R.L.: On finite 0-simple semigroups and graph theory. Math. Systems Theory 2, 325–339 (1968)
17. Jacob, G.: Un algorithme calculant le cardinal, fini ou infini, des demi-groupes de matrices. Theoret. Comput. Sci. 5(2), 183–204 (1977/1978)
18. Kari, J.: A counter example to a conjecture concerning synchronizing words in finite automata. Bull. Eur. Assoc. Theor. Comput. Sci. EATCS (73), 146 (2001)
19. Kari, J.: Synchronizing finite automata on Eulerian digraphs. Theoret. Comput. Sci. 295(1-3), 223–232 (2003) Mathematical foundations of computer science (Mariánské Lázně, 2001)
20. Klyachko, A.A., Rystsov, I.C., Spivak, M.A.: On an extremal combinatorial problem connected with an estimate for the length of a reflexive word in an automaton. Kibernetika (Kiev) (2), 16–20, 25, 132 (1987)
21. Krohn, K., Rhodes, J., Tilson, B.: Algebraic theory of machines, languages, and semigroups. Edited by Michael A. Arbib. With a major contribution by Kenneth Krohn and John L. Rhodes, vol. 1, pp. 5–9. Academic Press, New York (1968)
22. Mandel, A., Simon, I.: On finite semigroups of matrices. Theoret. Comput. Sci. 5(2), 101–111 (1977/1988)
23. Margolis, S.W., Steinberg, B.: The quiver of an algebra associated to the mantaci-reutenauer descent algebra and the homology of regular semigroups. Algebr. Represent. Theory (to appear)
24. McAlister, D.B.: Characters of finite semigroups. J. Algebra 22, 183–200 (1972)
25. McNaughton, R., Zalcstein, Y.: The Burnside problem for semigroups. J. Algebra 34, 292–299 (1975)
26. Paterson, M.S.: Unsolvability in 3×3 matrices. Studies in Appl. Math. 49, 105–107 (1970)
27. Pin, J.-E.: Sur un cas particulier de la conjecture de Cerny. In: Ausiello, G., Böhm, C. (eds.) ICALP 1978. LNCS, vol. 62, pp. 345–352. Springer, Heidelberg (1978)
28. Pin, J.-E.: Le problème de la synchronisation et la conjecture de Černý. In: Noncommutative structures in algebra and geometric combinatorics (Naples, 1978). Quad. Ricerca Sci., vol. 109, pp. 37–48. CNR, Rome (1981)
29. Pin, J.-E.: On two combinatorial problems arising from automata theory. In: Combinatorial mathematics (Marseille-Luminy, 1981). North-Holland Math. Stud., vol. 75, North-Holland, Amsterdam (1983)
30. Putcha, M.S.: Complex representations of finite monoids. II. Highest weight categories and quivers. J. Algebra 205(1), 53–76 (1998)

31. Renner, L.E.: Linear algebraic monoids. Encyclopaedia of Mathematical Sciences, vol. 134. Springer, Berlin (2005); Invariant Theory and Algebraic Transformation Groups, V
32. Rhodes, J., Steinberg, B.: The q-theory of finite semigroups. Springer Monographs in Mathematics. Springer, New York (2009)
33. Rhodes, J., Zalcstein, Y.: Elementary representation and character theory of finite semigroups and its application. In: Monoids and semigroups with applications (Berkeley, CA, 1989), pp. 334–367. World Sci. Publ., River Edge (1991)
34. Rystsov, I.: Reset words for commutative and solvable automata. Theoret. Comput. Sci. 172(1-2), 273–279 (1997)
35. Rystsov, I.C.: On the rank of a finite automaton. Kibernet. Sistem. Anal. 187(3), 3–10 (1992)
36. Rystsov, I.K.: Quasioptimal bound for the length of reset words for regular automata. Acta Cybernet. 12(2), 145–152 (1995)
37. Rystsov, I.K.: On the length of reset words for automata with simple idempotents. Kibernet. Sistem. Anal. 187(3), 32–39 (2000)
38. Schützenberger, M.P.: Sur le produit de concaténation non ambigu. Semigroup Forum 13(1), 47–75 (1976/1977)
39. Steinberg, B.: Černý's conjecture and group representation theory (preprint) (2008)
40. Steinberg, B.: Yet another solution to the Burnside problem for matrix semigroups. Canad. Math. Bull. (to appear)
41. Trahtman, A.N.: An efficient algorithm finds noticeable trends and examples concerning the Černy conjecture. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 789–800. Springer, Heidelberg (2006)
42. Trahtman, A.N.: The Černý conjecture for aperiodic automata. Discrete Math. Theor. Comput. Sci. 9(2), 3–10 (2007) (electronic)
43. Volkov, M.V.: Synchronizing automata and the Černý conjecture. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 11–27. Springer, Heidelberg (2008)

A Quadratic Upper Bound on the Size of a Synchronizing Word in One-Cluster Automata

Marie-Pierre Béal and Dominique Perrin

Université Paris-Est, LIGM CNRS, 77454 Marne-la-Vallée Cedex 2, France
{beal,perrin}@univ-mlv.fr

Abstract. Černý's conjecture asserts the existence of a synchronizing word of length at most $(n-1)^2$ for any synchronized n-state deterministic automaton. We prove a quadratic upper bound on the length of a synchronizing word for any synchronized n-state deterministic automaton satisfying the following additional property: there is a letter a such that for any pair of states p, q, one has $p \cdot a^r = q \cdot a^s$ for some integers r, s (for a state p and a word w, we denote by $p \cdot w$ the state reached from p by the path labeled w). As a consequence, we show that for any finite synchronized prefix code with an n-state decoder, there is a synchronizing word of length $O(n^2)$. This applies in particular to Huffman codes.

1 Introduction

Synchronized automata are deterministic and complete finite-state automata admitting a synchronizing word, that is a word which takes each state of the automaton to a single special state. Černý conjecture claims that each n-state synchronized automaton has a synchronizing word of length at most $(n-1)^2$ [5]. An extension of this conjecture due to Pin [12,14] was shown to be false by Kari [9]. The conjecture has been shown to be true for particular classes of automata like the class of circular automata by Dubuc [6] (see also [13]). A $n(n-1)/2$ upper bound has been obtained by Trahtman [17,19] for aperiodic automata. This upper bound was improved to $n(n+1)/6$ by Volkov [20] (see also [21]).

In a previous note [1], the first author gave a proof of a quadratic bound for circular automata which is simpler than the one given in [6]. Nevertheless, it does not allow one to get the tight $(n-1)^2$ bound. The proof uses rational series.

The formulation of the problem in terms of rational series is also used in [1] to provide a simple proof of a result from Kari [10] which proves Černý's conjecture for automata with an underlying Eulerian graph.

Later, the result on circular automata was generalized by Carpi and d'Alessandro to a larger class called strongly transitive automata [4]. Their proof uses rational series as in [1]. They use the same methods to generalize the result of Kari to unambiguous automata.

In this paper, we prove the existence of a quadratic upper bound for the length of a synchronizing word for a class of finite automata called one-cluster. This

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 81–90, 2009.

means that, for some letter a, there is only one cycle with all edges labeled by a. The proof is an extension of the argument of [1] and uses again rational series.

The class of one-cluster automata contains in particular the automata associated with finite prefix codes. We thus obtain the existence of a quadratic bound on the length of a synchronizing word for a finite maximal synchronized prefix code. This applies in particular to Huffman codes.

Let us mention two recent results connected to our work (we thank Hugo Vaccaro for pointing out these references to us). First, it is proved in [8] that almost all finite maximal prefix codes are synchronizing. Next, in [3], it is proved that a finite maximal synchronized prefix code with n codewords of maximal length h has a synchronizing word of length $O(nh \log n)$. This bound is not comparable with Indeed, since $\log n \leq h \leq n-1$, one has $n(\log n)^2 \leq nh \log n \leq n^2 \log n$.

2 Automata and Series

Let A be a finite alphabet and A^* be the set of finite words drawn from the alphabet A, the empty word ϵ included. A (finite) *automaton* $\mathcal{A}$ over some (finite) alphabet A is composed of a finite set Q of states and a finite set E of edges which are triples (p, a, q) where p, q are states and a is a symbol from A called the *label* of the edge. Note that we do not specify a set of terminal states and that, for this reason, our automata are sometimes called semi-automata.

An automaton is *deterministic* if, for each state p and each letter a, there is at most one edge starting in p and labeled with a. It is *complete deterministic* if, for each state p and each letter a, there is exactly one edge starting in p and labeled with a. This implies that, for each state p and each word w, there is exactly one path starting in p and labeled with w. We denote by $p \cdot w$ the state which is the end of this unique path.

An automaton is *irreducible* if its underlying graph is strongly connected.

A *synchronizing word* of a deterministic complete automaton is a word w such that for any states p, q, one has $p \cdot w = q \cdot w$. A synchronizing word is also called a *reset sequence* or a *magic sequence*, or also a *homing word*. An automaton which has a synchronizing word is called *synchronized* (see an example on the right part of Figure 1).

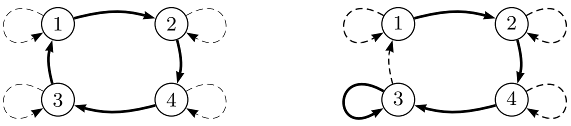


Fig. 1. Two complete deterministic automata labeled on $A = \{a, b\}$. A thick plain edge is an edge labeled by a while a thin dashed edge is an edge labeled by b. The automaton on the left is not synchronized. The one on the right is synchronized; for instance, the word aaa is a synchronizing word.

Let $\mathcal{A} = (Q, E)$ be a complete deterministic automaton. For any word $u \in A^*$, we denote by M_u the transition matrix of the action of u on the states Q. It is defined by:

$$(M_u)_{pq} = \begin{cases} 1 & \text{if } p \cdot u = q, \\ 0 & \text{otherwise.} \end{cases}$$

Note that if u, v are two words, we have

$$M_{uv} = M_u M_v.$$

We define the *rank* of a word u as the cardinality of $Q \cdot u$. Note that since the automaton is complete deterministic, this rank is non null, and that a word is synchronizing if and only if his rank is 1.

A *circular* automaton is a deterministic complete automaton on the alphabet A such that there is a letter a of A which induces a circular permutation of the states, *i.e.* such that M_a is a circular permutation matrix.

We shall consider the set of non commutative formal series with coefficients in a ring K (with $K = \mathbb{Z}$ or $K = \mathbb{Q}$), which are applications from A^* to K. If S is such a series, the image of a word u of A^* by S is denoted by $\langle S, u \rangle$ and called the coefficient of u in S.

As an example, the series S on $\{a, b\}^*$ with coefficients in $\mathbb{Z}$ defined by $\langle S, u \rangle = |u|_a - |u|_b$ maps a word $u \in \{a, b\}^*$ to the difference between the number of occurrences of a and b in u.

A *K-linear representation* of dimension d of a series S is a triple (λ, μ, γ) where $\lambda \in K^{1 \times d}$, μ is a morphism from A^* to $K^{d \times d}$, and $\lambda \in K^{d \times 1}$, such that

$$\langle S, u \rangle = \lambda \mu(u) \gamma.$$

A series s is *K-rational* if it has a K-linear representation. Its *rank* on K is the minimal dimension of all its linear representations.

For example, the series S defined on $\{a, b\}^*$ by $\langle S, u \rangle = |u|_a - |u|_b$ is rational of rank 2. The triple (λ, μ, γ) defined by

$$\lambda = \begin{bmatrix} 1 & 0 \end{bmatrix}, \mu(a) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \mu(b) = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}, \gamma = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

is a $\mathbb{Z}$-linear representation of S of dimension 2.

Černý's conjecture gives an upper bound on the size of a shortest synchronizing word in a synchronized automaton.

Conjecture 1 (Černý 1964). A synchronized n-state deterministic complete automaton has a synchronizing word of length at most $(n-1)^2$.

The conjecture was proved by Dubuc for circular automata.

Proposition 1 (Dubuc 1998). *A circular synchronized n-state deterministic complete automaton has a synchronizing word of size at most $(n-1)^2$.*

3 One-Cluster Automata

In the sequel $\mathcal{A} = (Q, E)$ denotes an n-state deterministic and complete automaton over an alphabet A. We fix a particular letter $a \in A$.

Let $\mathcal{R}$ be the subgraph of the graph of $\mathcal{A}$ made of the edges labeled by a. The graph $\mathcal{R}$ is a disjoint union of connected component called *a-clusters*. Since each state has exactly one outgoing edge in $\mathcal{R}$, each a-cluster contains a unique cycle, called an a-cycle, with trees attached to the cycle at their root. For each state p of the a-cluster, we define the *level* of p as the distance between p and the root of the tree containing p. If p belongs to the cycle, its level is null. The *level* of the automaton is the maximal level of its states.

A *one-cluster automaton* with respect to a letter a is a complete deterministic automaton which has only one a-cluster. Equivalently, an automaton is one-cluster if it satisfies the following condition: for any pair of states p, q, one has $p \cdot a^r = q \cdot a^s$ for some integers r, s.

Note that a one-cluster automaton whose level is null is circular.

Let C be a cycle of $\mathcal{A}$ and P be a subset of C. A word u is said to be *P-augmenting* if

$$\mathrm{card}(Pu^{-1} \cap C) > \mathrm{card}(P),$$

where we denote $Pu^{-1} = \{q \in Q \mid q \cdot u \in P\}$.

We now prove the existence of a quadratic upper bound on the size of a shortest synchronizing word in a synchronized automaton.

Proposition 2. *Let $\mathcal{A}$ be a synchronized n-state deterministic complete automaton. If $\mathcal{A}$ is one-cluster, then it has a synchronizing word of length at most $1 + 2(n-1)(n-2)$.*

We prove the proposition for irreducible automata. The case of reducible automata easily reduces to this one. Let $\mathcal{A} = (Q, E)$ be a deterministic complete and irreducible n-state automaton.

Since Černý's conjecture is proved for circular automata, we may assume that the level ℓ of the automaton is greater than or equal to 1.

Let C be the a-cycle and let m be the length of C. Let P be a subset of C. Note that a word u is a P-augmenting word if and only if

$$C\, M_u\, P^t > CP^t,$$

where P, C denote the characteristic row vectors of the sets P, C. Indeed,

$$C\, M_u\, P^t = \sum_{r \in C, s \in P} (M_u)_{rs} = \mathrm{card}\{r \in C \mid r \cdot u \in P\} = \mathrm{card}(Pu^{-1} \cap C).$$

Similarly, $CP^t = \mathrm{card}(P)$.

We denote by S the series defined by:

$$\langle S, u\rangle = C\, M_u P^t - C\, P^t,$$

By definition, one has $\langle S, u\rangle > 0$ if and only if u is a P-augmenting word.

Lemma 1. *The series S has rank on $\mathbb{Q}$ at most n.*

Proof. The series S is $\mathbb{Z}$-rational since it is the difference of two $\mathbb{Z}$-rational series (the second one is actually a constant). It has the following linear representation (λ, μ, γ) with $\lambda \in \mathbb{Z}^{1\times 2n}, \mu : A^* \to \mathbb{Z}^{2n\times 2n}, \lambda \in \mathbb{Z}^{2n\times 1}$,

$$\lambda = (C, -C),\ \mu(u) = \begin{bmatrix} M_u & 0 \\ 0 & I \end{bmatrix},\ \gamma = \begin{bmatrix} P^t \\ P^t \end{bmatrix},$$

since $\langle S, u\rangle = \lambda\mu(u)\gamma$. The rank of S on $\mathbb{Q}$ is bounded above by the dimension of the $\mathbb{Q}$-vector space generated by the row vectors $(CM_u, -C)$. This space is included in the vector space generated by the vectors $(CM_u - C, \mathbf{0})$ and the row vector $(C, -C)$, where $\mathbf{0}$ is the null column vector of size n. Thus the rank of S is at most equal to the dimension of the vector space V generated by the vectors $C(M_u - I)$, plus one. We now show that the dimension of V is at most $n-1$. Since the automaton $\mathcal{A}$ is complete deterministic, for any $u \in A^*$, $M_u\mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ is the column vector with coefficients 1 of size n. This implies that $C(M_u - I)\cdot\mathbf{1} = 0$. Thus the vectors of V are orthogonal to the vector $\mathbf{1}$. The dimension of V is thus at most $n-1$. This proves that the rank of S on $\mathbb{Q}$ is at most n.

We denote by T the $\mathbb{Z}$-rational series defined by

$$\langle T, u\rangle = \langle S, ua^\ell\rangle,$$

where ℓ denotes the level of the automaton $\mathcal{A}$. If (λ, μ, γ) is a $\mathbb{Q}$-linear representation of S of dimension n, then $(\lambda, \mu, \mu(a^\ell)\gamma)$ is a representation of T. Thus the rank of T on on $\mathbb{Q}$ is at most n.

Lemma 2. *For any subset P of C such that $P \neq \emptyset$ and $P \neq C$, there is a P-augmenting word of length at most $2(n-1)+\ell$.*

Proof. Since $\mathcal{A}$ is synchronized and irreducible, there is a synchronizing word u such that $Q\cdot u$ is a single state r belonging to P. Let k be a positive integer such that $km \geq \ell$, where m is the length of the cycle C. We have $Q\cdot ua^{km-\ell}a^\ell = r\cdot a^{km} = r$. Hence $ua^{km-\ell}a^\ell$ also a synchronizing word focusing to r. Let R denote the characteristic row vector of r. Since $q\cdot u = r$ for all $q \in Q$ and since C has m elements, we have $CM_u = mR$. Moreover, since $r \in C$, $RM_{a^m} = R$. We have

$$\begin{aligned}
\langle T, ua^{km-\ell}\rangle &= \langle S, ua^{km}\rangle \\
&= CM_{ua^{km}}P^t - CP^t \\
&= CM_uM_{a^{km}}P^t - CP^t \\
&= mRM_{a^{km}}P^t - CP^t \\
&= mRP^t - CP^t \\
&= m - \mathrm{card}(P) \neq 0.
\end{aligned}$$

As a consequence T is non null.

Since T has rank at most n on $\mathbb{Q}$, there is a word v of length at most $n-1$ such that $\langle T, v\rangle \neq 0$ (see [7, p. 145] or [15, p. 492 corollaire 4.19]).

If there is word v of length at most $n-1$ such that $\langle T, u\rangle > 0$, then va^ℓ is a P-augmenting word and the claim is proved. Otherwise, there is a word v of length at most $n-1$ such that $\langle T, v\rangle < 0$.

Since ℓ is the level of $\mathcal{A}$, the vector CM_{ua^ℓ} is a linear combination of elements of C and the sum of its coefficients is equal to m.

We have

$$\begin{aligned}\sum_{i=0}^{m-1}\langle T, ua^i\rangle = \sum_{i=0}^{m-1}\langle S, ua^\ell a^i\rangle &= \sum_{i=0}^{m-1} C(M_{ua^\ell a^i} - I)P^t,\\ &= (\sum_{i=0}^{m-1} CM_{ua^\ell}M_{a^i} - \sum_{i=0}^{m-1} C)P^t,\\ &= (CM_{ua^\ell}(\sum_{i=0}^{m-1} M_{a^i}) - mC)P^t,\\ &= (\sum_{r\in C} rM_{ua^\ell}\sum_{i=0}^{m-1} M_{a^i} - mC)P^t,\\ &= (\sum_{r\in C} C - mC)P^t = 0.\end{aligned}$$

Indeed, for any r in C, the state $q = r\cdot va^\ell$ is in C and for any state q in C, the row of index q of the matrix $\sum_{i=0}^{m-1} M_{a^i}$ is the row vector C.

As a consequence, there is a word w of length at most $n+m-2$ such that $\langle T, w\rangle > 0$. Hence there is a P-augmenting word of length at most $n+m-2+\ell$.

Thus, in all cases, there is a word of length at most $n+m-2+\ell$ which is P-augmenting.

To prove Proposition 2, we show that $\mathcal{A}$ has a synchronizing word of length at most $1+2m(n-2)$. Indeed, let P_1 be reduced to an arbitrary state of C. If $P_1 = C$ (that is to say if $m=1$), then $Q\cdot a^\ell \subseteq P_1$, and thus a^ℓ is a synchronizing word.

Otherwise, by Lemma 2, there exists a word u_1 of length at most $n+m-2+\ell$ which is P_1-augmenting. Set $P_2 = P_1u_1^{-1}\cap C$. If $P_2 \neq C$, there is a word u_2 of length at most $n+m-1+\ell$ which is P_2-augmenting, and so on. In this way, we build a sequence $u_1, .., u_{t-1}$ of words and a sequence $P_1, .., P_t$ of sets of states, with $t \leq m-1$, such that, for $1 \leq i < t$,

- u_i is a P_i-augmenting word of length at most $n+m-1+\ell$;
- $P_{i+1} = P_iu_i^{-1}\cap C$;
- $P_t = C$.

Then the word $a^\ell u_{t-1}..u_1$ is a synchronizing word of length at most $\ell+(m-1)(n+m-2+\ell)$. Indeed, $Q\cdot a^\ell u_{t-1}..u_1 \subseteq C\cdot u_{t-1}..u_1 \subseteq P_{t-1}\cdot u_{t-2}..u_1 \subseteq ..P_2\cdot u_1 \subseteq P_1$.

Since $m \le n - \ell$ and $m \le n - 1$, we have

$$\begin{aligned}\ell + (m-1)(n+m+\ell-2) &\le \ell + (m-1)(2n-2)\\ &\le n - m + 2mn - 2n - 2m + 2\\ &\le 2mn - n - 3m + 2\\ &= (n-2)(2m-1) + m\\ &\le 1 + 2m(n-2) \le 1 + 2(n-1)(n-2),\end{aligned}$$

which completes the proof.

4 Application to Finite Prefix Codes

In this section we show how the previous result can be applied to the automaton associated to a finite prefix code.

A *prefix code* on the alphabet A is a set X of words on A such that no element of X is a prefix of another word of X.

A prefix code is *maximal* if it is not contained in another prefix code on the same alphabet. As an equivalent definition, a prefix code X is maximal if for any word u in A^* has a prefix in X or is a prefix of a word of X.

For a deterministic automaton $\mathcal{A}$ and an initial state i, the set $X_{\mathcal{A}}$ of labels of first return paths from i to i is a prefix code. If the automaton is complete, the prefix code is maximal.

Conversely, for any finite prefix code X, there exists a deterministic automaton $\mathcal{A}$ such that $X = X_{\mathcal{A}}$. Moreover, the automaton $\mathcal{A}$ can be supposed to be irreducible. If X is a maximal prefix code, the automaton $\mathcal{A}$ is complete.

The automaton $\mathcal{A}$ can be chosen as follows. The set of states is the set Q of prefixes of the words of X. The transitions are defined for $p \in Q$ and $a \in A$ by $p \cdot a = pa$ if pa is a prefix of a word of X, and by $p \cdot a = \varepsilon$ if $pa \in X$. This automaton, denoted $\mathcal{A}_X$ is a *decoder* of X. Let indeed α be a one-to-one map from a source alphabet B onto X. Let us add an output label to each edge of $\mathcal{A}_X$ in the following way. The output label of (p, a, q) is ε if $q \neq \varepsilon$ and is equal to $\alpha^{-1}(pa)$ if $q = \varepsilon$. With this definition, for any word $x \in X^*$, the output label of the path $i \xrightarrow{x} i$ is the word $\alpha^{-1}(x)$.

Let us show that, as a consequence of the fact that X is finite, the automaton $\mathcal{A}$ is additionally one-cluster with respect to any letter.

Indeed, let a be a letter and let C be the set of states of the form $i \cdot a^j$. For any state q, there exists a path $i \xrightarrow{u} q \xrightarrow{v} i$. We may suppose that i does not occur elsewhere on this path. Thus $uv \in X$. Since X is a finite maximal code, there is an integer j such that $ua^j \in X$. Then $q \cdot a^j = i$ belongs to C. This shows that $\mathcal{A}$ is one-cluster with respect to a.

A maximal prefix code X is *synchronized* if there is a word $x \in X^*$ such that for any word $w \in A^*$, one has $wx \in X^*$. Such a word x is called a *synchronizing word* for X.

Let X be a synchronized prefix code. Let $\mathcal{A}$ be an irreducible deterministic automaton with an initial state i such that $X_{\mathcal{A}} = X$. The automaton $\mathcal{A}$ is synchronized. Indeed, let x be a synchronizing word for X. Let q be a state of $\mathcal{A}$. Since $\mathcal{A}$ is irreducible, there is a path $i \xrightarrow{u} q$ for some $u \in A^*$. Since x is synchronizing for X, we have $ux \in X^*$, and thus $q \cdot x = i$. This shows that x is a synchronizing word for $\mathcal{A}$.

Conversely, let $\mathcal{A}$ be an irreducible complete deterministic automaton. If $\mathcal{A}$ is a synchronized automaton, the prefix code $X_{\mathcal{A}}$ is synchronized. Indeed, let x be a synchronizing word for A. We may assume that $q \cdot x = i$ for any state q. Then x is a synchronizing word for X.

Proposition 3. *Let X be a maximal synchronized prefix code with n codewords on an alphabet of size k. The decoder of X has a synchronizing word of length at most $O((n)^2)$.*

Proof. The automaton $\mathcal{A}_X$ is one-cluster. The number N of its states is the number of prefixes of the words of X. Thus $N = (n-1)/(k-1)$ since a complete k-ary tree with n leaves has $(n-1)(k-1)$ internal nodes. By Proposition 2, there is a synchronizing word of length at most $1 + 2(N-1)(N-2)$, whence $O((n)^2)$.

Example 1. Let us consider the following Huffman code $X = (00+01+1)(0+10+11)$ corresponding to a source alphabet $B = \{a, b, c, d, e, f, g, h, i\}$ with a probability distribution $(1/16, 1/16, 1/8, 1/16, 1/16, 1/8, 1/8, 1/8, 1/4)$. The Huffman tree is pictured in the left part of Figure 2 while the decoder automaton $\mathcal{A}_X$ is given in its right part. The word 010 is a synchronizing word of $\mathcal{A}_X$.

When the lengths of the codewords in X are not relatively prime, the automaton $\mathcal{A}_X$ is never synchronized (see Example of Figure 3). When the lengths of the codewords in X are relatively prime, the code X is not necessarily synchronized. However, there is always another Huffman code X' corresponding to the same length distribution which is synchronized by a result of Schützenberger [16]. One can even choose X' such that the underlying graph of $\mathcal{A}_X$ and $\mathcal{A}_{X'}$ are the same. This is a particular case of the road coloring theorem of due to Trahtman [18] (see also [2]). The particular case corresponding to finite prefix codes was proved before in [11].

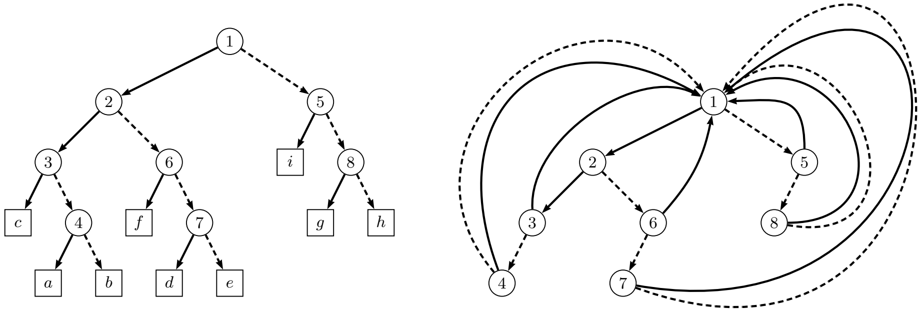


Fig. 2. A synchronized Huffman code X on the left and its decoder $\mathcal{A}_X$ on the right

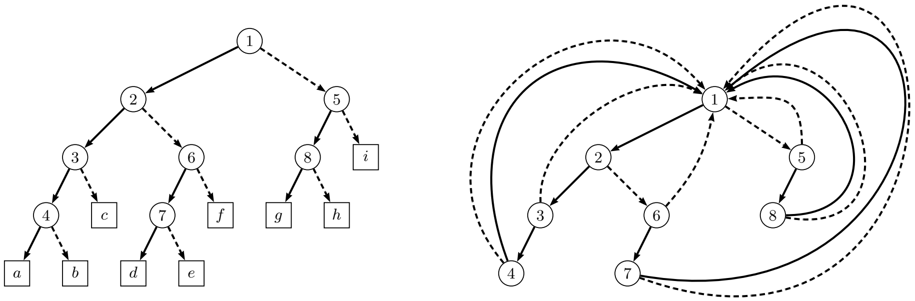


Fig. 3. A non synchronized Huffman code X on the left and its decoder on the right. The automaton on the right is not synchronized. Indeed, for any word w, the set of states reachable by w is either $\{1,3\}$, $\{2,4\}$, $\{1,5\}$, $\{1,6\}$, $\{2,7\}$ or $\{2,8\}$.

Our result guarantees that the Huffman decoder has a synchronizing word of length at most quadratic in the number of nodes of the Huffman tree.

References

1. Béal, M.-P.: A note on Černý's conjecture and rational series. preprint IGM 2003-05 (unpublished, 2003)
2. Béal, M.-P., Perrin, D.: A quadratic algorithm for road coloring. CoRR, abs/0803.0726 (2008)
3. Biskup, M.T.: Shortest synchronizing strings for huffman codes. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 120–131. Springer, Heidelberg (2008)
4. Carpi, A., D'Alessandro, F.: The synchronization problem for strongly transitive automata. In: Developments in Language Theory, pp. 240–251 (2008)
5. Černý, J., Poznámka, K.: Homogénnym experimentom s konecnými automatmi. Mat. fyz. čas SAV 14, 208–215 (1964)
6. Dubuc, L.: Sur les automates circulaires et la conjecture de Černý. RAIRO Inform. Théor. Appl. 32, 21–34 (1998)
7. Eilenberg, S.: Automata, languages, and machines, vol. A. Academic Press, A subsidiary of Harcourt Brace Jovanovich, Publishers, New York (1974); Pure and Applied Mathematics, vol. 58
8. Freiling, C.F., Jungreis, D.S., Théberge, F., Zeger, K.: Almost all complete binary prefix codes have a self-synchronizing string. IEEE Transactions on Information Theory 49, 2219–2225 (2003)
9. Kari, J.: A counter example to a conjecture concerning synchronizing words in finite automata. EATCS Bulletin 73, 146 (2001)
10. Kari, J.: Synchronizing finite automata on eulerian digraphs. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, pp. 432–438. Springer, Heidelberg (2001)
11. Perrin, D., Schützenberger, M.-P.: Synchronizing words and automata and the road coloring problem, in Symbolic Dynamics and its Applications. In: Walters, P. (ed.) American Mathematical Society, vol. 135, pp. 295–318. Contemporary Mathematics (1992)

12. Pin, J.-E.: Le problème de la synchronisation et la conjecture de Černý, thèse de 3ème cycle, Université Paris VI (1978)
13. Pin, J.-E.: Sur un cas particulier de la conjecture de Černý. In: Ausiello, G., Böhm, C. (eds.) ICALP 1978. LNCS, vol. 62, Springer, Heidelberg (1978)
14. Pin, J.-E.: On two combinatorial problems arising from automata theory. Annals of Discrete Mathematics, vol. 17, pp. 535–548 (1983)
15. Sakarovitch, J.: Éléments de théorie des automates, Éditions Vuibert (2003)
16. Schützenberger, M.-P.: On synchronizing prefix codes. Inform. and Control 11, 396–401 (1967)
17. Trahtman, A.N.: Synchronization of some DFA. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) TAMC 2007. LNCS, vol. 4484, pp. 234–243. Springer, Heidelberg (2007)
18. Trahtman, A.N.: The road coloring problem. Israel J. Math (to appear) (2008)
19. Trakhtman, A.: Some aspects of synchronization of DFA. J. Comput. Sci. Technol. 23, 719–727 (2008)
20. Volkov, M.V.: Synchronizing automata preserving a chain of partial orders. In: Holub, J., Žďárek, J. (eds.) CIAA 2007. LNCS, vol. 4783, pp. 27–37. Springer, Heidelberg (2007)
21. Volkov, M.V.: Synchronizing automata and the Černy conjecture. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 11–27. Springer, Heidelberg (2008)

Regular Languages Definable by Majority Quantifiers with Two Variables

Christoph Behle, Andreas Krebs, and Stephanie Reifferscheid*

Wilhelm-Schickard-Institut, Universität Tübingen, Sand 13, D-72076 Tübingen
{behlec,krebs,reiffers}@informatik.uni-tuebingen.de

Abstract. In this paper we consider the class of all regular languages definable by the extended majority quantifier and the order predicate but using only two variables. The main part of the paper is the presentation of a geometric method which is used to show that a given regular language cannot be defined by such formulas. Applying this method we can give a necessary condition in terms of an equation as well as an upper and a lower bound for the corresponding class of monoids. As a consequence we obtain that $\mathrm{FO} + \mathrm{MAJ}_2[<]$ does not contain $\mathrm{FO} + \mathrm{MOD}_2[<]$.

1 Introduction

Understanding the relation of the classes TC^0 and NC^1 is a fundamental problem in the theory of low level complexity classes. While it is well known that TC^0 is contained in NC^1, it is an open problem whether these two classes coincide. Methods developed so far have not succeeded to separate these classes although there is a promising three-way correspondence between circuits, algebra and logic (see [1], [2]). Barrington [3] proved that the word problem of any finite non-solvable group is complete for NC^1; thus regular languages play a key role in deciding this question. Moreover, it was recently shown in [4] that it is even sufficient to show that the word problem of such a group cannot be recognized by TC^0 circuits of almost linear size. Summing up: For a separation result one can concentrate on the regular languages of this subclass.

In this paper we consider the regular languages in a natural subclass of the logic class corresponding to TC^0 circuits of linear size (see [5]), namely the class of all regular languages definable by the extended majority quantifier and the order predicate but using only two variables, which we denote by $\widehat{\mathrm{MAJ}}_2[<] \cap \mathit{REG}$.

We first show that this class forms a language variety and thus has a corresponding monoid variety, which we denote by $\mathbf{Maj_2}$. Unfortunately we cannot give a complete algebraic description of $\mathbf{Maj_2}$ but we present non-trivial upper and lower bounds for it. The main contribution of this paper is an elementary geometric method based on the $\widehat{\mathrm{MAJ}}_2[<]$-formula of the language. We use

* The third author has been supported by Proyecto MTM2007-68010-C03-03, Ministerio de Educación y Ciencia and FEDER, Spain.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 91–102, 2009.

this method to show that a given language is not contained in $\mathrm{M\widehat{A}J}_2[<]$, allowing us to present an upper bound in terms of algebra without using deeper algebraic background. As a consequence of the upper bound we obtain that $\mathrm{FO} + \mathrm{MOD}_2[<]$ is not a subset of $\mathrm{M\widehat{A}J}_2[<] \cap \mathit{REG}$, while in the unrestricted case $\mathrm{FO} + \mathrm{MOD}[<] \subseteq \mathrm{M\widehat{A}J}[<] \cap \mathit{REG}$.

Structure of the paper: We start by giving the necessary background in logic and algebra, and state our results in Section 3. These are proved using a geometric method presented in Section 4. We conclude with a discussion on how our results fit in the general connections between logic and algebra.

2 Background in Logic and Algebra

We give a short introduction to logic over words and refer to the book of Straubing [1] for details and background.

Logic: Let $\mathcal{V}$ be a finite set of variables, a $\mathcal{V}$-structure is a string: $w = (w_1, \mathcal{V}_1)(w_2, \mathcal{V}_2) \ldots (w_n, \mathcal{V}_n) \in (\Sigma \times 2^{\mathcal{V}})^*$, where the $\mathcal{V}_i, 1 \leq i \leq n$ are pairwise disjoint and $\bigcup_{i=1}^n \mathcal{V}_i = \mathcal{V}$. For an alphabet Σ and a set of free variables $\mathcal{V}$ we denote the set of all $\mathcal{V}$-structures by $\Sigma^* \otimes \mathcal{V}$. We use $w_{x=i}$ to denote a $\mathcal{V}$-structure such that $x \in \mathcal{V}_i$. We denote by $|w|$ the length of w.

We define now the syntax of a $\mathrm{M\widehat{A}J}$-formula describing a language over an alphabet Σ. We will consider only the order predicate and monadic predicates. For two variables x, y the atomic formulas are either $x < y$, $P_k(x)$ for a numerical predicate P_k, or $Q_a(x)$ for $a \in \Sigma$. Recall that the truth value of a numerical predicate depends only on the length of the word and the positions of the variables, but not on the word itself.

Further we define the set of extended majority formulas recursively as follows: All atomic formulas are formulas and if φ and ψ are formulas then (i) $\varphi \wedge \psi$, (ii) $\varphi \vee \psi$ are formulas. For a formula φ we have that $\neg\varphi$ is a formula and finally for a variable x and formulas $\varphi_1, \ldots, \varphi_c$ we let $\mathrm{M\widehat{a}j}\, x\, \langle \varphi_1, \ldots, \varphi_c \rangle$ be a formula.

As usual we use the notions of free and bound variables. A variable is bound if it occurs in the scope of a quantifier, else it is a free variable.

Let φ be a formula whose variables are all contained in $\mathcal{V}$. For $w \in \Sigma^* \otimes \mathcal{V}$ we define inductively $w \models \varphi$ as follows: $w \models Q_a(x)$ iff w contains a letter (a, S) such that $x \in S$. $w \models x < y$ iff there are two letters $(a_i, S_i), (a_j, S_j)$ with $i < j$ and $x \in S_i$ and $y \in S_j$. To each monadic predicate P_k we have assigned a subset $P_k^{\mathcal{I}} \subseteq \mathbb{N}^2$, called the interpretation and $w \models P_k(x)$ iff $x \in S_i$ and $(i, |w|) \in P_k^{\mathcal{I}}$.

Finally we define what it means that $w \models \mathrm{M\widehat{a}j}\, x\, \langle \varphi_1, \ldots, \varphi_c \rangle$: Let $w_{x=i}$ be defined by $(w_1, \mathcal{V}'_1) \ldots (w_n, \mathcal{V}'_n)$, where $\mathcal{V}'_j := \mathcal{V}_j \setminus \{x\}$ for $j \neq i$ and $\mathcal{V}'_i := \mathcal{V}_i \cup \{x\}$. With the help of this definition we say that $w \models \mathrm{M\widehat{a}j}\, x\, \langle \varphi_1, \ldots, \varphi_c \rangle$ iff the following holds: $0 < \sum_{i=1}^{|w|} \sum_{j=1}^{c} \begin{cases} +1, & w_{x=i} \models \varphi_j \\ -1, & w_{x=i} \models \neg\varphi_j \end{cases}$.

For a set $\mathcal{P}$ of numerical predicates we denote by $\mathrm{M\widehat{A}J}[<, \mathcal{P}]$ the set of all extended majority formulas with numerical predicates in $\mathcal{P}$ and the order predicate. By $\mathrm{M\widehat{A}J}_2[<, \mathcal{P}]$ we denote the subset of $\mathrm{M\widehat{A}J}[<, \mathcal{P}]$ formulas, that are built using only two variables (which can be reused).

We want to point out that the extended majority quantifier coincides with the normal majority quantifier when quantifying over a single subformula, i.e. $w \models \widehat{\mathrm{Maj}}\, x \,\langle\varphi\rangle \iff w \models \mathrm{Maj}\, x\, \varphi$. The language described by φ is $\mathcal{L}(\varphi) = \{w \in \Sigma^* \otimes \mathcal{V} \mid w \models \varphi\}$. Finally we define how formulas describe a language $L \subseteq \Sigma^*$. Let φ be a majority formula with no free variable. Then for $w \in \Sigma^*$ we say $w \models \varphi$ iff $(w_1, \emptyset) \ldots (w_{|w|}, \emptyset) \models \varphi$. This means we identify words in Σ^* with $\mathcal{V}$-structures $\Sigma^* \otimes \emptyset$ in a natural way.

Algebra: Although the aim in this paper is the study of an algebraic counterpart of a special logic class, we need only a very limited amount of algebraic background. We use mainly that certain language classes correspond to certain monoid classes, and that the latter can be described via so-called identities, namely equations being valid for every monoid in the class (see below for a definition). We assume the reader to be acquainted with the very basic concepts of semigroup theory (such as monoid, morphism, divisor, direct product) and recall only a few definitions. We refer the interested reader to [6] and [7] for a comprehensive algebraic background; a short survey which includes all the necessary facts about monoids and emphasizes the connection to logic is [2], a detailed description of the triangle circuit classes, logic and algebra can be found in [1].

Let M be a finite monoid; recall that M recognizes a language L iff there is a subset $A \subset M$ and a morphism $h : \Sigma^* \to M$ such that $L = h^{-1}(A)$. To each regular language we can naturally assign a finite monoid $M(L)$, namely the smallest monoid recognizing L; this monoid is called the syntactic monoid of L. Unfortunately, this assignment is neither one-to-one nor onto, but by Eilenberg's theorem there is a natural bijection between varieties[1] of regular languages and pseudovarieties[2] of finite monoids.

Recall that an element $e \in M$ is idempotent iff $e^2 = e$, and that for each element $x \in M$ there exists a natural number i such that x^i is idempotent; we use the notation x^ω to denote $\omega(x)$ where $\omega : M \to M$ is the operation mapping each element x to its idempotent power.

Dealing with varieties of monoids we mainly use the description via identities – i.e. equations being valid for every monoid in the variety – in our proofs and give now a list of monoid varieties playing a crucial role in the meeting point between algebra, logic and formal languages. Again, the interested reader is referred to [2]: $\mathbf{G} = [\![x^\omega = 1]\!]$ is the class of all finite groups; $\mathbf{A} = [\![x^{\omega+1} = x^\omega]\!]$ is the class of all finite aperiodic monoids, i.e. finite monoids containing no groups; $\mathbf{Ab} = [\![x^\omega = 1, xy = yx]\!]$ is the class of all finite Abelian groups. Further (ordered by inclusion) $\mathbf{SL} = [\![xy = yx, x^2 = x]\!]$; $\mathbf{DA} = [\![(xyz)^\omega y (xyz)^\omega = (xyz)^\omega]\!]$; $\mathbf{DO} = [\![(xy)^\omega (yx)^\omega (xy)^\omega = (xy)^\omega]\!]$ and $\mathbf{DS} = [\![((xy)^\omega (yx)^\omega (xy)^\omega)^\omega = (xy)^\omega]\!]$.

Another important algebraic concept is that of the block product, and there is a crucial connection between taking the block product and nesting quantifiers in logic (this is introduced as the *block product/substitution principle* in

[1] A class of languages closed under Boolean operations, left and right quotients and inverse morphism between free monoids.

[2] A class of monoids closed under taking finite direct products and division; as usual we will write here variety instead of pseudovariety to ease notation.

[8]). Before giving the definition it should be noted that we mainly need the block product for standard argumentations, but not for the heart of the paper, namely the geometric method. Let $(M,+),(N,\cdot)$ be two monoids. The block product $M \square N$ is the monoid over the set $M^{N\times N} \times N$ with the multiplication $(f_1,n_1)(f_2,n_2) = (f_1 n_2 + n_1 f_2, n_1 \cdot n_2)$, where we have an action of N on $M^{N\times N}$ given by $(nfn')(x,y) = f(x\cdot n, n'\cdot y)$. For two varieties $\mathbf{V},\mathbf{W}$ the variety $\mathbf{V} \square \mathbf{W}$ is the smallest variety containing all monoids $M \square N$, where $M \in \mathbf{V}$ and $N \in \mathbf{W}$. A variety of this form is the variety $\mathbf{DA} \square \mathbf{G}$ which can also be characterized as follows ([9, Proof of Lemma 15]): For two idempotents e, f where e, f, ef are in the same D-class[3] we have ef is also an idempotent.

We will also deal with the class $\overline{\mathbf{Ab}}$ of all finite monoids containing only Abelian groups and the class $\mathbf{G}_{solv}$ of all finite solvable groups, where the latter can be ignored by all readers not familiar with the algebraic theory of languages.

Recall that we have the following connections to logic [10]: $\mathbf{SL}$ corresponds to FO_1; $\mathbf{Ab}$ corresponds to $\mathrm{MOD}_1[<]$; $\mathbf{G}_{solv}$ corresponds to $\mathrm{MOD}_2[<]$ and $\mathbf{DA} \square \mathbf{G}_{solv}$ corresponds to $\mathrm{FO} + \mathrm{MOD}_2[<]$.

3 Results

In this paper we study the languages in $\mathrm{M\widehat{A}J}_2[<] \cap REG$. We first show that this class forms a language variety (Theorem 1) and thus has a corresponding monoid variety by Eilenberg's theorem which we denote by $\mathbf{Maj_2}$. Unfortunately we cannot give a complete description of $\mathbf{Maj_2}$ (although we can do it for some subclasses of $\mathbf{Maj_2}$, see Corollary 1) but we present non-trivial upper and lower bounds for this class (Theorem 2 and Theorem 3). As a consequence of Theorem 3 we further obtain separation results of some logic classes (Corollary 2).

Theorem 1. *The languages described by* $\mathrm{M\widehat{A}J}_2[<]$ *formulas form a variety of languages. In particular, the languages in* $\mathrm{M\widehat{A}J}_2[<] \cap REG$ *form a variety of languages.*

For stating the lower bound we define the general n-wheel bicycle language, a generalization of the bicycle language , as follows: let $n \geq 2$, then a word $w \in \{a,b\}^*$ is in the n-wheel bicycle language iff for all prefixes w' of w holds $0 \leq \#_a(w') - \#_b(w') < n$. Note that this can also be seen as the subset of the Dyck language with one pair of parentheses where the number of unmatched parentheses is at most $n-1$. We denote the syntactic monoid of this language by B_n. Now let $\mathbf{X}$ be the smallest variety containing B_n, for all $n \geq 2$, and define $\mathbf{Y}$ as the smallest variety containing $\mathbf{Ab}$ and being closed under block product of $\mathbf{X}$ to the right, i.e. if $M_1 \in \mathbf{Y}$ and $M_2 \in \mathbf{X}$, then $M_1 \square M_2 \in \mathbf{Y}$.

Theorem 2. *If L is a language with $M(L) \in \mathbf{Y}$, then L is definable by a* $\mathrm{M\widehat{A}J}_2[<]$*-formula, i.e.* $\mathbf{Y} \subseteq \mathbf{Maj_2}$.

[3] Elements x, y of a finite monoid M are in the same D-class iff they are in the same J-class, i.e. iff $MxM = MyM$.

The proof is omitted. It can be easily given using the block product/substitution principle (see e.g. [2]). Loosely speaking one has to make clear how to check if the prefix (or suffix, respectively) of a word is in the generalized n-wheel bicycle language with a formula in $\widehat{\mathrm{MAJ}}_2[<]$ with a bounded variable.

Although the results of the previous two theorems might be of interest the proof methods are quite common. This is not the case for the following result that establishes a strict upper bound. Here the proof uses a new method based on geometric intuition to show a language is not definable in $\widehat{\mathrm{MAJ}}_2[<]$. This method works also whenever the considered varieties are given in terms of equations, and we use it to give an upper bound for $\mathbf{Maj_2}$.

Theorem 3. *If L is a regular language definable in $\widehat{\mathrm{MAJ}}_2[<]$, then $M(L) \in (\mathbf{DA} \mathbin{\Box} \mathbf{G}) \cap \overline{\mathbf{Ab}}$, moreover $\mathbf{Maj_2}$ is strictly contained in $(\mathbf{DA} \mathbin{\Box} \mathbf{G}) \cap \overline{\mathbf{Ab}}$.*

It seems reasonable to improve the upper bound, in fact we conjecture that $\mathbf{Maj_2}$ coincides with $\mathbf{Y}$. Although the algebraic characterization is not complete we have consequences in logic which in particular lead to a separation of the corresponding logic classes (see Corollary 2).

Though the upper and lower bound do not coincide, we can give a complete characterization if we focus on special subclasses of the regular languages:

Corollary 1 (Algebra Corollary)

- $\mathbf{Maj_2} \cap \mathbf{G} = \mathbf{Ab}$,
- $\mathbf{Maj_2} \cap \mathbf{DS} = \mathbf{DO} \cap \overline{\mathbf{Ab}}$.

The upper bound given above is somewhat odd since it shows that the majority quantifier behaves differently than quantifiers considered more frequently like FO, MOD, GROUP. The following corollary shows that the usual inclusion, which is preserved for these quantifiers when considering restrictions, is not valid for the case of majority logic:

Corollary 2 (Logic Corollary)

- $\widehat{\mathrm{MAJ}}_2[<] \cap \mathit{REG} \subsetneq \widehat{\mathrm{MAJ}}_3[<] \cap \mathit{REG}$,
- $\widehat{\mathrm{MAJ}}_2[<] \cap \mathit{REG} \not\supseteq \mathrm{FO} + \mathrm{MOD}_2[<]$,
- $\widehat{\mathrm{MAJ}}_2[<] \cap \mathit{REG} \subsetneq \mathrm{FO} + \mathrm{MOD}[<]$.

4 Geometry

We start with the following question: Given a $\widehat{\mathrm{MAJ}}_2[<]$ formula φ of quantifier depth one with one free variable x, and two words w_1, w_2; for which positions does φ have the same value? Formally how can we describe the tuples (w_1, i, w_2, i') such that $(w_1)_{x=i} \models \varphi \iff (w_2)_{x=i'} \models \varphi$?

For the sake of simplicity we will only consider words over the binary alphabet, although all proofs also work over an arbitrary finite alphabet.

One idea, which allows us to use geometry in the proof, is to interpret words as paths in a plane. To state this correspondence as well as the following results we need some technical definitions which are mostly induced by the situation in $\mathbb{R}^2$.

When speaking of a vector we refer to an element in $\mathbb{N}_0 \times \mathbb{N}_0$, and for two vectors $x = (x_1, x_2)$ and $y = (y_1, y_2)$ we define $x \leq y$ iff $x_1 \leq y_1$ and $x_2 \leq y_2$. Further we call two vectors $x = (x_1, x_2)$ and $y = (y_1, y_2)$ adjoint iff $|x_1 - y_1| + |x_2 - y_2| = 1$.

For a word $w \in \{a, b\}^*$ we denote by $\#_a(w)$ ($\#_b(w)$) the number of a's (b's) in w and by $\#(w)$ the vector $(\#_a(w), \#_b(w))$. For $1 \leq i \leq |w|$ we use $w_{<i}$ ($w_{\leq i}$) to denote the prefix of w of length $i - 1$ (of length i) and analogously $w_{>i}$ and $w_{\geq i}$ to denote the suffix of length $i - 1$ (of length i).

We can identify a word w of length $|w| = n$ with $\#w = (n_a, n_b)$ with a path in $\mathbb{N}_0 \times \mathbb{N}_0$ from $(0, 0)$ to (n_a, n_b). We do this by taking the sequence of vectors $v_0, v_1, \ldots v_n$, where $v_0 = (0, 0)$ and $v_i = \#(w_{\leq i})$. Note that this implies $v_{i-1} \leq v_i$ and v_{i-1} and v_i are adjoint for $1 \leq i \leq n$. On the other hand, we can assign to each sequence of vectors $v_0, v_1, \ldots v_n$ with $v_0 = (0, 0)$ and $v_n = (n_a, n_b)$ a unique word w with $\#w = (n_a, n_b)$ if the following holds: For $1 \leq i \leq n_a + n_b$ we have $v_{i-1} \leq v_i$ and v_{i-1} and v_i are adjoint.

Derived from the geometric view we define for constants α, β, γ the i-th point of a path w to be in the half-plane $H(\alpha, \beta, \gamma)$ iff

$$\alpha \#_a(w_{<i}) + \beta \#_b(w_{<i}) + \gamma > 0. \quad (*)$$

Finally we need the definition of an interval in $\mathbb{N}_0 \times \mathbb{N}_0$. For vectors $v = (v_1, v_2)$ and $v' = (v'_1, v'_2)$ with $v \leq v'$ we define the interval $[v, v']$ to be the set of all vectors u with $v \leq u \leq v'$. The size of the interval is given by the tuple $(v'_1 - v_1, v'_2 - v_2)$.

As mentioned above our aim is to develop a method to show a language is not definable in $\widehat{\mathrm{MAJ}}_2[<]$. In order to determine which words cannot be distinguished by a $\widehat{\mathrm{MAJ}}_2[<]$ formula we associate to each formula and to each tuple (n_a, n_b) a finite set of half-planes.

We begin by looking at some examples. Given the $\widehat{\mathrm{MAJ}}_2[<]$-formula $\varphi(x) = \widehat{\mathrm{Maj}}\ y\ Q_a(y)$, assume that we have a word w with $(n_a, n_b) = \#(w)$, then $w_{x=i} \models \varphi \iff n_a > n_b$. So this formula depends only on the absolute number of a's and b's in the word, not the order of the letters nor the position i of x.

Thus we look at a more complicated example. Let $\varphi(x) = \widehat{\mathrm{Maj}}\ y\ Q_a(y) \wedge y < x$, then $w_{x=i} \models \varphi$ iff $\#_a(w_{<i}) > \#_a(w_{\geq i}) + n_b$. We can rewrite this formula to $2 \cdot \#_a(w_{<i}) - n_a - n_b > 0$.

Now let $\varphi(x)$ be any extended majority formula of depth 1, we will show that we can determine the truth-value of $\varphi(x)$ for $x = i$ by checking the letter at position i and evaluating a fixed set of inequalities of the form $\alpha \cdot \#_a(w_{<i}) + \beta \cdot \#_b(w_{<i}) + \gamma > 0$.

In the general case the formula $\varphi(x)$ is a Boolean combination of formulas of the form $\widehat{\mathrm{Maj}}\ y\ \langle \varphi_1, \ldots, \varphi_l \rangle$, where each φ_k is a Boolean combination of $Q_a(x), Q_b(x), Q_a(y), Q_b(y), x < y,\ y < x$. First we look at the case $\varphi(x) = \widehat{\mathrm{Maj}}\ y\ \langle \varphi_1, \ldots, \varphi_l \rangle$. Depending on the letter w_i, we will give now a finite set of inequalities of the form above. Let us assume that $w_i = a$. Then $Q_a(x)$ is true and $Q_b(x)$ is false hence we may assume that each φ_k is a Boolean combination of $Q_a(y), Q_b(y), x < y, y < x$. We can rewrite each of these formulas as a disjunction of the formulas $Q_a(y) \wedge y < x, Q_a(y) \wedge y > x$, $Q_b(y) \wedge y < x, Q_b(y) \wedge y > x$. Note we assumed $w_i = a$, so we do not need formulas of the form $Q_a(y) \wedge y = x$ or $Q_b(y) \wedge y = x$, since they can replaced by true or false.

By definition $w_{x=i} \models \widehat{\text{Maj}}\, y\, \langle \varphi_1, \dots, \varphi_l \rangle$ iff

$$0 < \sum_{j=1}^{|w|} \sum_{k=1}^{l} \begin{cases} +1, & w_{x=i,y=j} \models \varphi_k \\ -1, & w_{x=i,y=j} \models \neg\varphi_k \end{cases} .$$

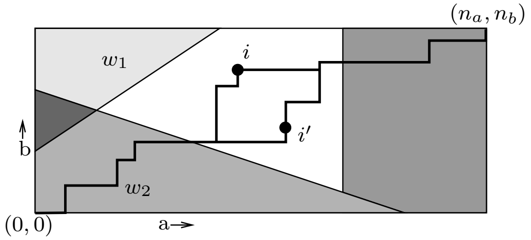
Since the φ_k's are of the form described above, the value of $\varphi_k \models w_{x=i,y=j}$ depends only on the letter at position j and whether j is smaller than/bigger than/equal to i. Thus we can split the sum corresponding to these conditions and obtain integers $\alpha_p, \beta_p, \alpha_s, \beta_s, \gamma_*$, acting as weights, such that $w_{x=i} \models \widehat{\text{Maj}} y \langle \varphi_1, \dots, \varphi_l \rangle$ iff $\alpha_p \#_a(w_{<i}) + \beta_p \#_b(w_{<i}) + \alpha_s \#_a(w_{>i}) + \beta_s \#_b(w_{>i}) + \gamma_* > 0$.

Since we assumed that $w_i = a$, the number of a's in the suffix $\#_a(w_{>i}) = n_a - 1 - \#_a(w_{<i})$ and $\#_b(w_{>i}) = n_b - \#_b(w_{<i})$. Thus we let $\alpha = \alpha_p - \alpha_s$, $\beta = \beta_p - \beta_s$, $\gamma = \gamma_* + (\alpha_s + 1) \cdot n_a + \beta_s \cdot n_b$, and obtain the inequality $\alpha \cdot \#_a(w_{<i}) + \beta \cdot \#_b(w_{<i}) + \gamma > 0$.

Analogously we find an inequality in the case $w_i = b$. Thus to decide if $w_{x=i} \models \varphi$ we need to check the letter at position i and the two inequalities found as above.

Now if φ is an arbitrary extended majority formula of depth 1 we might have more than two inequalities, since we allow finite Boolean combinations of the formulas considered above. But in any case this only leads to finitely many inequalities. Since each of these inequalities describes a half-plane we get:

Lemma 1. *Let φ be a formula in $\widehat{\text{MAJ}}_2[<]$ of depth one with one free variable x. Then there is a constant $c \in \mathbb{N}$ such that for all $n_a, n_b \in \mathbb{N}$ there exist half-planes $h_1, \dots, h_c$ with the following property: For $v, w \in \Sigma^*$ with $\#(v) = \#(w) = (n_a, n_b)$ and $1 \leq i, j \leq n_a + n_b$ holds: If $v_i = w_j$ and $\#(v_{\leq i}) \in h_l \Leftrightarrow \#(w_{\leq j}) \in h_l$ for all $1 \leq l \leq c$, then $v_{x=i} \models \varphi \Leftrightarrow w_{x=j} \models \varphi$.*



The paths of two words and positions for some sample half-planes

In particular if the paths of two words v, w with $\#(v) = \#(w)$ differ only in an interval that is contained in the same half-planes, then for all positions i, j such that $\#(v \leq i), \#(w_{\leq j})$ are in the interval we have: $v_{x=i} \models \varphi \iff w_{x=j} \models \varphi$ if $v_i = w_j$.

In order to apply the argument above we first show that for fixed numbers c, n'_a, n'_b we can always find n_a, n_b such that for any c half-planes we can find an interval of size (n'_a, n'_b) where all points are contained in exactly the same set of half-planes.

Lemma 2. *Given the natural numbers c, n'_a, n'_b, and an interval of size at least $(2^c \cdot n'_a, 2^c \cdot n'_b)$, for every set $\{h_1, \ldots, h_c\}$ of half-planes, we can find a subinterval of size (n'_a, n'_b) that is contained in exactly the same half-planes.*

Proof. We split the interval into four quadrants each of size $(2^{c-1}n'_a, 2^{c-1}n'_b)$ and pick a half-plane. Since a line can only intersect three quadrants at most there is one quadrant that is completely contained in the half-plane or completely outside. By induction on the number of half-planes we get the result.

Main idea: We have now developed the necessary tools to present our main lemma used to prove a language being not in $\mathrm{M\widehat{A}J_2}[<]$. The intuition behind this technique is the following: Pick φ in $\mathrm{M\widehat{A}J_2}[<]$, and find the finite set of half-planes corresponding to all subformulas of depth 1 of φ. For sufficiently large words we can find an interval of any desired size, that is completely contained in exactly the same set of half-planes. Take two words w_1, w_2 that differ only in that interval, i.e. have a common prefix and suffix. Then the truth value of every subformula of depth 1 of φ with the free variable x pointing to a position i inside the interval, depends only on the letter at position i. Thus the subformula can be replaced by the Q predicate if the free variable points to a position inside the interval. Doing induction on the depth of the formula we show that a given language is recognized by a given formula of depth d if and only if a certain subset of the language (consisting of all words with special prefixes and special suffixes) is recognized by a certain formula of depth one. The problem is now how to handle the prefixes and the suffixes.

More detailed: Assume we have a formula φ that recognizes a language L, and $\psi_1, \ldots, \psi_c$ are the subformulas of depth 1 of φ. We assume each of ψ_k has a free variable x. By Lemma 2 we can find an interval that is contained in the same set of half-planes corresponding to the formulas $\psi_1, \ldots, \psi_c$. Fix a prefix that leads to the interval, and a suffix from the end of the interval to the end of the rectangle. If the free variable x is assigned to a position inside the interval the truth value of each $\psi_k(x)$ is either constant or equivalent to $Q_a(x)$ or $Q_b(x)$. Ignoring the way we handle the prefix and suffix positions, we could replace the subformulas $\psi_k(x)$ by true, false, $Q_a(x)$, or $Q_b(x)$, and thus get a formula of lower depth that "inside the interval" recognizes the same language as φ. Iterating this leads to a formula of depth one.

The difficulty of this approach is to find suitable prefixes and suffixes, such that the languages restricted to the words with the common prefix and suffix are still "hard" enough. Note that the choice of prefixes/suffixes heavily depends on the considered language. As we will see later, for group languages we have a big choice while for the bicycle language our choice is very limited.

Before finally stating our main lemma we need some more notation. Given two languages L_1 and L_2 with $L_1 \cap L_2 = \emptyset$, we say φ separates L_1 from L_2 iff for all $w \in L_1 \cup L_2 : w \models \varphi \iff w \in L_1$. Note that this definition says nothing about words that are neither in L_1 nor L_2. For $k \in \mathbb{N}$ we set

$$D_{kn} = \{w \in \{a,b\}^* \mid \#_a(w) = kn, \#_b(w) = n\}$$

and $D_k = \bigcup_n D_{kn}$. Since in the following proofs we will choose prefixes and suffixes depending on the number of a's and b's, we will intersect the language with D_k – for a suitable k – to ensure that the number of a's and b's is determined by the length of the word. We will use D instead of D_k if k is understood or if the assertion is true for any $k \in \mathbb{N}$.

Given two families of words $p = (p_n)_{n\in\mathbb{N}}$ and $s = (s_n)_{n\in\mathbb{N}}$ and a language $L \subseteq D_k = D$ we define ${}_pL_s = \{w \in L \mid \exists w' : w = p_{|w|}w's_{|w|}\}$, we require here that $(kn, n) - \#(p_ns_n) \to (\infty, \infty)$ for $n \to \infty$. So we take the language L and consider only words that have a certain prefix and suffix depending on the length of the word. To ease dealing with the prefix and suffix families we introduce some numerical predicates that help in the induction. We say a unary numerical predicate $u(x)$ is *good* for p, s iff $u(x)$ is constant for all assignments $x = i$ if $|p_n| < i \leq n - |s_n|$. So the good predicates allow us to check the prefix and the suffix, while they do not help to recognize L. Lemma 1 remains true in the presence of good predicates iff the compared words have the same prefix and suffix. By a simple computation one obtains that we have only to modify the constant γ in equation $(*)$ for the half-planes.

The following lemma can be seen as induction step (when doing induction on the depth of the formula): Assume we have two languages $\tilde{L}_1$ and $\tilde{L}_2$ that for every word length n have already a fixed prefix p_n and suffix s_n and we assume that we can separate them by an extended majority formula. We show that we can extend the prefix by p'_n and the suffix by s'_n such that the languages with the new prefix $pp' = (p_np'_n)_n$ and suffix $s's = (s'_ns_n)_n$ can be separated by an extended majority formula of lower depth. We show that we have a certain choice of extending the prefix and suffix by restricting the extension to sets of certain words P_n, S_n. Recall, that in the geometric interpretation for fixed word length n the prefix leads to a certain interval, in which the innermost formulas depend only on the letter at the position the free variable points to; similar, the suffix can be seen as a path from the end of this interval to (kn, n).

Lemma 3. *Let $L_1, L_2 \subseteq D = D_k$ be two languages, and $p = (p_n)_n, s = (s_n)_n$ be two families of words. Let $\mathcal{U}$ be a set of good predicates for p, s, and φ be a formula in $\widehat{\mathrm{MAJ}}_2[<,\mathcal{U}]$ of depth $d > 1$ that separates ${}_p(L_1)_s$ from ${}_p(L_2)_s$.*

Further let $(P_n)_n$ and $(S_n)_n$ be a family of languages such that there is a family $(I_n)_n$ of intervals with growing size such that for every point $v \in I_n$, there is a word $\bar{p}_n \in P_n, \bar{s}_n \in S_n$ such that $\#(p_n\bar{p}_n) = v$ and $\#(\bar{s}_ns_n) = (kn, n) - v$.

Then there are two families of words $p' = (p'_n)_n, s' = (s'_n)_n$ with $p'_n \in P_n$, $s'_n \in S_n$ and $p_np'_ns'_ns_n \in D$, such that ${}_{pp'}(L_1)_{s's}$ can be separated from ${}_{pp'}(L_2)_{s's}$ by a formula φ' of depth $d-1$ in $\widehat{\mathrm{MAJ}}_2[<,\mathcal{U}']$, where $\mathcal{U}'$ is a set of good predicates for $pp', s's$.

Proof. By assumption there is a family of intervals with growing size which we denote by $(I_n)_n$. Also for each n we denote the set of half-planes corresponding to the innermost formulas of φ by $(H_n)_n$. By Lemma 2 we can find a family of intervals $(I'_n)_n$ with size (n'_a, n'_b) where $n'_a = kn'_b$ with $I'_n \subseteq I_n$ and every point in I'_n is in exactly the same half-planes of H_n. Since the size of $(I_n)_n$ grows arbitrary we can assume the same for $(I'_n)_n$, again by Lemma 2.

By the condition above we have a path $p'_n \in P_n$ from $(0,0)$ to the beginning of the interval I'_n. Also we have a path $s'_n \in S_n$ from the end of the interval I'_n to (kn, n). Because $\#(p_n p'_n s'_n s_n) = (kn, n) - (n'_a, n'_b)$ where (n'_a, n'_b) is the size of I'_n we know that $p_n p'_n s'_n s_n \in D$.

Now we know that all innermost formulas $\psi(x)$ for words in ${}_{pp'}(L_1 \cup L_2)_{s's}$ depend only on the length of the word and the position of x and the letter at the position of x. In particular they are constant except for the common prefix and suffix. Hence we can find good predicates for $pp', s's$ such that we can replace the innermost formulas of φ by a Boolean combination of these predicates and the Q predicates to obtain a formula φ' of lower depth.

The condition $p_n p'_n s'_n s_n \in D$ in the lemma above ensures that for all w' with $p_n p'_n w' s'_n s_n \in {}_{pp'}(L_1 \cup L_2)_{s's}$ we have $w' \in D$. We need this in order to use the lemma in the inductions below.

It turns out that the previous lemma is not applicable for certain languages, like the bicycle $(ab)^*$ (or generalizations of it, which can be seen as finite counter languages), where we have only very limited freedom in choosing the prefix and suffix, without making the remaining language trivial. On the other hand, languages like this, more detailed the languages in the class **X** defined in Section 3, seem to be the only languages we cannot handle with our method. So, since these languages can be described in $\widehat{\mathrm{MAJ}}_2[<]$we conjecture that we in fact have equality, i.e. $\mathbf{Maj_2} = \mathbf{Y}$.

Lemma 3 is our main tool to prove our results. We will first show that all groups definable are Abelian:

Lemma 4. *Let $G \in \mathbf{Maj_2}$ be a group, then G is Abelian.*

Proof. Assume by contradiction that G is non-Abelian, but $G \in \mathbf{Maj_2}$, so there is a language with syntactic monoid G that is recognized by a formula φ in $\widehat{\mathrm{MAJ}}_2[<]$. Pick two elements g_a, g_b of the group with $g_a g_b \neq g_b g_a$. Let $h : \Sigma^* \to G$ be defined by $h(a) = g_a$, $h(b) = g_b$, and $L_1 = h^{-1}(g_a g_b) \cap D$, $L_2 = h^{-1}(g_b g_a) \cap D$ for $D = D_1$. Let $o = |G|$ denote the order of G.

We will apply Lemma 3 in the induction, so we suppose we have a family of prefixes p and of suffixes s with $p_n \in (a^*(b^o)^*)^* a^* b^*$, $s_n \in a^* b^* (a^*(b^o)^*)^*$ for all n. For the formula φ that separates L_1, L_2 without prefixes and suffixes we can pick $p_n = s_n = \epsilon$ for all n. This yields our induction basis. By contradiction we assume to have a formula φ that separates ${}_p(L_1)_s$ from ${}_p(L_2)_s$.

Suppose φ has depth 1. We pick the word length $n \equiv 2 \mod o$ such that $n - |p_n s_n| > 2o + 2$ (note that this is possible since we can find an interval I of arbitrary size by Lemma 2). Then $p_n = a^{x_1}(b^o)^{y_1} a^{x_2}(b^o)^{y_2} a^{x_3} \dots (b^o)^{y_{c-1}} a^{x_c} b^{y_c}$, $s_n = b^{y'_1} a^{x'_1}(b^o)^{y'_2} a^{x'_2} \dots (b^o)^{y'_{c'}} a^{x'_{c'}}$, for suitable natural numbers $x_i, x'_i, y_i, y'_i, c, c'$ depending on n. We can find natural numbers $r_1, \dots, r_4$ such that for

$$w_1 = p_n b^{or_1 - y_c} a^{or_2 - \sum x_i} aba^{or_3 - \sum x'_i} b^{or_4 - y'_1} s_n,$$

$$w_2 = p_n b^{or_1 - y_c} a^{or_2 - \sum x_i} baa^{or_3 - \sum x_i'} b^{or_4 - y_1'} s_n,$$

holds $w_1 \in L_1$ and $w_2 \in L_2$; but $w_1 \models \varphi \iff w_2 \models \varphi$, by Lemma 1; contradiction.

Now we assume that φ has depth > 1. We will apply Lemma 3. Let I be the growing set of intervals I_n determined by $\#(p_n)+(0,o)$ and $(n,n)-\#(s_n)-(0,o)$. This family is a growing family of intervals since $(n,n)-\#(p_n s_n) \to (\infty,\infty)$ for $n \to \infty$. Also we choose the set of prefixes $P_n = b^{o-r}a^*b^*$, where $r = y_c \mod o$.

Similarly we choose the set of suffixes $S_n = b^*a^*b^{o-t}$, where $t = y_1' \mod o$. We apply the previous lemma and get a formula φ' that separates ${}_{pp'}(L_1)_{s's}$ from ${}_{pp'}(L_2)_{s's}$. We need to show that $p_n p_n' \in (a^*(b^o)^*)^*a^*b^*$ and $s_n' s_n \in a^*b^*(a^*(b^o)^*)^*$, but this is clear by the choice of P_n and S_n.

Applying Lemma 3 we can now give an identity, which is fulfilled by all monoids in $\mathbf{Maj_2}$. This is used later to show that the class $\mathbf{Maj_2}$ is contained in $\mathbf{DA} \Box \mathbf{G}$.

Lemma 5. *Let* $M \in \mathbf{Maj_2}$ *then* $\forall x,y \in M\colon\ (xy^\omega)^\omega(y^\omega x)^\omega = ((xy^\omega)^\omega(y^\omega x)^\omega)^\omega$.

Proof. Pick any M that does not fulfill this property, then there are x,y in M such that: $(xy^\omega)^\omega(y^\omega x)^\omega \neq ((xy^\omega)^\omega(y^\omega x)^\omega)^\omega$. We assume $y = y^\omega$ then the inequality is equivalent to $(xy)^\omega(yx)^\omega \neq (xy)^\omega(yx)^\omega(xy)^\omega(yx)^\omega$. Further we can modify the right side $(xy)^\omega(yx)^\omega(xy)^\omega(yx)^\omega = (xy)^{2\omega-1}yxxy(yx)^{2\omega-1}$, since $yy = y$. We get the equation $(xy)^{2\omega-1}xyyx(yx)^{2\omega-1} \neq (xy)^{2\omega-1}yxxy(yx)^{2\omega-1}$. Now the proof is similar to the previous lemma, show that the $xyyx$ can be replaced by $yxxy$ for suitable prefix and suffix.

Again we apply our main lemma to show that a certain language that is definable by a FO+MOD formula of depth 2 is not recognizable by a monoid in $\mathbf{Maj_2}$.

Lemma 6. *Let* $R = ((aa)^*b)^*(aa)^*$*, then the syntactic monoid of* R *is not contained in* $\mathbf{Maj_2}$.

Proof. Let $P = ((aa)^*b^*)^*a^*$, $S = (b^*(aa)^*)^*$. In order to prove the result we need to show that the languages $L_1 = ((aa)^*b^*)^*(aa)^*aab(aa)^*(b^*(aa)^*) \cap D_3$ and $L_2 = ((aa)^*b^*)^*(aa)^*aba(aa)^*(b^*(aa)^*) \cap D_3$ cannot be separated by a $\mathrm{M\widehat{A}J}_2[<]$ formula. The proof is the same as in Lemma 4.

Now we can prove the upper bound, i.e. that $\mathbf{Maj_2} \subsetneqq (\mathbf{DA} \Box \mathbf{G}) \cap \overline{\mathbf{Ab}}$:

Proof (Proof of Theorem 3). Let $M \notin \mathbf{DA} \Box \mathbf{G}$. Then there are idempotents $e,f \in M$ in one D-class such that ef is in the same D-class but not an idempotent. Since e,f,ef are in the same D-class, there are s,t such that $e = efs$ and $tef = f$. It follows $fs = te$. Let $y = (fs)^\omega$ and $x = ef$. But then Lemma 5 fails, since $(xy^\omega)^\omega(y^\omega x)^\omega = ef$ is not an idempotent. It follows that $\mathbf{Maj_2} \subseteq \mathbf{DA} \Box \mathbf{G}$. Since a subgroup of a monoid M is always a divisor of M we know by Lemma 4 that $\mathbf{Maj_2} \subseteq \overline{\mathbf{Ab}}$ and we obtain $\mathbf{Maj_2} \subseteq \mathbf{DA} \Box \mathbf{G} \cap \overline{\mathbf{Ab}}$.

We know R is not describable by $\mathrm{M\widehat{A}J}_2[<]$, hence the syntactic monoid of R is not in $\mathbf{Maj_2}$ but it is in $\mathbf{SL} \Box \mathbf{Ab} \subset \mathbf{DA} \Box \mathbf{G}$. Thus we can conclude $\mathbf{Maj_2} \subsetneqq (\mathbf{DA} \Box \mathbf{G}) \cap \overline{\mathbf{Ab}}$.

5 Discussion

The main achievement of this paper is the development of a method based on geometric intuition to show that a given language is not in $\widehat{\mathrm{MAJ}}_2[<]$. We think this method is promising in several ways. First of course, it allows to cope with majority logic and we think it should be applicable to every regular language outside $\widehat{\mathrm{MAJ}}_2[<]$. Additionally this method can be applied whenever a variety is given in terms of equations as demonstrated in Lemma 5 and it should be possible to extend it to non-regular languages, too. As a result we obtained that $\mathrm{FO} + \mathrm{MOD}_2[<]$ is not a subclass of $\widehat{\mathrm{MAJ}}_2[<]$.

Our motivation was to study the regular languages in $\widehat{\mathrm{MAJ}}_2[<]$. We have shown that these languages form a variety and hence have an algebraic counterpart by Eilenberg's theorem. We have not been able to give an exact characterization but our current knowledge of this class leads us to our conjecture that the target variety is **Y**, a variety built from Abelian groups and natural generalizations of the bicycle monoid. To our information **Y** has not been characterized in other terms which is one obstacle to obtain a better upper bound. Any further knowledge about this variety or varieties between **Y** and our upper bound, for example in terms of equations, could be helpful to obtain improvements.

Another direction of research might be to see if our method can be applied to other logic classes as well. Since it makes not much use of algebra this might lead to simpler proofs of known results in this area.

Acknowledgments. We thank Pascal Tesson for valuable discussions on the delicate behavior of varieties of monoids.

References

1. Straubing, H.: Finite Automata, Formal Logic, and Circuit Complexity. Birkhäuser, Boston (1994)
2. Tesson, P., Thérien, D.: Logic meets algebra: the case of regular languages. Logical Methods in Computer Science 3 (2007)
3. Barrington, D.A.M.: Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. J. Comput. Syst. Sci. 38, 150–164 (1989)
4. Allender, E., Koucký, M.: Amplifying lower bounds by means of self-reducibility. In: IEEE Conference on Computational Complexity, pp. 31–40 (2008)
5. Behle, C., Krebs, A., Mercer, M.: Linear circuits, two-variable logic and weakly blocked monoids. In: Kučera, L., Kučera, A. (eds.) MFCS 2007. LNCS, vol. 4708, pp. 147–158. Springer, Heidelberg (2007)
6. Pin, J.E.: Varieties of formal languages. Plenum, London (1986)
7. Howie, J.M.: Fundamentals of Semigroup Theory. Clarendon Press, Oxford (1995)
8. Thérien, D., Wilke, T.: Nesting until and since in linear temporal logic. Theory Comput. Syst. 37, 111–131 (2004)
9. Tesson, P., Thérien, D.: The computing power of programs over finite monoids. Electronic Colloquium on Computational Complexity (ECCC) 8 (2001)
10. Straubing, H., Thérien, D.: Regular languages defined by generalized first-order formulas with a bounded number of bound variables. Theory Comput. Syst. 36, 29–69 (2003)

The Inclusion Problem of Context-Free Languages: Some Tractable Cases[*]

Alberto Bertoni[1], Christian Choffrut[2], and Roberto Radicioni[3]

[1] Università degli Studi di Milano
Dipartimento di Scienze dell'Informazione
Via Comelico 39, 20135 Milano, Italy
bertoni@dsi.unimi.it

[2] L.I.A.F.A. (Laboratoire d'Informatique Algorithmique,
Fondements et Applications),
Université Paris VII, 2 Place Jussieu, 75221 Paris - France
cc@liafa.jussieu.fr

[3] Università degli Studi dell'Insubria
Dipartimento di Informatica e Comunicazione
Via Mazzini 5, 21100 Varese, Italy
radicion@dsi.unimi.it

Abstract. We study the problem of testing whether a context-free language is included in a fixed set L_0, where L_0 is the language of words reducing to the empty word in the monoid defined by a complete string rewrite system. We prove that, if the monoid is cancellative, then our inclusion problem is polynomially reducible to the problem of testing equivalence of straight-line programs in the same monoid. As an application, we obtain a polynomial time algorithm for testing if a context-free language is included in a Dyck language (the best previous algorithm for this problem was doubly exponential).

Introduction

In this paper we analyze problems of this kind: given a fixed language L_0, to decide whether the language L_G generated by a context-free grammar is contained in L_0. It is known that the class of context-free grammars generating languages L_0 for which the problem is decidable is not recursive ([7]). This problem is undecidable even for some deterministic context-free language L_0, while it turns out to be decidable if L_0 is superdeterministic ([6]); in this case a doubly exponential time algorithm has been shown.

Since the class of superdeterministic languages includes the AFDL (Abstract Family of Deterministic Languages) closure of Dyck languages, it is decidable if a context-free grammar generates a language which is included in a Dyck language, i.e., the language of well-parenthesized expressions over a fixed but arbitrary subset of parentheses. The case with a single type of parenthesis has

[*] Partially supported by Project M.I.U.R. PRIN 2007–2009: Mathematical aspects and forthcoming applications of automata and formal languages.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 103–112, 2009.

been studied by Knuth in [8], while a doubly exponential algorithm for the general case can be deduced from an algorithm proposed in [1].

In this work, we exhibit a class of languages L_0 for which the inclusion problem can be solved efficiently. In particular, given a fixed complete string rewrite system, we study the problem of verifying whether a context-free language is included in the set L_0 of words which reduce to the empty word. We prove that, if the string rewrite system is complete and defines a cancellative monoid, then the problem polynomially reduces to testing the compressed equivalence problem on the monoid, i.e., whether two straight-line programs evaluate the same element of the monoid. Straight-line programs are a flexible compressed representation of strings; the compressed equivalence problem has been deeply studied, in particular in the case of 2-homogeneous and 2-monadic string rewrite systems ([11]). In the case of 2-homogeneous string rewrite systems, equivalence is solvable in polynomial time if and only if the rewrite system is N-free, unless P=NP ([11]).

By applying the previous results, we obtain a polynomial algorithm for testing inclusion in the case L_0 belongs to a class of languages that extends that of "Dyck-like" languages. This result also extends to the noncancellative monoids defined by N-free 2-homogeneous string rewrite systems; on the other hand, for 2-homogeneous non N-free rewrite systems, the inclusion problem is at least coNP-hard.

Furthermore, we study the more general case of complete unitary string rewrite systems ([3]), that is, systems for which all rules are of the form $u \to 1$, and prove that the inclusion of a regular language in a regular set over the generated monoid can be tested again in polynomial time.

1 Definitions

Given an alphabet Σ, the free monoid it generates is denoted by Σ^* and the unit, or *empty word* is denoted by 1. A *context-free grammar* is a quadruple $G = (\Sigma, S, P, N)$ where N is the finite alphabet of *nonterminals* which is disjoint from the alphabet Σ of terminal letters, $S \in N$ is the *axiom* and $P \subseteq N \times (N \cup \Sigma)^*$ is the finite set of *productions* or *rules*. A rule is usually written in the form $X \to \alpha$. The smallest relation over $(N \cup \Sigma)^*$ containing $\to$ which is invariant under right and left concatenation is denoted by $\Rightarrow$ and its reflexive and transitive closure by $\overset{*}{\Rightarrow}$. The *language* generated by the grammar G is the set L_G of all words $w \in \Sigma^*$ such that $S \overset{*}{\Rightarrow} w$ holds.

A grammar is in *Chomsky's normal form* if its rules are of the form $X \to YZ$ or of the form $X \to a$ where X, Y, Z are nonterminals and a is a terminal. Such a grammar generates a context-free language which does not contain the empty word. Since a language $L \subseteq \Sigma^*$ is context-free if and only if $L \setminus \{1\}$ so is, there is no loss of generality to restrict the study to context-free languages not containing the empty word (also called ϵ-free in the literature). In other words, the expression "Given a context-free language" means in our context that the language is given by some grammar in Chomsky's normal form. It is convenient to enumerate the nonterminals $X_0, \dots, X_n$ and to assume that X_0 is the axiom. Furthermore, as usual, we assume that all nonterminals appear in some derivation of a word of the language, i.e., for all $X_i \in N$ there exist a word $w \in L_G$ and a derivation of the form $X_0 \overset{*}{\Rightarrow} \alpha X_i \beta \overset{*}{\Rightarrow} w$.

A *straight-line program* (SLP) is a restricted context-free grammar $G = (\Sigma, S, P, N)$ such that:

- for every $X \in N$, there exists exactly one production of the form $(X, \alpha) \in P$ for $\alpha \in (N \cup \Sigma)^*$, and
- there exists a linear order $\prec$ on the set of nonterminals N such that $X \prec Y$ whenever there exists a production of the form $(X, \alpha) \in P$ for $\alpha \in (N \cup \Sigma)^* Y (N \cup \Sigma)^*$.

Every SLP S obviously generates a single word, denoted by eval(S). In our work, we will always use SLPs in Chomsky's normal form. The size $g(S)$ of a SLP S is defined as the number of nonterminal symbols appearing in it. Since $g(S)$ could be logarithmic with respect to the length of eval(S), SLPs are considered as a compression encoding comparable to Lempel-Ziv encoding ([14]). For several well known string problems solvable in polynomial time, the compressed version has been studied, that is, the complexity of the same problem was analyzed when the input is given in a compressed form ([5,9,12,13,14]).

Let Σ be a finite alphabet. A *semi-Thue system* (or *string rewrite system*) is a pair of *generators* and *relators* $\langle \Sigma; R \rangle$ where R is a finite subset of $\Sigma^* \times \Sigma^*$. The pair $\langle \Sigma; R \rangle$ is a *monoid presentation*; it defines the monoid $M(R)$ which is the quotient of Σ^* by the congruence $\sim_R$ generated by the relators R. The *canonical* morphism ϕ_R maps Σ^* onto $M(R)$ by assigning to every word its congruence class. We use the same notation when we apply it to subsets and, more generally, to n-tuples of subsets. A pair $(u, v) \in R$ is also denoted by $u = v$.

The methodology introduced by Knuth in the seventies requires, whenever possible, to define a ordering of the free monoid which is invariant under right and left concatenation and to orient each relator (u, v) as $u \rightarrow v$ if u is greater than v. Once oriented, such a pair can be considered as a *rewrite rule*, also called *reduction rule* with the purpose that each occurrence of u in a word can be replaced by an occurrence of v. A word containing no occurrence of a left handside is called *reduced*, otherwise it is *reducible*. We still denote by $\rightarrow_R$ the closure of the set of reduction rules relative to right and left concatenation and by $\rightarrow_R^*$ its transitive closure. If the ordering thus defined is *complete*, i.e., it has the property of finite termination and of confluence, [4], each word x is equivalent to a unique reduced word denoted by $\mathrm{Red}_R(x)$, also simply written $\mathrm{Red}(x)$ when R is understood. Furthermore, this reduced word depends on the equivalence class only, and we say that it is its *normal form*. In this case, two words $x, y \in \Sigma^*$ are congruent, i.e., $x \sim_R y$, if and only if $\mathrm{Red}(x) = \mathrm{Red}(y)$.

Given a fixed complete string rewrite system, we are interested in the following decision problem

Inclusion Problem for $\langle \Sigma; R\rangle$

Instance: a grammar G in Chomsky normal form.

Question: does $L_G \subseteq \phi_R^{-1}(1)$ hold?

In this work, we show that, if the monoid $M(R)$ is cancellative, this problem is polynomially reducible to the compressed version of an equality problem:

Compressed-Equality Problem for $\langle \Sigma; R\rangle$

Instance: two SLPs S_1 and S_2 with $\text{eval}(S_1), \text{eval}(S_2) \in \Sigma^*$.

Question: does $\phi_R(\text{eval}(S_1)) = \phi_R(\text{eval}(S_2))$ hold?

2 Rewrite Systems Defining Cancellative Monoids

A monoid M is said to be *cancellative* if $xy = xz$ implies $y = z$ and $yx = zx$ implies $y = z$, for every $x, y, z \in M$. In this section, we prove that, if $\langle \Sigma; R\rangle$ defines a cancellative monoid $M(R)$, then Inclusion Problem for $\langle \Sigma; R\rangle$ is polynomially reducible to Compressed-Equality Problem for $\langle \Sigma; R\rangle$.

Now, let $m = |\Sigma|$ and consider a system of equations whose solutions are n-vectors of elements in the semiring $\langle 2^{\Sigma^*}, \cup, \cdot\rangle$ or in the semiring $\langle 2^M, \cup, \cdot\rangle$ accordingly

$$X_i = \bigcup_{0\le j,k\le n-1} \alpha^{(i)}_{j,k} X_j X_k \cup \bigcup_{1\le \ell\le m} \beta^{(i)}_\ell a_\ell \tag{1}$$

where a_ℓ are constant singletons and $\alpha^{(i)}_{j,k}$ and $\beta^{(i)}_\ell$ have values $\emptyset$ or 1.

When the X_i's are interpreted as subsets in Σ^*, the component X_0 of the least fixed point of (1) is the context-free language generated by the grammar whose productions are $X_i \to X_j X_\ell$ with $\alpha^{(i)}_{jk} \neq \emptyset$ and $X_i \to a_\ell$ with $\beta^{(i)}_\ell \neq \emptyset$. Therefore, we will not distinguish between systems of equations and systems of productions.

Lemma 1. *Consider a grammar G in Chomsky normal form as in (1) and a cancellative rewrite system $\langle \Sigma, R\rangle$. Assume that $\phi_R(L_G) = \{1\}$. Then, the images $\phi_R(X_i)$ of all components are singletons and satisfy Equation (1) interpreted in the quotient monoid $M(R)$. Conversely, if there exist elements $y_i \in \phi_R(X_i)$ such that the n-tuple $(y_0, y_1, \ldots, y_{n-1})$ with $y_0 = 1$ satisfies the system (1) interpreted in $M(R)$, then $\phi_R(L_G) = \{1\}$.*

Proof. Consider the graph whose vertices are the nonterminal symbols of G and there is an edge (X, Y) where $X \to YZ$ or $X \to ZY$. We perform a breadth-first visit starting from the axiom and verify by induction that, when visiting (X, Y), we have that $\phi_R(Y)$ is a singleton.

Indeed, this is true if X is the axiom, by definition. Now, consider the rule $X_i \to X_j X_k$ and assume without loss of generality that the visited edge is (X_i, X_j). Since $\phi_R(X_i) \supseteq \phi_R(X_j)\phi_R(X_k)$ holds and since $M(R)$ is cancellative, the induction hypothesis that $\phi_R(X_i)$ is a singleton implies that so is $\phi_R(X_j)$.

Let us prove the converse. First observe that the image in the morphism ϕ_R of the least (in the subset ordering) solution of the system (1) in the power set of Σ^* is the least solution of the system when interpreted in the power set of $M(R)$. Indeed, let $\tau : \left(2^{\Sigma^*}\right)^n \to \left(2^{\Sigma^*}\right)^n$ and $\sigma : \left(2^M\right)^n \to \left(2^M\right)^n$ be the transformations defined by the system (1) when interpreted in $\left(2^{\Sigma^*}\right)^n$ and $\left(2^M\right)^n$ respectively and observe that these transformations are monotone and continuous. The least solution of the system in $\left(2^{\Sigma^*}\right)^n$ is the subset $\bigcup_{k\to\infty} \tau^k(\emptyset)$. Now we have

$$\phi_R\left(\bigcup_{k\to\infty} \tau^k(\emptyset)\right) = \bigcup_{k\to\infty} \phi_R(\tau^k(\emptyset))$$

Because of equality $\phi_R\tau = \sigma\phi_R$ between the two composed mappings and of the equality $\phi_R(\emptyset) = \emptyset$ we finally obtain

$$\phi_R\left(\bigcup_{k\to\infty} \tau^k(\emptyset)\right) = \bigcup_{k\to\infty} \sigma^k(\phi_R(\emptyset)) = \bigcup_{k\to\infty} \sigma^k(\emptyset)$$

as claimed.

The previous Lemma has the following interesting consequence.

Theorem 1. *In the case of a cancellative complete rewrite system $\langle\Sigma, R\rangle$,* Inclusion Problem for *$\langle\Sigma, R\rangle$ is polynomially reducible to* Compressed-Equality problem for *$\langle\Sigma, R\rangle$.*

Proof. (Outline) In view of the previous Lemma, the condition $\phi_R(L_G) = \{1\}$ can be tested in two steps.

First step: To every nonterminal X_i of G, we assign a SLP Φ_i representing a word w_i, chosen as explained below.

Second step: We check whether or not the n-tuple $(w_0, \dots, w_{n-1})$ satisfies the following conditions:

- $w_0 \sim_R 1$;
- for all productions $X_i \to X_jX_k$ (resp. $X_i \to a_\ell$), $w_i \sim_R w_jw_k$ (resp. $w_i \sim_R a_\ell$)). These conditions are not verified on the words, but through SLPs representing the words.

We now explain how we choose the SLPs Φ_i representing the words w_i. Denote by $h(X_i)$ the minimal height of a derivation tree with root labelled by X_i. If $h(X_i) = 1$, then take as Φ_i an arbitrary production rule of the form $X_i \to w_i$ in the grammar G. More generally, if $h(X_i) > 1$, there exist two non-terminals X_j, X_k such that $h(X_i) > h(X_j), h(X_k)$ and $X_i \to X_jX_k$ is a production of G. Then recursively choose w_jw_k as the word w_i represented by the SLP obtained by considering the production $X_i \to X_jX_k$ along with the productions of the two SLPs Φ_j and Φ_k, representing w_j and w_k.

From a complexity viewpoint, considering the grammar size as the size of the input, the first step can be executed in polynomial time, while the second requires to call the SLPs equality test in $M(R)$ a polynomial number of times. This proves the result.

Example 1. Fix the monoid M presented by $\langle\{a,\bar{a},b,\bar{b}\};\{(a\bar{a},1),(b\bar{b},1)\}\rangle$ and let $\phi:\{a,\bar{a},b,\bar{b}\}^* \rightarrow M$ be the canonical morphism.

Consider the language $L \subset \{a,\bar{a},b,\bar{b}\}^*$ defined by the grammar in Chomsky's Normal Form with axiom X_0 whose productions are:

$$\begin{array}{lll} X_0 \rightarrow X_3X_1/X_4X_2/1, & X_3 \rightarrow a, & X_5 \rightarrow \bar{a}, \\ X_1 \rightarrow X_0X_5, & X_4 \rightarrow b, & X_6 \rightarrow \bar{b}, \\ X_2 \rightarrow X_0X_6. & & \end{array}$$

We are interested in establish if L is contained in the preimage of 1 in M. The set of SLPs described in the proof of Theorem 1 is presented in the following table. Last row contains the evaluations of the SLPs:

Φ_6	Φ_5	Φ_4	Φ_3	Φ_2	Φ_1	Φ_0
$X_6 \rightarrow b$	$X_5 \rightarrow \bar{a}$	$X_4 \rightarrow b$	$X_3 \rightarrow a$	$X_0 \rightarrow 1$ $X_6 \rightarrow \bar{b}$ $X_2 \rightarrow X_0X_6$	$X_0 \rightarrow 1$ $X_5 \rightarrow \bar{a}$ $X_1 \rightarrow X_0X_5$	$X_0 \rightarrow 1$
$\bar{b}$	$\bar{a}$	b	a	$\bar{b}$	$\bar{a}$	1

Once the table is available, the following equalities must be verified:

$$\begin{array}{ll} \mathrm{eval}(\Phi_0) = \mathrm{eval}(\Phi_3\Phi_1), & \mathrm{eval}(\Phi_3) = a, \\ \mathrm{eval}(\Phi_0) = \mathrm{eval}(\Phi_4\Phi_2), & \mathrm{eval}(\Phi_4) = b, \\ \mathrm{eval}(\Phi_0) = 1, & \mathrm{eval}(\Phi_5) = \bar{a}, \\ \mathrm{eval}(\Phi_1) = \mathrm{eval}(\Phi_0\Phi_5), & \mathrm{eval}(\Phi_6) = \bar{b}, \\ \mathrm{eval}(\Phi_2) = \mathrm{eval}(\Phi_0\Phi_6), & \end{array}$$

where $\Phi_j\Phi_s$ is intended as the SLP representing $\mathrm{eval}(\Phi_j)\mathrm{eval}(\Phi_s)$. Since each of these equalities holds, $L \subseteq \phi^{-1}(1)$.

Theorem 1 can be applied to obtain an upper bound to the INCLUSION PROBLEM complexity for those cancellative monoids whose COMPRESSED-EQUALITY PROBLEM complexity is known. This is the case of 2-homogeneous rewrite systems, introduced in [2], that will be discussed in the next section.

3 Unitary Confluent Systems

A presentation $\langle\Sigma;R\rangle$ is *unitary* if R is composed of elements of the form $(u,1)$. We study the case where R can be oriented as a confluent and therefore complete system.

First of all, observe that for all unitary rewrite systems, the set of words which can be reduced to the empty word is generated by a context-free grammar with a unique nonterminal S whose productions are of the form $S \rightarrow 1$ and $S \rightarrow Sa_1Sa_2\cdots Sa_pS$ where $(a_1a_2\cdots a_p,1) \in R$. Hence, INCLUSION PROBLEM for unitary rewrite systems is a special case of the general inclusion problem for context-free grammars (see for example [7,6]).

Before studying more specifically the so-called 2-homogeneous case, we prove a result for the general case. Consider the following problem:

Regular Set Inclusion Problem

Input: a regular language $L \subseteq \Sigma^*$, a unitary rewrite system $\langle \Sigma, R \rangle$ and a regular subset $X \subseteq M(R)$.

Question: does $\phi_R(L) \subseteq X$ hold?

Theorem 2. Regular Set Inclusion Problem *is solvable in polynomial time.*

Proof. We briefly sketch how one can check the inclusion $\phi_R(L) \subseteq X$ efficiently. Since each equivalence class has a unique normal form, the previous inclusion is equivalent to saying that the set of normal forms of all elements in L is included in the set of normal forms which are representatives of an element of X. By definition of a regular set in the quotient $M(R)$, we may suppose that X is given by a finite automaton which recognizes for each element $x \in X$ and only for elements in X, a word representing this element. It thus suffices to show that the set of normal forms is an effective regular language. Indeed, if L is a regular language recognized by a finite automaton, augment it by adding, as long as possible, a transition from state q to state p labeled by the empty word whenever there exists a path from q to p labeled by a left handside of a rule. This procedure can be easily executed in polynomial time with respect to the size of the automaton and the size of the presentation of the monoid. The set of its normal forms is the intersection of the language recognized by this augmented automaton with the regular set of words containing no occurrence of a left handside of a rewrite rule. The inclusion problem consists thus of checking the inclusion $\mathrm{Red}_R(L) = \mathrm{Red}_R(\phi^{-1}(X))$.

3.1 2-Homogeneous Rewrite Systems

A unitary rewrite system is said to be *2-homogeneous* if every left-hand member of R is in Σ^2. In [2] it is shown that every 2-homogeneous rewrite system is equivalent to a 2-homogeneous confluent rewrite system $\langle \Theta, R \rangle$. In addition, Lohrey ([10]) proved that, for such a presentation, it is always possible to define a partition $\Theta = \Sigma \uplus \Delta \uplus \Gamma$ and an involution $^- : \Sigma \to \Sigma$ with

$$\{(a\bar{a}, 1) \mid a \in \Sigma\} \;\subseteq\; R \;\subseteq\; \{(a\bar{a}, 1) \mid a \in \Sigma\} \cup \{(ab, 1) \mid a \in \Delta, b \in \Gamma\}.$$

A rewrite system is called *N-free* if it is 2-homogeneous and $ac, ad, bc \in \mathrm{dom}(R)$ implies that $bd \in \mathrm{dom}(R)$, for every $a, b \in \Delta$ and $c, d \in \Gamma$. This means that the graph determined by nodes in $\Delta \cup \Gamma$ and edges $\{(a, b) \in \Delta \times \Gamma \mid ab \in \mathrm{dom}(R)\}$ is the disjoint union of complete bipartite graphs. In this case, we can rewrite Θ as

$$\Theta = \Sigma \cup \bigcup_{1 \le i \le k} \Delta_i \cup \bigcup_{1 \le i \le k} \Gamma_i \tag{2}$$

and R as

$$R \; = \; \{(a\bar{a},1) \mid a \in \Sigma\} \; \cup \; \{(ab,1) \mid a \in \Delta_i, b \in \Gamma_i, \; 1 \leq i \leq k\}. \qquad (3)$$

In this construction, for each i, $\Delta_i \cup \Gamma_i$ are the nodes of a complete bipartite graph. Moreover, if the sets Δ_i and Γ_i are all singletons, then the rewrite system is cancellative.

Particularly important cases are:

- The free group reduction

$$\left\langle a_1, \ldots, a_m, a_1^{-1}, \ldots, a_m^{-1}; \bigcup_{1 \leq i \leq m} (a_i a_i^{-1}, 1) \cup (a_i^{-1} a_i, 1) \right\rangle.$$

- The Dyck reduction ([1])

$$\left\langle a_1, \ldots, a_m, \bar{a}_1, \ldots, \bar{a}_m; \bigcup_{1 \leq i \leq m} (a_i \bar{a}_i, 1) \right\rangle;$$

  in this case, $\phi_R^{-1}(1)$ is the Dyck set on $\{a_1, \ldots, a_m, \bar{a}_1, \ldots, \bar{a}_m\}$.

Lemma 2. *Let $\langle \Theta, R \rangle$ be a N-free rewrite system having $|\Delta_i| = |\Gamma_i| = 1$ for $1 \leq i \leq n$. Then,* Inclusion Problem for *$\langle \Theta, R \rangle$ is solvable in polynomial time.*

Proof. It is known that, in the case of a N-free rewrite system $\langle \Sigma, R \rangle$, Compressed Equality Problem for $\langle \Sigma, R \rangle$ is solvable in polynomial time ([11]). Hence, by Theorem 1, the result is straightforwardly proved.

We now show how to extend this result to the general case of N-free rewrite systems. Consider a presentation $\langle \Theta; R \rangle$ defined by the conditions (2) and (3). We introduce two symbols δ_i and γ_i for each Δ_i and Γ_i, respectively, and consider the alphabet

$$\Theta' = \Sigma \cup \bigcup_{1 \leq i \leq k} \{\delta_i, \gamma_i\} \qquad (4)$$

and the relation R' over Θ' defined by the same relators as those of Eq. (3) for Σ and the relators

$$\delta_i \gamma_i = 1, \quad \text{for } 1 \leq i \leq k.$$

We denote by $\sim_{R'}$ the congruence induced by R' on Θ'^* and by π the morphism from $\Theta^* / \sim_R$ to $\Theta'^* / \sim_{R'}$ leaving every element of Σ invariant and mapping all elements of Δ_i to δ_i and all elements of Γ_i to γ_i.

Theorem 3. Inclusion Problem for *$\langle \Sigma, R \rangle$ is solvable in polynomial time whenever $\langle \Sigma, R \rangle$ is N-free.*

Proof. Let the input of the problem be the context-free grammar G in Chomsky's normal form. Then, it suffices to prove that $\phi(L_G) = 1$ if and only if $\phi(\pi(L_G)) = 1$.

We prove by induction on the length of the words that $w \sim_R 1$ implies $\pi(w) \sim_{R'} 1$. As the system of rewriting rules is confluent, the relation $w \sim_R 1$ implies that we may write $w = w_1 u w_2$ where $u = a\overline{a}$ or $u = \overline{a}a$ with $a \in \Sigma$ or $u = \delta\gamma$ with $\delta \in \Delta_i$ and $\gamma \in \Gamma_i$, for some i. The former case is trivial, so we assume the latter holds. Take the image by π:

$$\begin{aligned}\pi(w) = \pi(w_1)\pi(\delta)\pi(\gamma)\pi(w_2) = \pi(w_1)\delta_i\gamma_i\pi(w_2) \sim_{R'}\\ \pi(w_1)\pi(w_2) = \pi(w_1 w_2) \sim_{R'} 1.\end{aligned}$$

Conversely if $\pi(w) \sim_{R'} 1$, then $\pi(w) = \pi(w_1)v\pi(w_2)$ with $v = a\overline{a}$ or $v = \overline{a}a$ with $a \in \Sigma$ or $v = \delta_i\gamma_i$ for some i. In this case, $v = \pi(u)$, where $u = \delta\gamma$ with $\delta \in \Delta_i$ and $\gamma \in \Gamma_i$ for some i. Hence, $w \sim_R w_1 w_2$, which completes the proof.

Now we turn to the simplest non N-free presentation, to wit the presentation $\langle \Sigma; R\rangle$ where $\Sigma = \{a, b, c, d\}$ and R is defined by the relators

$$ac \to 1, bc \to 1, \text{ and } ad \to 1, \tag{5}$$

while $bd \notin \text{dom}(R)$. In order to give a lower bound to the complexity of Inclusion Problem for $\langle \Sigma; R\rangle$ in the case $M(R)$ is non N-free, we consider the following compressed string problem

Compressed 1-Equality Problem for $\langle \Sigma; R\rangle$

Input: an SLP S with $\text{eval}(S) \in \Sigma^*$.

Question: does $\phi_R(\text{eval}(S)) = 1$ hold?

Since a SLP is a special case of context-free grammar, such a problem is obviously a special case of Inclusion Problem for $\langle \Sigma; R\rangle$. This means that the complexity of Inclusion Problem for $\langle \Sigma; R\rangle$ lies between the complexity of Compressed 1-Equality Problem and that of Compressed Equality Problem for the same rewrite system.

In [11, Theorems 5.2, 5.4], Compressed 1-Equality Problem for non N-free is proved to be coNP-complete. Hence, the following straightforwardly follows

Remark 1. If $\langle \Sigma; R\rangle$ is non N-free, then Inclusion Problem for $\langle \Sigma; R\rangle$ is at least coNP-Hard.

4 Conclusions

We studied some conditions for which the inclusion problem for context-free languages is decidable in polynomial time. We showed that verifying whether all the words of a context-free language are mapped to the empty word in a cancellative monoid can be reduced to testing the equality of two SLPs in the same monoid. This result solves the inclusion problem for the 2-homogeneous rewrite systems, for which there exists a polynomial solution if the system is N-free, while, in the other case, the problem is at least coNP hard.

In the more general case of complete unitary rewrite systems, we proved that the inclusion of a regular language in a regular set over the generated monoid is solvable in polynomial time.

An open problem on 2-homogeneous rewrite systems is to study the precise complexity of Inclusion Problem in the non N-free case; moreover, it would be interesting to identify complete unitary rewrite systems for which Inclusion Problem is solvable in polynomial time.

References

1. Berstel, J., Boasson, L.: Formal properties of XML grammars and languages. Acta Inform. 38(9), 649–671 (2002)
2. Book, R.V.: Homogeneous Thue systems and the Church-Rosser property. Discrete Math. 48(2-3), 137–145 (1984)
3. Cochet, Y.: Church-Rosser congruences on free semigroups. In: Algebraic theory of semigroups (Proc. Sixth Algebraic Conf., Szeged, 1976). Colloq. Math. Soc. János Bolyai, vol. 20, pp. 51–60. North-Holland, Amsterdam (1979)
4. Dershowitz, N., Jouannaud, J.-P.: Rewrite systems. In: Handbook of theoretical computer science, vol. B, pp. 243–320. Elsevier, Amsterdam (1990)
5. Farach, M., Thorup, M.: String matching in Lempel-Ziv compressed strings. Algorithmica 20(4), 388–404 (1998)
6. Greibach, S.A., Friedman, E.P.: Superdeterministic PDAs: a subcase with a decidable inclusion problem. J. Assoc. Comput. Mach. 27(4), 675–700 (1980)
7. Hopcroft, J.E.: On the equivalence and containment problems for context-free languages. Math. Systems Theory 3, 119–124 (1969)
8. Knuth, D.E.: A characterization of parenthesis languages. Inform. Control 11(3), 269–289 (1967)
9. Lifshits, Y.: Processing compressed texts: A tractability border. In: Ma, B., Zhang, K. (eds.) CPM 2007. LNCS, vol. 4580, pp. 228–240. Springer, Heidelberg (2007)
10. Lohrey, M.: Word problems for 2-homogeneous monoids and symmetric logspace. In: Sgall, J., Pultr, A., Kolman, P. (eds.) MFCS 2001. LNCS, vol. 2136, pp. 500–511. Springer, Heidelberg (2001)
11. Lohrey, M.: Word problems and membership problems on compressed words. SIAM J. Comput. 35(5), 1210–1240 (2006)
12. Miyazaki, M., Shinohara, A., Takeda, M.: An improved pattern matching algorithm for strings in terms of straight-line programs. J. Discrete Algorithms (Oxf.) 1(1), 187–204 (2000)
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. Rytter, W.: Algorithms on compressed strings and arrays. In: Bartosek, M., Tel, G., Pavelka, J. (eds.) SOFSEM 1999. LNCS, vol. 1725, pp. 48–65. Springer, Heidelberg (1999)

On the Complexity of Deciding Avoidability of Sets of Partial Words*

Brandon Blakeley[1], Francine Blanchet-Sadri[2], Josh Gunter[1],
and Narad Rampersad[3]

[1] Department of Computer Sciences, The University of Texas at Austin,
1 University Station C0500 Taylor Hall 2.124, Austin, TX 78712-0233, USA
[2] Department of Computer Science, University of North Carolina,
P.O. Box 26170, Greensboro, NC 27402–6170, USA
blanchet@uncg.edu
[3] Department of Mathematics and Statistics, University of Winnipeg,
515 Portage Avenue, Winnipeg, MB R3B 2E9, Canada

Abstract. Blanchet-Sadri et al. have shown that Avoidability, or the problem of deciding the avoidability of a finite set of partial words over an alphabet of size $k \geq 2$, is $\mathcal{NP}$-hard [Theoret. Comput. Sci. **410** (2009) 968–972]. Building on their work, we analyze in this paper the complexity of natural variations on the problem. While some of them are $\mathcal{NP}$-hard, others are shown to be efficiently decidable. Using some combinatorial properties of de Bruijn graphs, we establish a correspondence between lengths of cycles in such graphs and periods of avoiding words, resulting in a tight bound for periods of avoiding words. We also prove that Avoidability can be solved in polynomial space, and reduces in polynomial time to the problem of deciding the avoidability of a finite set of partial words of equal length over the binary alphabet. We give a polynomial bound on the period of an infinite avoiding word, in the case of sets of full words, in terms of two parameters: the length and the number of words in the set. We give a polynomial space algorithm to decide if a finite set of partial words is avoided by a non-ultimately periodic infinite word. The same algorithm also decides if the number of finite words of length n avoiding a given finite set of partial words grows polynomially or exponentially with n.

1 Introduction

A set of (full) words X over a finite alphabet A is called *unavoidable* if every two-sided infinite word over A has a factor in X (when a word w has no factor in X, we say that w avoids X); otherwise X is called *avoidable*. Consequently, a subset X of A^* is unavoidable if and only if $A^* \setminus A^*XA^*$ is finite, and any

* This material is based upon work supported by the National Science Foundation under Grant No. DMS–0754154. The Department of Defense is also gratefully acknowledged. We thank the referees of a preliminary version of this paper for their very valuable comments and suggestions.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 113–124, 2009.

unavoidable set contains a finite one. An alternate characterization of a finite unavoidable set is that every periodic two-sided infinite word has a factor in X [1]. Among other topics, the cardinality of such sets has been investigated [1]. If we take, for example, one element from each of the conjugacy classes $\{aa\}$, $\{bb\}$ and $\{ab, ba\}$ of the set of length two words over the alphabet $\{a, b\}$, then we build an unavoidable set. Note that there is at least one element from each class in an unavoidable set of words of length two since we can construct an infinite word whose factors of length two all belong to the same class. This observation can be generalized, so that any unavoidable set of words of length m over a k-letter alphabet contains at least as many words as there are conjugacy classes. In [2], it was proved that this bound is sharp (see [3] for a simpler proof).

A set of partial words X over a finite alphabet A is called *unavoidable* if every two-sided infinite full word over A has a factor *compatible* with a member of X. Partial words are sequences that may contain some "holes," denoted by "$\diamond$'s," that match any letter of the alphabet (we also say that $\diamond$ is compatible with any letter of the alphabet). Unavoidable sets of partial words were introduced in [4] where the number theoretic problem of classifying such sets of size $l \geq 2$ over a k-letter alphabet with $k \leq l$ was initiated.

Efficient algorithms to determine if a finite set of full words is unavoidable are well known [1]. For example, we can check whether there is a loop in the finite automaton of Aho and Corasick [5] recognizing $A^* \setminus A^*XA^*$. These same algorithms can be used to decide if a finite set of partial words X is unavoidable by determining the unavoidability of $\hat{X}$, the set of all full words compatible with an element of X. Indeed, by the definition of $\hat{X}$, a two-sided infinite word w has a factor in $\hat{X}$ if and only if that factor is compatible with a member of X. Thus the infinite words which avoid $X \subset A_\diamond^*$ are exactly those which avoid $\hat{X} \subset A^*$, and $X \subset A_\diamond^*$ is unavoidable if and only if $\hat{X} \subset A^*$ is unavoidable. However this incurs a dramatic loss in efficiency, as each partial word u in X can contribute as many as $|A|^{|H(u)|}$ elements to $\hat{X}$ ($H(u)$ denotes the set of holes of u).

In [6], it was proved that testing the unavoidability of a finite set of partial words is much harder to handle than the similar problem for full words. Indeed, it turns out that the problem of deciding whether a finite set of partial words over a k-letter alphabet where $k \geq 2$ is unavoidable is *$\mathcal{NP}$-hard* (the complexity class of those decision problems that are at least as hard as any problem that can be solved by a non-deterministic Turing machine in polynomial time), which is in contrast with the well known feasability results for unavoidability of a set of full words [7, Chapter 7.4] (note that the case $k = 1$ is trivial).

The enumeration problem for words of length n avoiding a finite set of full words has been studied by several authors. For example, Kobayashi [8] presented a matrix-theoretic approach to this problem; Goulden and Jackson [9] describe another method. A set of words L is of *polynomial growth* if there exists a polynomial $p(n)$ such that the number of words in L of length n is at most $p(n)$ for all $n \geq 0$. The set L is of *exponential growth* if there exists a real number $r > 1$ such that for infinitely many $n \geq 0$, the number of words in L of length n is at least r^n. Over any fixed alphabet, the set of finite words avoiding any given finite set of partial words can be of either polynomial growth or exponential growth; no intermediate growth is possible. This is a consequence of previous work on the avoidability of sets of full words (see [8] for example).

The contents of our paper is as follows: In Section 2, we review basic concepts on partial words and discuss previous work on avoidability of sets of such words. In Section 3, we analyze the complexity of natural variations on the problem of deciding avoidability of sets of partial words. While some of them are shown to be $\mathcal{NP}$-hard, others are shown to be efficiently decidable. We establish a correspondence between lengths of cycles in de Bruijn graphs and periods of avoiding words, resulting in a bound for periods of avoiding words. We also show that the problem of deciding the avoidability of a finite set of partial words over an alphabet of size $k \geq 2$ can be solved in polynomial space, and reduces in polyomial time to the problem of deciding the avoidability of a finite set of partial words of equal length over the binary alphabet. In Section 4, we give a polynomial bound on the period of an avoiding word, in the case of sets of full words, in terms of two parameters: the length and the number of words in the set. In Section 5, we give a polynomial space algorithm to decide if a finite set of partial words is avoided by a non-ultimately periodic infinite word over a fixed alphabet. Our algorithm also decides if the number of words of length n avoiding a given finite set of partial words grows polynomially or exponentially with n. We also apply the probabilistic method to show that if a set X of partial words is not too large, the number of words of length n avoiding X grows exponentially. Finally in Section 6, we conclude with some remarks.

2 Preliminaries

Throughout this paper A is a fixed non-empty finite set called an *alphabet* whose elements we call *letters*. A *word* of length n over A is a finite sequence of elements of A. We denote by A^* (respectively, A^n) the set of finite words (respectively, the set of words of length n) over A. For $u \in A^*$, we write $|u|$ for the length of u. Under the concatenation operation of words, A^* forms a free monoid whose identity is the empty word which we denote by ε.

A *two-sided infinite word* w is a function $w : \mathbb{Z} \to A$. A finite word u is a factor of w if there exists some $i \in \mathbb{Z}$ such that $u = w(i) \cdots w(i + |u| - 1)$. For a positive integer p, w has *period* p, or w is *p-periodic*, if $w(i) = w(i + p)$ for all $i \in \mathbb{Z}$. If w has period p for some p, then we call w *periodic*. If v is a non-empty finite word, then we denote by $v^{\mathbb{Z}}$ the unique two-sided infinite word w with period $|v|$ such that $v = w(0) \cdots w(|v| - 1)$. A *one-sided infinite word* w is a function $w : \mathbb{N} \to A$. It is *ultimately periodic* if it can be written as $w = uvvvv \cdots$ for some finite words u and v, where v is non-empty.

A *partial word* (or *pword*) u of length n over an alphabet A can be defined as a function $u : [0..n - 1] \to A_\diamond$, where $A_\diamond = A \cup \{\diamond\}$, and will be written as $u(0)u(1) \cdots u(n - 1)$. For $0 \leq i < n$, if $u(i) \in A$, then i belongs to the *domain* of u, denoted $D(u)$; otherwise, i belongs to the *set of holes* of u, denoted $H(u)$. Whenever $H(u)$ is empty, we say that u is a *full* word. We refer to an occurrence of the symbol $\diamond$ as a *hole*. We denote by $A_\diamond^*$ the set of all partial words over A with an arbitrary number of holes. We denote the set of all factors of u by $F(u)$.

Two partial words u and v of equal length are *compatible*, denoted $u \uparrow v$, if $u(i) = v(i)$ whenever $i \in D(u) \cap D(v)$. If X is a set of partial words, we denote by $\hat{X}$ the set of all full words compatible with an element of X. The partial word u is *contained* in v, denoted $u \subset v$, if $|u| = |v|$ and $u(i) = v(i)$ for all $i \in D(u)$. Two partial words u and v are *conjugate* if there exist partial words x, y such that $u \subset xy$ and $v \subset yx$. It is well-known that conjugacy on full words is an equivalence relation, but it is not such a relation on partial words [10]. If a partial word u can be written as $u = u_1 \diamond u_2 \diamond \cdots u_{n-1} \diamond u_n$, then the set $\{u_1 a_1 u_2 a_2 \cdots u_{n-1} a_{n-1} u_n \mid a_1, a_2, \ldots, a_{n-1} \in A\}$ is called a *partial expansion* on u (note that $u_1, u_2, \ldots, u_n$ are partial words that may contain holes, and also note that $u \subset v$ for every member v of a partial expansion on u).

A two-sided infinite word w over A *avoids* $X \subset A_\diamond^*$ if no factor of w is an element of $\hat{X}$. We say that X is *unavoidable* if no two-sided infinite word over A avoids X. Previous work shows that Avoidability is $\mathcal{NP}$-hard. In [6], it is proved that determining if a finite set of partial words over an alphabet of size $k \geq 2$ is avoidable or not is much harder to handle than the similar problem for full words. This is done by using a reduction from the 3SAT problem, known to be $\mathcal{NP}$-complete. We refer the reader to Reference [4] that gives an algorithm, that will be used in some of the proofs, for deciding Avoidability based on reductions from a set X to a set Y that maintain avoidability: factoring, prefix-suffix, hole truncation, and expansion. A set $X \subset A_\diamond^*$ is unavoidable if and only if X can be reduced to $\{\varepsilon\}$ by these reductions.

3 Complexity of Avoidability Problems

In this section, we discuss natural variations on the problem of deciding avoidability of sets of partial words. While some of them are $\mathcal{NP}$-hard, others are shown to be efficiently decidable. We establish a correspondence between lengths of cycles in de Bruijn graphs and periods of avoiding words. We also show that the problem of deciding the avoidability of a finite set of pwords over a k-letter alphabet can be solved in polynomial space, and reduces in polynomial time to the problem of deciding the avoidability of such a set over the binary alphabet.

Testing if a word avoids a finite set can be done using Lemma 1 which we will implicitly use when proving membership in certain complexity classes in the following results concerning restricted Avoidability.

Lemma 1. *Given a finite word v, the problem of deciding if the infinite periodic word $v^{\mathbb{Z}}$ avoids a finite set of partial words can be solved in polynomial time.*

Theorem 1. *The problem of deciding the avoidability of a finite set of partial words, such that each element has at most two defined positions, is $\mathcal{NP}$-hard.*

Proof. Our proof proceeds by reduction from the Directed Hamiltonian Circuit problem, one of Karp's original twenty-one $\mathcal{NP}$-complete problems [11]. In the

Directed Hamiltonian Circuit problem, we decide whether a given digraph has a Hamiltonian circuit. Given a graph $G = (V, E)$, we construct a set X of partial words, where each element has at most two defined positions, such that X is avoidable if and only if G has a Hamiltonian circuit. Our alphabet is $V = \{v_1, v_2, \ldots, v_n\}$ and the set X is composed of the following three parts: (1) $\{v_i v_j \mid (v_i, v_j) \notin E\}$, (2) $\{v_i \diamond^{n-1} v_j \mid v_i \neq v_j\}$, and (3) $\{v_i \diamond^j v_i \mid 0 \leq j < n-1\}$.

For the forward implication, suppose there exists a Hamiltonian circuit in G, say $(u_1, u_2, \ldots, u_n, u_1)$. We claim that $w = (u_1 u_2 \cdots u_n)^{\mathbb{Z}}$ avoids X. Indeed, w avoids Part (1) of X because each $(u_i, u_{i+1}) \in E$ for $0 < i < n$ and $(u_n, u_1) \in E$. Part (2) is avoided because w is n-periodic. Part (3) is avoided because consecutive occurrences of the same letter are separated by $n-1$ other letters. For the reverse implication, suppose there exists a two-sided infinite word w which avoids X. To avoid Part (3), consecutive occurrences of the same letter must be separated by at least $n-1$ other letters. To avoid Part (2), $w(i) = w(i+n)$ for all $i \in \mathbb{Z}$, so w must be n-periodic. From our previous observations, this period must be of the form $u_1 u_2 \cdots u_n$, where each u_i is distinct. Finally, to avoid Part (1), $(u_i, u_{i+1}) \in E$ where $0 < i < n$ and $(u_n, u_1) \in E$. Therefore, $(u_1, u_2, \ldots, u_n, u_1)$ is a Hamiltonian circuit in G. □

The following proposition shows membership in $\mathcal{NP}$ of the problem defined in Theorem 1 over a binary alphabet.

Proposition 1. *The problem of deciding the avoidability of a finite set of partial words over the binary alphabet, such that each element has at most two defined positions, can be solved in non-deterministic polynomial time.*

The next theorem shows that another natural variation (see, for example, [3]), constant length sets, on the problem of deciding avoidability is $\mathcal{NP}$-hard.

Remark 1. When we consider constant length sets of partial words, we implicitly require that neither the first or last position in any of the words be a hole.

Theorem 2. *The problem of deciding the avoidability of a finite set of partial words of equal length over an alphabet of size $k \geq 2$ is $\mathcal{NP}$-hard.*

Proof. We present a reduction from the $\mathcal{NP}$-hard unrestricted Avoidability problem. Given a finite set X of partial words over a k-size alphabet A, we construct a set X' of pwords of equal length as follows. Let l denote the maximum length of the words in X. Then X' is formed by the following two parts: $\{u \diamond^{l-|u|-1} a \mid u \in X, |u| < l, a \in A\}$ and $\{u \mid u \in X, |u| = l\}$. We show that X' is avoided by the same words as X. Consider for any $u \in X$ where $|u| < l$ the set $X'_u = \{u \diamond^{l-|u|-1} a \mid a \in A\}$ which has the same avoidability as $Y'_u = \{u \diamond^{l-|u|}\}$ because an Expansion operation on Y'_u results in X'_u. Furthermore, a Hole Truncation operation on Y'_u results in the set $\{u\}$. Therefore, X'_u is avoided by the same words as $\{u\}$. By our construction of X', clearly X' is avoided by the same words as X. Therefore, X is avoidable if and only if X' is avoidable. Finally, we note that the length of the description of X, that is $\|X\| = \sum_{x \in X} |x|$, satisfies $\|X'\| < \|X\| lk$, and so this reduction runs in polynomial time. □

A tractable variation is provided in the next theorem. As a direct corollary, combining the two restrictions presented in the previous two theorems results in a problem which can be efficiently decided.

Theorem 3. *The problem of deciding the avoidability of a finite set X of partial words, where for some positive integer p every element $x \in X$ is defined at position $0 \leq i < |x|$ if and only if p divides i, can be solved in polynomial time.*

Corollary 1. *The problem of deciding the avoidability of a given finite set of partial words of equal length n, where each element has at most two defined positions (by Remark 1, each element has the form $a\diamond^{n-2}b$), can be solved in polynomial time.*

Another natural variation of Avoidability is presented in the next theorem.

Theorem 4. *The problem of deciding whether a finite set of partial words is avoided by a word of length l is strongly $\mathcal{NP}$-complete.*

Using de Bruijn graphs, well known to be Hamiltonian and Eulerian, Theorem 5 will give a bound on periods of avoiding words by establishing a correspondence between them and lengths of cycles in such graphs. The *de Bruijn graph* of order m over a k-size alphabet A, denoted $G(m,k)$, is the digraph (V,E) defined as follows: For $m \geq 1$, $V = A^m$ and $E = \{(z,z') \mid z' = \sigma(z,a)\}$, where we denote by $\sigma(z,a)$ the word z' of length m such that for some $b \in A$, $za = bz'$ (here $b = z(0)$, $z'(0) = z(1)$, $z'(1) = z(2)$, …, $z'(m-2) = z(m-1)$, $z'(m-1) = a$).

Theorem 5. *If a finite set of partial words of length m over an alphabet A is avoidable, then it is avoided by a word of period at most $|A|^m$.*

Proof. Let X be a finite avoidable set of pwords of length m. Consider the subgraph $G = (V,E)$ of $G(m,k)$ induced by the set $\{u \mid u \not\uparrow x \text{ for all } x \in X\}$. Essentially, incidence in G corresponds to transitions in the automaton of Aho and Corasick [5]. We claim that there exists a cycle in G of length p if and only if there exists an infinite word with period p which avoids X. Consider any cycle C in G of length p. Construct the word v_C formed by concatenating the first letters of each vertex along the cycle. By our construction of G, no subword of $(v_C)^{\mathbb{Z}}$ of length p is compatible with any word in X. Therefore, the infinite word $(v_C)^{\mathbb{Z}}$ of period p avoids X. Now suppose there exists a cycle C in G of length greater than $|V|$. Then, by the pigeonhole principle, C is not simple, and so we can find a simple cycle C' of length at most $|V|$. Therefore, since $|V| \leq |A|^m$, there exists an infinite word with period at most $|A|^m$ that avoids X. □

Theorem 6. *The problem of deciding the avoidability of a finite set of partial words of equal length can be solved in polynomial space.*

Proof. We apply the bound found in Theorem 5 to obtain a polynomial space algorithm which decides the avoidability of a finite set X of partial words of length m over alphabet A. Algorithm 1 searches for a cycle in the graph defined in Theorem 5 without constructing the graph. The correctness of this algorithm

Algorithm 1. Deciding Avoidability in Polynomial Space

```
Non-deterministically select a word $w$ of length $m$
Set $z = w$, $i = 0$
while $i < |A|^m$ do
    Increment $i$
    Non-deterministically select a letter $a \in A$
    Set $z = \sigma(z, a)$
    If $\exists x \in X$ such that $z \uparrow x$, reject
    If $z = w$, accept
Reject
```

can be proved with the loop invariant that, at iteration i, there is a path of length i from w to z. So if there is a cycle in G, then there is a cycle with at most $|A|^m$ vertices and our algorithm will accept. If there is no cycle, then there is no path from w to w of length at least 1 and our algorithm will reject. Because our algorithm stores only two words of length m and a counter of length $m \log |A|$, it uses $\mathcal{O}(m)$ non-deterministic space, and so, by Savitch's theorem [12], only $\mathcal{O}(m^2)$ deterministic space, which is polynomial in the input's length. □

Generalizing to arbitrary sets, we get the following corollary.

Corollary 2. Avoidability *is in PSPACE.*

We now consider reducing Avoidability to the binary alphabet.

Theorem 7. *The problem of deciding the avoidability of a finite set of partial words over an alphabet of size $k > 2$ reduces in polynomial time to the problem of deciding the avoidability of a finite set of partial words over the binary alphabet.*

Proof. Given a finite set X of partial words over alphabet $A = \{a_1, a_2, \ldots, a_k\}$, we construct a set X' of partial words over the alphabet $B = \{0, 1\}$ such that X' is avoidable if and only if X is avoidable. At a high level, our reduction encodes each symbol in a binary representation and delimits adjacent encodings with a special binary word. Let $l = \lceil \log_2 |A| \rceil$ be the length of an encoding, $d = 101$ be the delimiting word, and define the sets $S = \{00, 11\}$ and $T = B^3 \setminus \{101\}$. Finally, define the function $b : A_\diamond \to S^l \cup (\diamond\diamond)^l$ to be such that $b(a_i)$ equals the binary representation of the natural number $i - 1$, where each bit is replaced with two copies of itself, and $b(\diamond) = (\diamond\diamond)^l$. We now describe the elements in X':

1. First, add every word of length $2l + 3$ which does not contain 101 as a factor in order to ensure that any avoiding infinite word has 101 as a factor.
2. Second, for each $t \in T$, add $101(\diamond\diamond)^l t$. To avoid these words, every occurrence of 101 in an infinite word must be followed by another 101 after $2l$ other bits.
3. Third, for each $0 \leq i < l$, add the words $101(\diamond\diamond)^i 01$ and $101(\diamond\diamond)^i 10$. This forces avoiding words to have only valid binary representations (that is, words from S^l) between consecutive pairs of 101.
4. Fourth, for each word $u_0 u_1 \cdots u_{m-1} \in X$, where each $u_i \in A_\diamond$, add to X' the word $101 b(u_0) 101 b(u_1) \cdots 101 b(u_{m-1})$ which enforces a bijection between words which avoid X and words which avoid X'.

5. Finally, for each $|A| < i \leq 2^l$, add the corresponding binary representation of $i-1$ where each bit is replaced with two copies of itself. This ensures that every factor of length $2l$ delimited by 101 corresponds to a symbol in A.

Suppose some infinite word $w = (w_0 w_1 \cdots w_{n-1})^{\mathbb{Z}}$ avoids X. Then clearly the word $w' = (101b(w_0)101b(w_1)\cdots 101b(w_{n-1}))^{\mathbb{Z}}$ avoids X'. Next, suppose that some infinite word w' avoids X'. Then, to avoid the first part, w' must have 101 as a factor. Additionally, to avoid the second part, following every occurrence of 101 in w' there must be another occurrence of 101 in w' after $2l$ other bits. Furthermore, to avoid the third part, these bits must come in pairs. Moreover, to avoid the last part, these $2l$ bits must form a binary representation of some symbol in A. So one period of our word must be of the form $101b(u_0)101b(u_1)\cdots 101b(u_{n-1})$ for some $u_i \in A_\diamond$. Finally, to avoid the fourth part, $(u_0 u_1 \cdots u_{n-1})^{\mathbb{Z}}$ must avoid X. Note that all but the fourth part of X' are functions of only the size of the alphabet. Because the alphabet is constant, these sets are constant with respect to the input. Therefore, because the fourth part grows linearly with respect to X, this reduction can be performed in polynomial time. □

Theorem 7 shows that problems of deciding avoidability of sets over alphabets of sizes at least two are equivalent with respect to polynomial time reductions; that is, they are all in the same complexity class. A more rigorous analysis of the space complexity of our reduction, in conjunction with Theorem 1, provides an alternate proof of the $\mathcal{NP}$-hardness of the general problem.

Corollary 3. Avoidability *is $\mathcal{NP}$-hard.*

Proof. The problem of deciding the avoidability of a finite set of partial words over an alphabet of size $k \geq 2$ is $\mathcal{NP}$-hard. Indeed, we prove that the reduction from the Directed Hamiltonian Circuit problem followed by the reduction to the binary alphabet is polynomial time, and therefore suffices to show the avoidability problem $\mathcal{NP}$-hard. The reduction in Theorem 1 uses $\mathcal{O}(|V|^3)$ space, while the reduction in Theorem 7 uses $\mathcal{O}(|A|^2 + |X|\log|A|)$ space. Because $A = V$, the composition of these reductions uses $\mathcal{O}(|V|^2 + |V|^3 \log|V|) = \mathcal{O}(|V|^3 \log|V|)$ space. As both are polynomial time, so is their composition. This concludes the proof when $k = 2$. As in [6], for $k > 2$, we simply forbid the other letters, $a_3, \ldots, a_k$, of the alphabet by including them in the set. □

Additionally, Theorem 7 shows that if every finite avoidable set of partial words over some alphabet of size k is avoided by an infinite word with a period bounded by a polynomial in the size of the set, then so is every finite avoidable set over an alphabet of any size. Moreover, by applying Theorem 2, we can reduce all these avoidability problems to the problem of deciding the avoidability of a finite set of partial words of equal length over the binary alphabet. In the next section, we exploit properties of these reduced sets to present some partial results towards a polynomial bound on the period of an avoiding word.

4 Polynomial Bound on Periods of Avoiding Words

In the previous section, we reduced the problem of deciding the avoidability of a finite set of partial words over an alphabet of size $k \geq 2$ to the problem of deciding the avoidability of a finite set of partial words of equal length over the binary alphabet. This reduction simplifies our problem significantly, most notably by allowing us to consider only two parameters when establishing a bound on the shortest period of an avoiding word: the length of the words in the set and the number of elements in the set.

The following theorem establishes a bijection from simple cycles to subset-minimal cycles in de Bruijn graphs (a cycle C in a graph G is *subset-minimal* if there does not exist a shorter cycle D such that every vertex in D is also in C).

Theorem 8. *Let $G(m,k)$ be the de Bruijn graph of order m over an alphabet of size k. There exists a bijection from simple cycles in $G(m,k)$ to subset-minimal cycles in $G(m+1,k)$ which preserves cycle length.*

The first corollary gives the lengths of the longest subset-minimal cycles in de Bruijn graphs, the second, which strengthens Theorem 5, provides a tight bound for periods of avoiding words, and the third is a negative result on polynomially bounded periods.

Corollary 4. *Let $G(m,k)$ be the de Bruijn graph of order m over an alphabet of size k. The length of the longest subset-minimal cycle in $G(m,k)$ is k^{m-1}.*

Corollary 5. *If a finite set of partial words of length m over a k-letter alphabet is avoidable, then it is avoided by a word with period at most k^{m-1}. Furthermore, for every m, a finite set of partial words of length m over a k-letter alphabet exists such that the smallest period of an infinite word which avoids the set is k^{m-1}.*

Corollary 6. *No polynomial function p of m exists such that all avoidable sets of partial words of length m over a k-letter alphabet are avoided by an infinite word with period at most $p(m)$.*

In short, if there exists a polynomial function p from a set of n partial words of length m over a k-letter alphabet to an upper bound on the shortest period of an infinite word which avoids the set, then p is a function of both n and m.

Conjecture 1. If a set of n partial words of length m over a k-letter alphabet is avoidable, it is avoided by an infinite periodic word with period at most mn.

The following propositions present some positive results towards verifying Conjecture 1.

Proposition 2. *Conjecture 1 is true when $n \leq 2$.*

We note that proving Conjecture 1 is much more difficult for sets of partial words than for sets of full words, as we must consider how many fewer elements are needed in a set of partial words. Consequently, the following result is restricted to sets of full words. For positive integers m and k, let $c(m,k)$ be the number of conjugacy classes of words of length m over a k-letter alphabet.

Proposition 3. *Conjecture 1 is true for sets of full words.*

Proof. We show that every avoidable set X of n full words of length m over a k-letter alphabet is avoided by an infinite word of period at most mn. We first claim that $c(m,k) \geq \frac{k^m}{m}$. This follows because there are k^m words of length m and each conjugacy class has at most m elements. We now consider two possible cases: First, suppose $n < \frac{k^m}{m}$. Then because $n < c(m,k)$, some conjugacy class is not represented in X; that is, there exists some word v of length m such that for all $u \in [v]$ and $x \in X$, $u \neq x$. Therefore, the word $v^{\mathbb{Z}}$ of period m avoids X. Now, suppose $n \geq \frac{k^m}{m}$. By Corollary 5, X is avoided by an infinite word with period at most k^{m-1}. The result follows since $mn \geq k^m > k^{m-1}$. □

The difficulty in proving this bound for sets where $n \geq c(m,k)$ arises from a tenuous balance in the constructing sets of smallest cardinality which are avoidable, yet avoided only by words with large periods. In particular, it was shown in [3] that for every m and k, there exists an unavoidable set of words of length m over an alphabet of size k having $c(m,k)$ elements. And so, in proving this bound when $n \geq c(m,k)$, our task is complicated by the existence of unavoidable sets.

5 Sets Avoidable by Aperiodic Infinite Words

We now consider the problem of determining whether or not there is a non-ultimately periodic infinite word avoiding a given set of partial words.

Theorem 9. *There is a polynomial space algorithm to decide if a finite set of partial words over a k-letter alphabet is avoided by a non-ultimately periodic infinite word. Equivalently, there is a polynomial space algorithm to decide if the number of finite words of length n that avoid a finite set of partial words over a k-letter alphabet grows polynomially or exponentially with n.*

Proof. Suppose we are given a finite set X of partial words over a k-letter alphabet. Let us first perform the transformation of X to X' as described in the proof of Theorem 2. The set X' consists of partial words of the same length m, and the words avoiding X' are exactly the words avoiding X.

Let $G(m,k)$ be the de Bruijn graph of order m and let G be the subgraph of $G(m,k)$ induced by the set $\{u \mid u \not\uparrow x \text{ for all } x \in X'\}$. It is clear that there is a non-ultimately periodic word avoiding X' if and only if G contains two distinct directed cycles C_1 and C_2 such that there is a directed path P_1 from C_1 to C_2 and a directed path P_2 from C_2 to C_1. Similarly, the number of finite words of length n that avoid X' grows exponentially with n if and only if there exist C_1, C_2, P_1, and P_2 as described above.

To determine the existence of C_1, C_2, P_1, and P_2, we apply a variation of Algorithm 1. We only describe the changes required to Algorithm 1. Instead of non-deterministically choosing a single word w, we instead choose two distinct words w and v of length m. We then non-deterministically search for cycles in

G from w to w and from v to v of lengths at most k^m, just as in Algorithm 1. Using the same technique, we non-deterministically search for paths P_1 from w to v and P_2 from v to w in G, where P_1 and P_2 have length at most k^m. This non-deterministic algorithm uses only $\mathcal{O}(m)$ space, so there is an equivalent deterministic algorithm that runs in $\mathcal{O}(m^2)$ space by Savitch's theorem [12]. □

Our next theorem uses the probabilistic method and is therefore non- constructive. Let $A_1, \ldots, A_n$ be events in a probability space. A graph $G = (V, E)$ is a *dependency graph* if $V = \{1, \ldots, n\}$ and for all i, A_i is mutually independent of all the A_j's for which there is no edge $\{i, j\} \in E$.

Lemma 2 ([13], Lemma 19.1). *Let $G = (V, E)$ be a dependency graph for events $A_1, \ldots, A_n$ in a probability space. Suppose that the maximum degree of G is d and that there is a real number p for which $\Pr[A_i] \leq p$ for all $i = 1, \ldots, n$. If $4pd \leq 1$, then $\Pr[\cap \overline{A_i}] \geq (1 - 2p)^n > 0$.*

We use the above result, known as Lovász Local Lemma (symmetric version), to prove that if a set X of partial words is not too large, the number of finite words of length n avoiding X grows exponentially.

Theorem 10. *Let X be a set of pwords of length $m \geq 2$ with at most $h < m$ holes over an alphabet A of size $k \geq 2$. If $|X| \leq \frac{k^{m-h}}{4(2m-1)}$, then for $n \geq 1$, there are at least $\left[k\left(1 - \frac{1}{4m-2}\right)\right]^n$ words of length n over A that avoid X. Furthermore, there is a non-ultimately periodic infinite word over A that avoids X.*

Proof. Let n be an arbitrary positive integer and let w be a random word of length n over A. For $i = 1, \ldots, n$, let A_i denote the event that w contains a factor compatible with a partial word in X at position $i - 1$. Let $p = \frac{|X|}{k^{m-h}}$, so that for all i, $\Pr[A_i] \leq p$. To apply the local lemma we may take $d = 2m-1$, since there can be at most $2m - 1$ overlapping pairs of occurrences of factors of length m in w. Observe that for $|X| \leq \frac{k^{m-h}}{4(2m-1)}$, we have $p = \frac{|X|}{k^{m-h}} \leq \frac{1}{4(2m-1)}$, so that $4pd \leq 1$. By the local lemma, with probability at least $(1 - 2p)^n \geq (1 - \frac{1}{4m-2})^n$, w contains no factor compatible with a partial word in X. There are therefore at least $\left[k\left(1 - \frac{1}{4m-2}\right)\right]^n$ words of length n that avoid X. Since $k, m \geq 2$, we have $k(1 - \frac{1}{4m-2}) > 1$, so the number of words of length n that avoid X grows exponentially with n. We conclude by observing that by our discussion in the proof of Theorem 9, there are exponentially many words of length n avoiding X if and only if there is a non-ultimately periodic infinite word avoiding X. □

6 Conclusion and Open Problems

In this paper, we have established the membership of Avoidability in PSPACE, have reduced Avoidability to constant length sets over the binary alphabet, have formulated a conjecture about polynomially bounding periods of infinite avoiding words and have proven it for the special case of sets of full words, have given a polynomial space algorithm that determines if a given finite set of partial words is avoided by a non-ultimately periodic infinite word and that also determines if the number of finite words of length n avoiding the given set grows polynomially or exponentially with n, and have also applied the probabilistic method to show that if a set of partial words is not too large, the number of finite words of length n avoiding it grows exponentially. However, membership of Avoidability in $\mathcal{NP}$ remains open. A World Wide Web server interface has been established at `www.uncg.edu/cmp/research/unavoidablesets3` for automated use of a program that when given as input a finite set of partial words over a given alphabet will output the shortest period of an infinite avoiding word in case the set is avoidable.

References

1. Lothaire, M.: Algebraic Combinatorics on Words. Cambridge University Press, Cambridge (2002)
2. Mykkeltveit, J.: A proof of golomb's conjecture for the de bruijn graph. Journal of Combinatorial Theory, Series B 13, 40–45 (1972)
3. Champarnaud, J., Hansel, G., Perrin, D.: Unavoidable sets of constant length. International Journal of Algebra and Computation 14, 241–251 (2004)
4. Blanchet-Sadri, F., Brownstein, N., Kalcic, A., Palumbo, J., Weyand, T.: Unavoidable sets of partial words. Theory of Computing Systems (to appear, 2009)
5. Aho, A., Corasick, M.: Efficient string machines, an aid to bibliographic research. Communications of the ACM 18, 333–340 (1975)
6. Blanchet-Sadri, F., Jungers, R., Palumbo, J.: Testing avoidability on sets of partial words is hard. Theoretical Computer Science 410, 968–972 (2009)
7. Choffrut, C., Karhumäki, J.: Combinatorics of words. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 329–438. Springer, Heidelberg (1997)
8. Kobayashi, Y.: Repetition-free words. Theoretical Computer Science 44, 175–197 (1986)
9. Goulden, I., Jackson, D.: Combinatorial Enumeration. Dover (2004)
10. Blanchet-Sadri, F.: Algorithmic Combinatorics on Partial Words. Chapman & Hall/CRC Press, Boca Raton (2007)
11. Karp, R.: Reducibility among combinatorial problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum, New York (1972)
12. Savitch, W.J.: Relationship between nondeterministic and deterministic tape classes. Journal of Computer and System Sciences 4, 177–192 (1970)
13. Jukna, S.: Extremal Combinatorics. Springer, Heidelberg (2001)

Closures in Formal Languages and Kuratowski's Theorem

Janusz Brzozowski, Elyot Grant, and Jeffrey Shallit

David R. Cheriton School of Computer Science,
University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
{brzozo,egrant,shallit}@cs.uwaterloo.ca

Abstract. A famous theorem of Kuratowski states that, in a topological space, at most 14 distinct sets can be produced by repeatedly applying the operations of closure and complement to a given set. We re-examine this theorem in the setting of formal languages, where closure is either Kleene closure or positive closure. We classify languages according to the structure of the algebra they generate under iterations of complement and closure. There are precisely 9 such algebras in the case of positive closure, and 12 in the case of Kleene closure. We study how the properties of being open and closed are preserved under concatenation. We investigate analogues, in formal languages, of the separation axioms in topological spaces; one of our main results is that there is a clopen partition separating two words if and only if the words do not commute. We can decide in quadratic time if the language specified by a DFA is closed, but if the language is specified by an NFA, the problem is PSPACE-complete.

1 Introduction

In 1922, Kuratowski proved that, if S is any set in a topological space, then at most 14 distinct sets can be produced by repeatedly applying the operations of topological closure and complement to S [11,7]. Furthermore, there exist sets achieving this bound of 14 in many common topological spaces. There is a large and scattered literature on Kuratowski's theorem, most of which focuses on topological spaces; an admirable survey is the paper of Gardner and Jackson [8]. For the analogous result on relations, see [9].

The basic properties of closure systems and a version of Kuratowski's theorem in a general setting are presented in Section 2; this version can be found in Hammer [10]. Our point of view most closely matches that of Peleg [13], who briefly observed that Kleene and positive closure are closure operators, and hence Kuratowski's theorem holds for them.

Positive and Kleene closures are discussed in Section 3. In Section 4 we reconsider Kuratowski's theorem in the context of formal languages, where closure is replaced by Kleene closure or positive closure. We describe all possible algebras of languages generated by a language under the operations of complement and closure. We classify languages according to the structure of the algebras they generate, and give a language of each type (Theorems 5 and 7).

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 125–144, 2009.

In Section 5 we study how the properties of being open and closed are preserved under concatenation. In Section 6 we investigate analogues, in formal languages, of the separation axioms in topological spaces; one of our main results (Theorem 14) is that there is a clopen partition separating two words if and only if the words do not commute. In Section 7 we show that we can decide in quadratic time if the language specified by a DFA is closed, but if the language is specified by an NFA, the problem is PSPACE-complete.

Because of space limitations, some proofs are omitted or only sketched. For more complete versions, see [2,3].

2 Closure Systems and Kuratowski's Theorem

We recall the definitions and properties of closures in general. Let S be a set which we call the *universal set*. An operator $\square$ operating on a set $X \subseteq S$ will be denoted by $X^\square$. Then a mapping $\square : 2^S \to 2^S$ is a *closure operator* if and only if it satisfies the following, for all subsets X and Y of S:

$$\begin{array}{cl} X \subseteq X^\square & (\square \text{ is extensive}); \\ X \subseteq Y \text{ implies } X^\square \subseteq Y^\square & (\square \text{ is isotone}); \\ X^{\square\square} = X^\square & (\square \text{ is idempotent}). \end{array} \tag{1}$$

A pair $(S,^\square)$ satisfying (1) is a *closure system*. The *complement* $S \setminus X$ of a set $X \subseteq S$ is denoted X^-. The set $X^\square$ is the *closure* of X. We say X is *closed* if $X = X^\square$. Also, X is *open* if its complement is closed, and X is *clopen* if it is both open and closed. The *interior* of X, denoted X°, is defined to be $X^{-\square-}$.

Note the duality between $\square$ and $\circ$: $X^\circ = X^{-\square-}$ and $X^\square = X^{-\circ-}$. This duality also applies to (1), since we have

$$\begin{array}{cl} X \supseteq X^\circ & (\circ \text{ is intensive}); \\ X \subseteq Y \text{ implies } X^\circ \subseteq Y^\circ & (\circ \text{ is isotone}); \\ X^{\circ\circ} = X^\circ & (\circ \text{ is idempotent}). \end{array} \tag{2}$$

Moreover, it is equivalent to define $(S,^\square)$ via an *interior operator* satisfying (2).

We now list some fundamental properties of closure systems.

Proposition 1. *The intersection of an arbitrary family of closed sets is closed.*

Proposition 2. *For $X \subseteq S$, the following are identical: (a) $X^\square$; (b) $\bigcap\{Y \subseteq S : Y \supseteq X \text{ and } Y \text{ is closed}\}$; (c) $\{a \in S : \text{for all open } Y \subseteq S, a \in Y \text{ implies } Y \cap X \neq \emptyset\}$; (d) $X^{-\circ-}$.*

Proposition 3. *Let $X, Y \subseteq S$. Then the following hold:*

(a) $X^\square$ *is closed.*
(b) $(X \cup Y)^\square = (X^\square \cup Y^\square)^\square$.
(c) $(X \cap Y)^\square \subseteq X^\square \cap Y^\square$.

Duals of Propositions 1–3 also hold [2]. For example, the union of an arbitrary family of open sets is open.

We now state two versions of Kuratowski's theorem. The first [11] is equivalent to Kuratowski's original result generalized to an arbitrary closure system, not necessarily topological:

Theorem 1. *Let $(S, \Box)$ be a closure system, and let $X \subseteq S$. Starting with X, apply the operations of closure and complement in any order, any number of times. Then at most* 14 *distinct sets are generated. Also, any $X \subseteq S$ satisfies*

$$X^{\Box-\Box-\Box-\Box} = X^{\Box-\Box}. \tag{3}$$

A closure operator $\Box$ *preserves openness* if $X^{\Box}$ is open for all open sets X, or equivalently, if Y° is closed for all closed sets Y. Hence if $\Box$ preserves openness, then $X^{\Box\circ}$ and $X^{\circ\Box}$ are clopen for all sets X. We will see later that the positive closure of languages preserves openness.

In 1983, Peleg [13] defined a closure operator to be *compact* if it satisfies Eq. (4) below. He showed that at most 10 different sets are generated if $\Box$ is compact, and proved that $\Box$ preserves openness if and only if it is compact. The following theorem is a modified version of Peleg's result:

Theorem 2. *Let $(S, \Box)$ be a closure system such that $\Box$ preserves openness, and let $X \subseteq S$. Starting with X, apply the operations of closure and complement in any order, any number of times. Then at most* 10 *distinct sets are generated. Also, any $X \subseteq S$ satisfies*

$$X^{\Box-\Box-\Box} = X^{\Box-\Box-}. \tag{4}$$

3 Positive and Kleene Closures of Languages

We deal now with closures in the setting of formal languages. Our universal set is Σ^*, the set of all finite words over a finite non-empty alphabet Σ. We consider two closure operators: positive closure and Kleene closure. For $L \subseteq \Sigma^*$, we define $L^- = \Sigma^* \setminus L$, $L^+ = \bigcup_{i \geq 1} L^i$, and $L^* = \bigcup_{i \geq 0} L^i$.

Proposition 4. *Positive closure and Kleene closure are both closure operators.*

We note, importantly, that the positive and Kleene closures are *not* topological (the union of two closed languages is not necessarily closed). As a counterexample, observe that $(aa)^+ \cup (aaa)^+ \subsetneq (aa \cup aaa)^+$, as a^5 belongs to the right-hand side but not the left. Consequently, languages *do not* form a topology under positive or Kleene closure.

A language is *positive-closed* if it is a closed set under positive closure. It is *positive-open* if its complement is positive-closed. The terms *Kleene-closed*, and *Kleene-open* are defined similarly. The *positive interior* of a language L is $L^{\oplus} = L^{-+-}$; the *Kleene interior* is $L^{\circledast} = L^{-*-}$.

Proposition 5. *Let $L \subseteq \Sigma^*$. The following are equivalent:*

(a) L is positive-closed.
(b) $L \cup \{\epsilon\}$ is Kleene-closed.

(c) $L = L^+$.
(d) $L = M^+$ *for some* $M \subseteq \Sigma^*$.
(e) For all $u, v \in L$, *we have* $uv \in L$.

The dual of Proposition 5 also holds [2]. For example, (a) L is positive-open is equivalent to (e) For all $u, v \in \Sigma^*$ such that $uv \in L$, we have $u \in L$ or $v \in L$.

In algebraic terms, $L \subseteq \Sigma^*$ is a *semigroup* if $uv \in L$ for all $u, v \in L$. Proposition 5 states that a language is positive-closed if and only if it is a semigroup. Also, $L \subseteq \Sigma^*$ is Kleene-closed if and only if it is a monoid. We verify that if L is positive-closed, then so are $L \setminus \{\epsilon\}$ and $L \cup \{\epsilon\}$. So there is an obvious 2-to-1 mapping between positive-closed and Kleene-closed languages—positive-closed languages may or may not contain ϵ, and Kleene-closed languages must.

Since positive closure and Kleene closure are so similar, we restrict our attention to positive closure from this point on. This allows us to state our theorems more elegantly, as we need not worry about ϵ. For the remainder of this article, a language is *closed* if it is positive-closed, *open* if it is positive-open, and *clopen* if it is both positive-closed and positive-open.

Example 1. Clopen languages: Let Σ be an alphabet and let $\Sigma_1, \Sigma_2 \subseteq \Sigma$. For $w \in \Sigma^*$, let $|w|_1$ (respectively, $|w|_2$) denote the number of distinct values of i for which $w[i] \in \Sigma_1$ (respectively, $w[i] \in \Sigma_2$). Suppose $k \geq 0$. Then $L = \{w \in \Sigma^* : |w|_1 < k|w|_2\}$ is clopen.

To prove this, let $u, v \in L$. Then $|u|_1 < k|u|_2$ and $|v|_1 < k|v|_2$. But $|uv|_1 = |u|_1 + |v|_1 < k|u|_2 + k|v|_2 = k|uv|_2$, so $uv \in L$, and thus L is closed. By a similar argument, we can prove that $L^- = \{w \in \Sigma^* : |w|_1 \geq k|w|_2\}$ is closed. Thus L is clopen. ■

Example 2. Open languages: A language L is *prefix-closed* if and only if for every $w \in L$, each prefix of w is in L. We analogously define *suffix-closed*, *subword-closed*, and *factor-closed* languages. Here by subword, we mean an arbitrary subsequence, and by factor, we mean a contiguous subsequence. For any $L \subseteq \Sigma^*$, if L is prefix-, suffix-, factor-, or subword-closed, then L is open.

For prefix-closed languages, we show that L satisfies the dual of Proposition 5 (e), which states that $uv \in L$ implies $u \in L$ or $v \in L$. Let $w \in L$ and suppose $w = uv$. Then $u \in L$ if L is prefix-closed, so our characterization holds and L is open. The proof is similar if L is suffix-closed. Since factor- and subword-closed languages are also prefix-closed, the claim holds. ■

Example 3. Closed languages: Left ideals (satisfying $L = \Sigma^* L$), right ideals ($L = L\Sigma^*$), two-sided ideals ($L = \Sigma^* L \Sigma^*$), or languages of the form $L = \bigcup_{a_1 \cdots a_n \in L} \Sigma^* a_1 \Sigma^* \cdots \Sigma^* a_n \Sigma^*$, all satisfy $L = L^+$, and so are positive closed. ■

In the 1970's, D. Forkes proved Eq. (3) with the Kleene closure as $\square$, and the first author then proved that Eq. (4) holds when $\square$ is positive closure. (They were both unaware of [11].) Peleg [13] proved this over a wider class of operators. Here, we state an equivalent fact: positive closure preserves openness.

Theorem 3. *Let $L \subseteq \Sigma^*$ be open. Then L^+ is open.*

Proof. This follows from the facts that Eq. (4) holds for positive closure, and that Eq. (4) is equivalent to compactness. ■

By arguments similar to those in the proof of Theorem 2, we may conclude:

Corollary 1. *Let $L \subseteq \Sigma^*$. Then $L^{+\oplus}$ and $L^{\oplus+}$ are clopen. Moreover, if L is open, then L^+ is clopen, and if L is closed, then $L^\oplus$ is clopen.*

The converses of the above results are false; for example, there exist languages such as $\{a, aaaa\}$ which are not open, but have clopen closures. We discuss such possibilities extensively in the next section. For now, we give a characterization of the languages with clopen closures and clopen interiors.

Theorem 4. *Let $L \subseteq \Sigma^*$.*

(a) L^+ is clopen iff there exists an open language M with $L \subseteq M \subseteq L^+$.
(b) $L^\oplus$ is clopen iff there exists a closed language M with $L \supseteq M \supseteq L^\oplus$.

Proof. We prove only (a); (b) can be proved using a similar argument. The forward direction of (a) is trivial since we can take $M = L^+$. For the converse, we note that $L \subseteq M$ implies $L^+ \subseteq M^+$ by isotonicity, and $M \subseteq L^+$ implies $M^+ \subseteq L^{++} = L^+$ by isotonicity and idempotency. Thus $M^+ = L^+$, and since M^+ is the closure of an open language, it is clopen and the result follows. ■

4 Kuratowski's Theorem for Languages

For any language L, let $A(L)$ be the family of all languages generated from L by complementation and positive closure. Since positive closure preserves openness, Theorem 2 implies that $A(L)$ contains at most 10 languages. As we shall see, this upper bound is tight. Moreover, there are precisely 9 distinct finite algebras $(A(L), ^+, ^-)$. Since the languages in $A(L)$ must occur in complementary pairs, there can only exist algebras containing 2, 4, 6, 8, or 10 distinct languages. We will provide a list of conditions that classify languages according to the structure of $(A(L), ^+, ^-)$, and thus completely describe the circumstances under which $|A(L)|$ is equal to 2, 4, 6, 8, or 10.

We will also explore Kleene closure, where there are subtle differences. Let $D(L)$ be the family of all languages generated from L by complementation and Kleene closure. Kleene closure does not preserve openness, since Kleene-closed languages contain ϵ and Kleene-open languages do not. Therefore we must fall back to Theorem 1, which implies that $D(L)$ contains at most 14 languages, and we will show that this bound is also tight. There are precisely 12 distinct finite algebras $(D(L), ^*, ^-)$. We shall describe these algebras by relating them to those in the positive case.

In a sense, our results are the formal language analogue of topological results obtained by Chagrov [5] and discussed in [8]. Peleg [13] noted the tightness of the bounds of 10 and 14 in the positive and Kleene cases, but went no further.

4.1 Structures of the Algebras with Positive Closure

We may better understand the structure of $A(L)$ by first analyzing a related algebra of languages. Let $B(L)$ be the family of all languages generated from L by positive closure and positive interior, and let $C(L) = \{M : M^- \in B(L)\}$ be their complements. Recall that the closure of an open language is clopen and the interior of a closed language is clopen by Corollary 1. Since the closure and interior operators are idempotent on the clopen languages $L^{+\oplus}$ and $L^{\oplus +}$, it follows that $B(L) = \{L, L^+, L^{+\oplus}, L^{\oplus}, L^{\oplus +}\}$. Of course, these five languages may not all be distinct; we will address this later. At the moment, we provide the following proposition, which demonstrates that it suffices to analyze the structure of $B(L)$ to determine the structure of $A(L)$.

Proposition 6. *Let $L \subseteq \Sigma^*$. Then $A(L) = B(L) \cup C(L)$, and the union is disjoint.*

Proof. Clearly $A(L) \supseteq B(L) \cup C(L)$, since any language generated from L by closure, interior, and complement can be generated using only closure and complement, by the identity $L^{\oplus} = L^{-+-}$. To prove the reverse inclusion, we let $M \in A(L)$. Then there is some string of symbols $z \in \{+,-\}^*$ such that $M = L^z$. We constuct a string $z' \in \{+,-,\oplus\}^*$ by starting with z and repeatedly replacing all instances of $-+$ by $\oplus-$ and all instances of $-\oplus$ by $+-$, until no such replacements are possible. Since $L^{-+} = L^{\oplus -}$ and $L^{-\oplus} = L^{+-}$, we have $M = L^{z'}$. However, in producing z', we effectively shuffle all complements to the right. Consequently, the operation performed by z' is a series of positive closures and interiors followed by an even or odd number of complements. Hence either $M \in B(L)$ or $M \in C(L)$, and thus $A(L) = B(L) \cup C(L)$.

We now prove that $B(L) \cap C(L) = \emptyset$. We assume otherwise to obtain a contradiction; $B(L)$ must then contain some complementary pair of languages M and M^-. We note that $L^{\oplus} \subseteq L^{\oplus +}$ by extensivity, $L^{\oplus} \subseteq L \subseteq L^+$ by intensivity and extensivity, and $L^{\oplus} \subseteq L^{+\oplus}$ by isotonicity, and hence $L^{\oplus} \subseteq M$ for all $M \in B(L)$. Thus for two languages in $B(L)$ to be complements, $L^{\oplus}$ must be empty. Then L contains no strings of length 1, and hence L^+ and $L^{+\oplus}$ do not either. But then no language in $B(L)$ contains a string of length 1, and thus no pair of languages in $B(L)$ are complements, and we have our contradiction. ■

Proposition 6 implies that $|A(L)| = 2|B(L)|$, and moreover that there is an exact 1-to-2 correspondence between the languages in $B(L)$ and $A(L)$: each language in $B(L)$ can be associated with itself and its complement. Hence the algebra $(A(L),^+,^-)$ can be constructed by simply merging the two algebras $(B(L),^+,^\oplus)$ and $(C(L),^+,^\oplus)$ and adding the complement operator. Thus we have reduced the problem of describing all algebras $(A(L),^+,^-)$ to the simpler task of describing the algebras $(B(L),^+,^\oplus)$. Before we proceed, we need to exclude a possible case via the following:

Lemma 1. *Suppose $L \subseteq \Sigma^*$. If L^+ and $L^{\oplus}$ are both clopen, then L must be open or closed.*

Proof. Seeking a contradiction, we assume that both L^+ and $L^\oplus$ are clopen but L is neither open nor closed. If L is not open, then $L \setminus L^\oplus$ is non-empty.

Let w be the shortest word in $L \setminus L^\oplus$. Consider $M = L^\oplus \cup \{w\}$. It must not be open, because if it were, we would have $M \subseteq L^\oplus$ by the dual to Proposition 2. Then the dual of Proposition 5 (e) must fail to hold for some word in M. But it holds for all words in $L^\oplus$ and thus must fail for w. Then there exist non-empty words x and y with $xy = w$, but $x \notin M$ and $y \notin M$. Then neither x nor y is in $L^\oplus$.

By our assumption that L^+ is open, the fact that $w \in L^+$ implies that either $x \in L^+$ or $y \in L^+$. Without loss of generality, suppose that $x \in L^+$. Then x is the concatenation of a list of words from L; we write $x = u_1 u_2 \cdots u_n$ with $u_i \in L$ for all $1 \leq i \leq n$. Then $|u_i| \leq |x| < |w|$ for all i, and thus $u_i \in L^\oplus$ for all i by our definition of w as the shortest word in $L \setminus L^\oplus$. However, x is then the concatenation of a list of words from $L^\oplus$ and is thus an element of $L^{\oplus +}$, which is $L^\oplus$ since we assumed $L^\oplus$ was closed. This is a contradiction since $x \notin L^\oplus$. ■

Finally, we characterize the 9 possible algebras $(B(L), ^+, ^\oplus)$. Table 1 classifies all languages according to the structures of the algebras they generate and gives an example of each type. Here, we briefly explain our analysis. Clearly $B(L) = \{L\}$ if and only if L is clopen, giving Case (1). If L is open but not closed, then $B(L) = \{L, L^+\}$ since L^+ must then be clopen. Similarly, if L is closed but not open, then $B(L) = \{L, L^\oplus\}$. These situations yield Cases (2) and (3). We henceforth assume that L is neither open nor closed, and thus L, $L^\oplus$, and L^+ are all different. The remaining cases depend on the values of $L^{\oplus +}$ and $L^{+\oplus}$. Both must be clopen, so neither can equal L. Lemma 1 proves that $L^\oplus$ and L^+ cannot both be clopen. If neither $L^\oplus$ nor L^+ are clopen, then we have Case (8) if $L^{\oplus +}$ and $L^{+\oplus}$ are equal, and Case (9) if they are not. The remaining cases occur when one of L^+ and $L^\oplus$ is clopen and the other is not. If L^+ is clopen and $L^\oplus$ is not, then we get Case (4) if $L^{\oplus +} = L^+$ and Case (6) otherwise. Analogously, if $L^\oplus$ is clopen and L^+ is not, then we get Case (5) if $L^{+\oplus} = L^\oplus$ and Case (7) otherwise.

We see that if $(B(L), ^+, ^\oplus)$ has algebraic structure (2), then $(C(L), ^+, ^\oplus)$ has structure (3). Thus we shall say that Case (3) is the *dual* of Case (2). By examining the conditions under which each case holds, we can easily see that Cases (4) and (5) are also duals, as are Cases (6) and (7). Cases (1), (8), and (9) are self-dual. This notion is useful in constructing the algebra $(A(L), ^+, ^-)$; we connect an instance of $(B(L), ^+, ^\oplus)$ to its dual structure in the obvious way via the complement operator. Figure 1 gives an example of this for Case (6).

In summary, we have proven

Theorem 5. *Start with any language L, and apply the operators of positive closure and complement in any order, any number of times. Then at most* 10 *distinct languages are generated, and this bound is optimal. Furthermore, Table 1 classifies languages according to the algebra they generate and gives a language generating each algebra.*

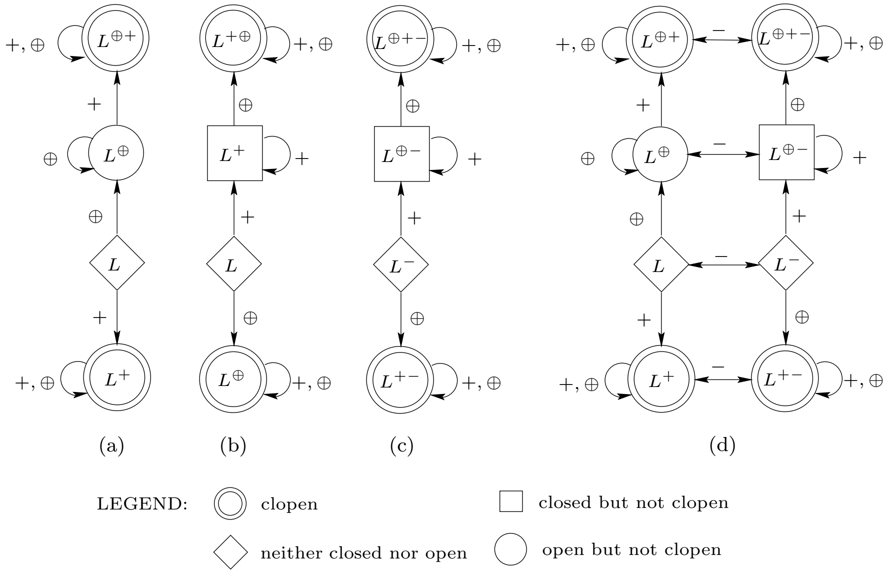


Fig. 1. Construction of $A(L)$, Case (6): (a) $(B(L),^+,^\oplus)$, Case (6); (b) $(B(L),^+,^\oplus)$, Case (7), the dual of Case (6) obtained by interchanging $+$ with $\oplus$, and "open" with "closed"; (c) $(C(L),^+,^\oplus)$, that is, $(B(L),^+,^\oplus)$, Case (7), with elements renamed as complements of those of Case (6); (d) $A(L)$ constructed from $B(L)$ and $C(L)$

In the unary case, we obtain the following:

Theorem 6. *Start with any unary language L, and apply the operators of positive closure and complement in any order, any number of times. Then at most 6 distinct languages are generated, and this bound is optimal. Furthermore, precisely cases (1) through (5) in Table 1 are possible for a unary language.*

Note that all the example languages are regular. Hence Theorems 5 and 6 also hold for any regular language and any regular unary language, respectively.

4.2 Structures of the Algebras with Kleene Closure

As we did in the positive case, first we restrict ourselves to closure and interior. Let $E(L)$ be the family of all languages generated from L by Kleene closure and Kleene interior, and let $F(L) = \{M : M^- \in E(L)\}$ be their complements. Our next results relate $D(L)$ and $E(L)$ to $A(L)$ and $B(L)$. Our discussion involves both closure operators, so we will be explicit about which closure properties we are invoking (although the word *clopen* will still mean positive-clopen). We first claim the following, which can be proven in the same manner as Proposition 6:

Proposition 7. *Let $L \subseteq \Sigma^*$. Then $D(L) = E(L) \cup F(L)$, and the union is disjoint.*

Table 1. Classification of languages by the structure of $(B(L),^{+},^{\oplus})$

Case	Necessary and Sufficient Conditions	$\|B(L)\|$	$\|A(L)\|$	Example	Dual
(1)	L is clopen.	1	2	a^*	(1)
(2)	L is open but not closed.	2	4	a	(3)
(3)	L is closed but not open.	2	4	aaa^*	(2)
(4)	L is neither open nor closed; L^+ is clopen and $L^{\oplus+} = L^+$.	3	6	$a \cup aaaa$	(5)
(5)	L is neither open nor closed; $L^{\oplus}$ is clopen and $L^{+\oplus} = L^{\oplus}$.	3	6	aa	(4)
(6)	L is neither open nor closed; L^+ is open but $L^{\oplus}$ is not closed; $L^{\oplus+} \neq L^+$.	4	8	$a \cup abaa$	(7)
(7)	L is neither open nor closed; $L^{\oplus}$ is closed but L^+ is not open; $L^{+\oplus} \neq L^{\oplus}$.	4	8	$(a \cup b)^* \setminus (a \cup abaa)$	(6)
(8)	L is neither open nor closed; $L^{\oplus}$ is not closed and L^+ is not open; $L^{+\oplus} = L^{\oplus+}$.	4	8	$a \cup bb$	(8)
(9)	L is neither open nor closed; $L^{\oplus}$ is not closed and L^+ is not open; $L^{+\oplus} \neq L^{\oplus+}$.	5	10	$a \cup ab \cup bb$	(9)

Next, we give a way of relating $E(L)$ to $B(L)$. We recall that $L^* = L^+ \cup \{\epsilon\}$ and $L^{\circledast} = L^{\oplus} \setminus \{\epsilon\}$. Consequently, $E(L) \subseteq \bigcup_{M \in B(L)} \{M \cup \{\epsilon\}, M \setminus \{\epsilon\}\}$. We now know enough to explicitly determine $D(L)$ in the following case:

Proposition 8. *Let $L \subseteq \Sigma^*$ be clopen. Then $D(L) = \{L \cup \{\epsilon\}, L \setminus \{\epsilon\}, L^- \cup \{\epsilon\}, L^- \setminus \{\epsilon\}\}$.*

Since the operations of positive closure and positive interior preserve the presence or absence of ϵ in a language, we may also note that if $\epsilon \in L$, then all languages in $B(L)$ contain ϵ, and conversely if $\epsilon \notin L$, then no language in $B(L)$ contains ϵ. For $M \in E(L)$, we write $\phi(M)$ to denote either $M \cup \{\epsilon\}$ or $M \setminus \{\epsilon\}$, whichever lies in $B(L)$. We note that $\phi(M)$, $\phi(M \cup \{\epsilon\})$, and $\phi(M \setminus \{\epsilon\})$ are equal. Moreover, we note that $\phi(M^*) = \phi(M)^+$ and $\phi(M^{\circledast}) = \phi(M)^{\oplus}$; ϕ can therefore be thought of as a homomorphism from $E(L)$ to $B(L)$. Consequently, $E(L) \subseteq \{M : \phi(M) \in B(L)\}$. We use this idea and the classifications of Table 1 to determine all possible algebras $(E(L),^{*},^{\circledast})$. As we shall see, there are precisely 12 distinct algebras, each containing at most 14 elements.

We have seen what happens in Case (1) when L is clopen; two algebras are possible depending on whether $\epsilon \in L$ or not, and we refer to these as Cases (1a) and (1b) respectively. We next examine Cases (2) and (3), in which L is not clopen but is open or closed. Suppose L is open but not clopen, and hence $B(L) = \{L, L^+\}$. Then L^* is clopen and thus $E(L^* = \{L^*, L^* \setminus \{\epsilon\}\}$. Since $E(L^*) \subseteq E(L)$ we thus have $\{L, L^*, L^* \setminus \{\epsilon\}\} \subseteq E(L) \subseteq \{M : \phi(M) \in \{L, L^+\}\}$. Therefore, we have two cases; either one or both of $L \setminus \{\epsilon\}$ and $L \cup \{\epsilon\}$ may be in $E(L)$, depending on whether or not $L^{\circledast} = L$. If $\epsilon \notin L$, then $L^{\circledast} = L$ and thus $E(L) = \{L, L^*, L^* \setminus \{\epsilon\}\}$. If $\epsilon \in L$, then $L^{\circledast} = L \setminus \{\epsilon\}$ and thus $E(L) = \{L, L \setminus \{\epsilon\}, L^*, L^* \setminus \{\epsilon\}\}$. We refer to these situations as Cases (2a) and (2b) respectively.

Similar possibilities occur when L is closed but not clopen. If $\epsilon \in L$ then $E(L) = \{L, L^{\circledast}, L^{\circledast} \cup \{\epsilon\}\}$. If $\epsilon \notin L$ then $L^* = L \cup \{\epsilon\}$ and thus $E(L) = \{L, L \cup \{\epsilon\}, L^{\circledast}, L^{\circledast} \cup \{\epsilon\}\}$. We refer to these situations as Cases (3a) and (3b) respectively.

We now turn to Cases (4)–(9), when L is neither closed nor open.

Lemma 2. *Let $L \subseteq \Sigma^*$ be neither open nor closed. Then*

$$\begin{aligned} E(L) = \{L\} &\cup \{M \cup \{\epsilon\} : M \in B(L) \text{ and } M \text{ closed}\} \\ &\cup \{M \setminus \{\epsilon\} : M \in B(L) \text{ and } M \text{ open}\}. \end{aligned}$$

Proof. Clearly $L \in E(L)$. We claim that no other language M with $\phi(M) = L$ can be in $E(L)$. If we suppose otherwise, then such an M must be generated by taking the Kleene closure or interior of some other language in $E(L)$. This would imply that M is open or closed, which is impossible since $\phi(M) = L$ and L is neither open nor closed.

For each remaining $M \in B(L) \setminus \{L\}$, we wish to show that $M \cup \{\epsilon\} \in E(L)$ if and only if M is closed, and $M \setminus \{\epsilon\} \in E(L)$ if and only if M is open. Let $M \in B(L) \setminus \{L\}$ be generated by some non-empty sequence S of positive closures and positive interiors. If we replace each positive closure by a Kleene closure and each positive interior by a Kleene interior, then we obtain a sequence S' that generates some $M' \in E(L)$ with $\phi(M') = M$. Now M' contains ϵ if and only if the last operation in S' was a Kleene closure. If M is closed, we may append a final positive closure to any such S to obtain one in which the last operation is a closure. Conversely, if there exists an S whose last operation is a closure, then M must be closed. Thus there exists an $M' \in E(L)$ containing ϵ with $\phi(M') = M$ if and only if M is closed. By a similar argument, there exists an $M' \in E(L)$ not containing ϵ with $\phi(M') = M$ if and only if M is open. The result follows. ■

Lemma 2 allows us to describe the structure of the algebra $(E(L), ^*, ^{\circledast})$ in Cases (4) through (9). Algebra $E(L)$ contains $M \cup \{\epsilon\}$ for all closed M in $B(L)$, $M \setminus \{\epsilon\}$ for all open M in $B(L)$, and both for all clopen M in $B(L)$.

We classify the 12 distinct algebras in Table 2. The conditions are identical to those found in Table 1; the only differences lie in Cases (1), (2), and (3), where the initial presence or absence of ϵ can affect the structure of the algebra.

We now summarize our results for the Kleene case:

Theorem 7. *Start with any language L, and apply the operators of Kleene closure and complement in any order, any number of times. Then at most* 14 *distinct languages are generated, and this bound is optimal. Furthermore, Table 2 describes the 12 algebras generated by this process, classifies languages according to the algebra they generate, and gives a language generating each algebra.*

Table 2. Classification of languages by the structure of $(E(L),^*,^\circledast)$

Case	Necessary and Sufficient Conditions	$\lvert E(L)\rvert$	$\lvert D(L)\rvert$	Example	Dual
(1a)	L is clopen; $\epsilon \in L$.	2	4	a^*	(1b)
(1b)	L is clopen; $\epsilon \notin L$.	2	4	a^+	(1a)
(2a)	L is open but not clopen; $\epsilon \in L$.	3	6	$a \cup \epsilon$	(3a)
(2b)	L is open but not clopen; $\epsilon \notin L$.	4	8	a	(3b)
(3a)	L is closed but not clopen; $\epsilon \notin L$.	3	6	aaa^*	(2a)
(3b)	L is closed but not clopen; $\epsilon \in L$.	4	8	$aaa^* \cup \epsilon$	(2b)
(4)	L is neither open nor closed; L^+ is clopen and $L^{\oplus +} = L^+$.	4	8	$a \cup aaa$	(5)
(5)	L is neither open nor closed; $L^\oplus$ is clopen and $L^{+\oplus} = L^\oplus$.	4	8	aa	(4)
(6)	L is neither open nor closed; L^+ is open but $L^\oplus$ is not closed; $L^{\oplus +} \neq L^+$.	6	12	$a \cup abaa$	(7)
(7)	L is neither open nor closed; $L^\oplus$ is closed but L^+ is not open; $L^{+\oplus} \neq L^\oplus$.	6	12	$(a \cup b)^* \setminus (a \cup abaa)$	(6)
(8)	L is neither open nor closed; $L^\oplus$ is not closed and L^+ is not open; $L^{+\oplus} = L^{\oplus +}$.	5	10	$a \cup bb$	(8)
(9)	L is neither open nor closed; $L^\oplus$ is not closed and L^+ is not open; $L^{+\oplus} \neq L^{\oplus +}$.	7	14	$a \cup ab \cup bb$	(9)

Theorem 8. *Start with any unary language L, and apply the operators of positive closure and complement in any order, any number of times. Then at most* 8 *distinct languages are generated, and this bound is optimal. Furthermore, precisely cases (1a) through (5) in Table 2 describe the* 8 *possible algebras that can be generated from a unary language by this process.*

5 Closure Operators and Concatenation

We note that the concatenation of two closed languages need not be closed, and that the concatenation of two open languages need not be open. For example, consider the languages $L = \{a\}^+$ and $M = \{b\}^+$ for $a, b \in \Sigma$, which are both clopen (under positive closure). Then $ab \in LM$ but $abab \notin LM$, so LM is not closed. Additionally, $ab \in LM$, but neither a nor b is in LM, so LM is not open. However, we do have several results regarding cases when the concatenation of closed or open languages must be closed or open.

Here, we deal mainly with positive closure, but most of our theorems have obvious analogues for the Kleene closure. However, the presence or absence of ϵ can be crucial when dealing with concatenation of languages, so we mention a few exceptional cases where the choice of positive or Kleene closure is important.

Theorem 9. *Let $L, M \subseteq \Sigma^*$.*

(a) Suppose L is positive-closed, and let k be a positive integer. Then $L^k \subseteq L$ and L^k is positive-closed.
(b) Suppose L is Kleene-closed, and let k be a positive integer. Then $L^k = L$.
(c) Suppose L and M are positive-closed (respectively, Kleene-closed) and satisfy $LM = ML$. Then LM is positive-closed (respectively, Kleene-closed).
(d) Suppose L and M are positive-closed (respectively, Kleene-closed) unary languages. Then LM is positive-closed (respectively, Kleene-closed).

Proof. (a) If L is positive-closed then $L = L^+$, and $L^k \subseteq L^+ = L$. Also, for $k > 1$, $L^k L^k = L^{k-1} L^{k+1} \subseteq L^{k-1} L = L^k$ and L^k is positive-closed.
(b) If L is Kleene-closed, then $L^k = (L^*)^k \subseteq (L^*)^* = L^* = L$, and $L \subseteq L^* = (L^*)^k$.
(c) For positive closure, $LMLM = LLMM \subseteq LM$; hence LM is positive-closed.
(d) This is a special case of part (c), since unary languages commute. ■

Theorem 10. *Let $L, M \subseteq \Sigma^*$. Suppose L and M are positive-closed (respectively, Kleene-closed) and such that $L \cup M$ is positive-closed. Then*

(a) LM is positive-closed (respectively, Kleene-closed).
(b) More generally, consider the semigroup of languages $\{L, M\}^+$ generated by L and M. Let $W \in \{L, M\}^+$. Then W is positive-closed (respectively, Kleene-closed) when considered as a language over Σ.

Proof. (a) It suffices to show that $(LM)^k \subseteq LM$; we do this by induction on k. For $k > 1$, $(LM)^k \subseteq L(L \cup M)(L \cup M)M(LM)^{k-2} \subseteq L(L \cup M)M(LM)^{k-2} = (LLM \cup LMM)(LM)^{k-2} \subseteq LM(LM)^{k-2} = (LM)^{k-1} \subseteq LM$.
(b) The cases where $W = L^k$ or $W = M^k$ are proven by Theorem 9, so we may assume that W contains at least one L and one M (when considered as a word in $\{L, M\}^+$.) This implies that either LM or ML is a factor of W. Without loss of generality suppose that LM is a factor of W. Let $W = W_1 W_2 \cdots W_k W_{k+1} \cdots W_n$ where $W_i \in \{L, M\}$ for all i, and specifically $W_k = L$ and $W_{k+1} = M$. Now, to prove that W is closed, we let $u, v \in W$ be words in Σ^*. We show that $uv \in W$. Let $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_n$ where $u_i, v_i \in W_i$ for all i. Consider $x = u_{k+1} \cdots u_n v_1 \cdots v_k$, a factor of uv. We see that $x \in (L \cup M)^+$. But since $L \cup M$ is positive-closed, $(L \cup M)^+ = L \cup M$, and hence either $x \in L$ or $x \in M$. If $x \in L$, then $u_k x \in L = W_k$ by closure of L and thus $uv = u_1 \cdots u_{k-1}(u_k x) v_{k+1} \cdots v_n \in W_1 \cdots W_n = W$. If $x \in M$, then $x v_{k+1} \in M = W_{k+1}$ by closure of M and thus $uv = u_1 \cdots u_k (x v_{k+1}) v_{k+2} \cdots v_n \in W_1 \cdots W_n = W$. So we must have $uv \in W$ in either case, and thus W is closed. For the Kleene-closed case, we again simply note that if $\epsilon \in L$ and $\epsilon \in M$, then $\epsilon \in W$. ■

Theorem 11. *Let L and M be open.*

(a) Suppose $\epsilon \in L$ and $\epsilon \in M$. Then LM is open.
(b) Suppose $\epsilon \notin L$ and $\epsilon \notin M$. Then LM is open if and only if $L = \emptyset$ or $M = \emptyset$.

(c) LL is open if and only if $\epsilon \in L$ or $L = \emptyset$.

(d) If neither L nor M is empty and $\epsilon \in L \cup M$ but $\epsilon \notin L \cap M$, then we may or may not have LM open, even in the unary case.

Proof. (a) Let $ab \in LM$ where $a \in L$ and $b \in M$. Let $ab = uv$ for some words u and v. To prove that LM is open, we must show that either $u \in LM$ or $v \in LM$. We have two cases: either u is a prefix of a, or v is a suffix of b.

If u is a prefix of a, let $a = ux$, so $ab = uxb$ and hence $v = xb$. Since L is open, applying Proposition 5 (b) to $a \in L$ implies that either $u \in L$ or $x \in L$. If $u \in L$, then, since $\epsilon \in M$, we have $u = u\epsilon \in LM$ and we are done. If $x \in L$, then $v = xb \in LM$ and we are also done.

The case where v is a prefix of b is similar and relies on the fact that $\epsilon \in L$.

(b) If $L = \emptyset$ or $M = \emptyset$, then $LM = \emptyset$, which is open. Conversely, if $\epsilon \notin L$, $\epsilon \notin M$, and neither $L = \emptyset$ nor $M = \emptyset$, then LM is non-empty but contains no words of length 0 or 1 and is thus not open.

(c) This follows immediately from parts (a) and (b).

(d) If $L = \{\epsilon, a, aaa, aaaaa\}$ and $M = \{a\}$ (which are both easily verified to be open), then we have $aaaaaa \in LM$, but $aaa \notin LM$, and thus LM is not open. On the other hand, if $L = \{\epsilon, a, aaa\}$ and $M = \{a\}$, then $LM = \{a, aa, aaaa\}$, which is clearly open. ■

Theorem 12. *Let $L, M \subseteq \Sigma^*$ both be clopen.*

(a) If $L \cup M = \Sigma^*$, then LM is clopen.

(b) Suppose that $L \cup M = \Sigma^*$ and consider the semigroup of languages $\{L, M\}^+$ generated by L and M. Let $W \in \{L, M\}^+$. Then W is clopen if and only if $W = \emptyset$ or W contains at most one occurrence of a language which does not contain ϵ.

(c) The converses of the above statements are false; indeed, it is possible that LM is clopen, but $L \cup M$ is not even positive-closed.

Proof. (a) From Theorem 10 (a) we have that LM is closed, since Σ^* is closed. To show that LM is open, let $ab \in LM$ where $a \in L$ and $b \in M$. Let $ab = uv$ for some words u and v. To prove that LM is open, we must show that either $u \in LM$ or $v \in LM$. There are two cases: either u is a prefix of a, or v is a suffix of b.

Without loss of generality, we assume that u is a prefix of a and let $a = ux$, so $ab = uxb$ and hence $v = xb$. Since L is open, applying Proposition 5 (b) to $a \in L$ implies that either $u \in L$ or $x \in L$. If $x \in L$, then $v = xb \in LM$ and we are done. Otherwise, we have $x \notin L$, implying $u \in L$ and $x \in M$ since $L \cup M = \Sigma^*$. If $\epsilon \in M$, $u = u\epsilon \in LM$ and we are done. Otherwise, we have $\epsilon \notin M$, and thus $\epsilon \in L$ since $L \cup M = \Sigma^*$. In this case, we note that $xb \in M$ since $x \in M$, $b \in M$, and M is closed. Then $\epsilon xb = v \in LM$. So in all cases, we have either $u \in LM$ or $v \in LM$. Thus LM is open and hence is clopen.

(b) Let $W = W_1 W_2 \cdots W_n$ where $W_i \in \{L, M\}$ for all i. By Theorem 10 (b), W is closed. If each W_i contains ϵ, then W is open by repeated applications of Theorem 11 (a) and is thus clopen.

If there exist i and j with $i \neq j$, $\epsilon \notin W_i$, and $\epsilon \notin W_j$, then W contains no words of length 1, so either $W = \emptyset$ or W is not open (and thus not clopen).

Finally, we deal with the case where there exists a unique i such that $\epsilon \notin W_i$. Suppose, without loss of generality, that $W_i = M$. Then $W = L^{i-1}ML^{n-i}$. Since $L \cup M$ is Kleene-closed, it must contain ϵ, so $\epsilon \in L$. Thus $L^k = L$ for all positive k by Theorem 9 (b), so we must have $W = M$, $W = LM$, $W = ML$, or $W = LML$. In the first case, $W = M$ is known to be clopen, and in the second and third cases, W is clopen by part (a). Thus we must only consider the case where $W = LML$. We know that LM is clopen by part (a). Furthermore, $M \subseteq LM$ since $\epsilon \in L$, so $LM \cup L \supseteq M \cup L = \Sigma^*$ and thus $LM \cup L = \Sigma^*$. Thus we can apply part (a) on LM and L, proving that LML is clopen.

(c) As a counterexample, we let $L = \{\epsilon\} \cup \{w \in \{a,b\}^* : |w|_a < |w|_b\}$ and let $M = \{\epsilon\} \cup \{w \in \{a,b\}^* : |w|_a > |w|_b\}$, where by $|w|_c$ for a letter c, we mean the number of occurrences of c in w. As we proved in Example 1, L and M are both clopen. Furthermore, L and M both contain ϵ, so LM is open by Theorem 11.

Next, we show that LM is closed. Let $u, v \in LM$, then let $u = u_1u_2$ and $v = v_1v_2$, where $u_1, v_1 \in L$ and $u_2, v_2 \in M$. We observe that $|u_1|_a < |u_1|_b$ and $|v_2|_a > |v_2|_b$. We examine the factor u_2v_1 and consider two cases. If $|u_2v_1|_a \geq |u_2v_1|_b$, then $|u_2v_1v_2|_a > |u_2v_1v_2|_b$ and thus $u_2v_1v_2 \in M$. Since $u_1 \in L$, we must then have $uv = u_1u_2v_1v_2 \in LM$. Similarly, if $|u_2v_1|_a \leq |u_2v_1|_b$, then $|u_1u_2v_1|_a < |u_1u_2v_1|_b$ and thus $u_1u_2v_1 \in L$. Since $v_2 \in M$, we must then have $uv = u_1u_2v_1v_2 \in LM$. So in all cases, $uv \in LM$, and LM is closed. Hence LM is clopen.

However, $L \cup M$ is not closed, since we have $b \in L \subseteq L \cup M$ and $a \in M \subseteq L \cup M$, but $ba \notin L \cup M$. ■

6 Separation of Words and Languages

Next, we discuss analogies of the separation axioms of topology in the realm of languages. Although languages do not form a topology under Kleene or positive closure, there are many interesting results describing when there exist open, closed, and clopen languages that separate given words or languages. In most of these theorems, we only consider words in Σ^+, as ϵ is always a trivial case.

Lemma 3. *Let $w \in \Sigma^+$, and let $L \subseteq \Sigma^*$ be closed with $w \notin L$. Then there exists a finite open language M such that $w \in M$ but $M \cap L = \emptyset$,*

Proof. We simply take $M = L^- \cap \{x \in \Sigma^+ : |x| \leq |w|\}$. This is clearly finite, and is open by the dual to part (e) of Proposition 5. ■

Theorem 13. *Let $u, v \in \Sigma^+$.*

(a) There exists an open language L with $u \in L$ and $v \notin L$ if and only if for all natural numbers k, we have $u \neq v^k$.

(b) If $u \neq v$, then either there exists an open language L with $u \in L$ and $v \notin L$, or there exists an open language L with $u \notin L$ and $v \in L$. (In other words, all words are distinguishable by open languages.)

Proof. (a) For the forward direction, we note that if $u = v^k$ for some positive k, then any open language containing u must contain v by Proposition 5 (b). For the reverse direction, we apply Lemma 3 to u and $\{v\}^+$, which is closed.

(b) Without loss of generality, let $|u| \leq |v|$. This implies that, for all k, $u \neq v^k$, and hence the claim follows from (a). ■

We now recall a basic result from combinatorics on words (see, e.g., [12]). Recall that a word w is *primitive* if it cannot be expressed in the form x^k for a word x and an integer $k \geq 2$.

Lemma 4. *Let $u, v \in \Sigma^+$. The following are equivalent:*

(1) $uv = vu$, that is, u and v commute.

(2) There exists a word x and integers $p \geq 1$ and $q \geq 1$ such that $u = x^p$ and $v = x^q$.

(3) There exists a word y and integers $p \geq 1$ and $q \geq 1$ such that $y = u^p$ and $y = v^q$.

(4) u and v are each a power of the same primitive word.

Let $u, v \in \Sigma^+$. Suppose there exists a clopen language $L \subseteq \Sigma^*$ with $u \in L$ and $v \notin L$. We note that L^- is also clopen whenever L is, and we call the pair (L, L^-) a *clopen partition separating u and v.*

Theorem 14. *Let $u, v \in \Sigma^+$. There exists a clopen partition separating u and v if and only if u and v do not commute.*

Proof. We handle the forward direction first. Suppose a clopen language L exists with $u \in L$ and $v \notin L$. If u and v commute, then there exists a word x and integers p and q such that $u = x^p$ and $v = x^q$. In particular, this implies that any open set containing u will also contain x, and any open set containing v will also contain x. Then we must have both $x \in L$ (since L is open and contains u) and $x \in L^-$, since L^- is open and contains v. Thus we have a contradiction, and u and v must not commute.

For the reverse direction, we proceed by induction on $|u| + |v|$. We will apply the induction hypothesis on words in various alphabets, so we make no assumption that $|\Sigma|$ is constant.

For our base case, suppose $|u| + |v| = 2$. If u and v do not commute, then they must be distinct words of length 1, and thus the language $\{u\}^+$ is a clopen language separating u from v.

Suppose, as a hypothesis, that for some $k \geq 2$, the result holds for all finite alphabets Σ and for all $u, v \in \Sigma^+$ such that $2 \leq |u| + |v| \leq k$. Now, given any Σ, let $u, v \in \Sigma^+$ be such that u and v do not commute and $|u| + |v| = k + 1$. Let Σ_u and Σ_v, respectively, be the symbols that occur one or more times in u and v. If $\Sigma_u \cap \Sigma_v = \emptyset$, then Σ_u^+ is a clopen language containing u but not v, and our result holds. If not, suppose $a \in \Sigma_u \cap \Sigma_v$. Let $\lambda_u = \frac{|u|_a}{|u|}$ and $\lambda_v = \frac{|v|_a}{|v|}$ be the respective relative frequencies of a in u and v. If $\lambda_u > \lambda_v$, then $\{w \in \Sigma^* : |w|_a \geq \lambda_u |w|\}$ is clopen (by Example 1) and contains u but not v, and we are done. Similarly, if $\lambda_u < \lambda_v$, then $\{w \in \Sigma^* : |w|_a \leq \lambda_u |w|\}$ is a clopen language containing u but not v. Thus it remains to show that the result holds when $\lambda_u = \lambda_v$.

Assume $\lambda_u = \lambda_v = \lambda$. If $\lambda = 1$, then $u = a^i$ and $v = a^j$ for some positive integers i and j, and thus u and v commute, contradicting our original assumption. Hence we must have $0 < \lambda < 1$. Let $n = \frac{|u|}{\gcd(|u|_a, |u|)} = \frac{|v|}{\gcd(|v|_a, |v|)}$ be the denominator of λ when it is expressed in lowest terms. We must have $n > 1$ since λ is not an integer.

Next, we consider a new alphabet Δ with $|\Sigma|^n$ symbols, each corresponding to a word of length n in Σ^*. We consider the bijective morphism ϕ mapping words in Δ^* to words in $(\Sigma^n)^*$ by replacing each symbol in Δ with its corresponding word in Σ^n. Since n divides both $|u|$ and $|v|$, there must then exist unique words $p, q \in \Delta^*$ such that $\phi(p) = u$ and $\phi(q) = v$.

Our plan is now to inductively create a clopen language L over Δ which contains p but not q, and then use this language to construct our clopen partition over Σ separating u and v. We must check that p and q do not commute. If $pq = qp$ then we would have $uv = \phi(p)\phi(q) = \phi(pq) = \phi(qp) = \phi(q)\phi(p) = vu$, since ϕ is a morphism. This is impossible since $uv \neq vu$, so p and q do not commute. We also have $n|p| + n|q| = |u| + |v|$. Since $n > 1$ implies $|p| + |q| < |u| + |v| = k + 1$, the induction hypothesis can be applied to p and q. Thus there exists a clopen language $L \subseteq \Delta^*$ with $p \in L$ and $q \notin L$.

We now construct our clopen partition over Σ separating u and v. We introduce some notation to make this easier. As usual, define $\phi(L) = \{w \in \Sigma^* : w = \phi(r)$ for some $r \in L\}$. Let $A^< = \{w \in \Sigma^* : |w|_a < \lambda |w|\}$ and let $A^= = \{w \in \Sigma^* : |w|_a = \lambda |w|\}$. Additionally, let $A^\leq = A^< \cup A^=$. It is easy to verify that $A^<$, $A^\leq$, and $A^=$ are all closed, and both $A^<$ and $A^\leq$ are open as well. Finally, we let $M = (\phi(L) \cap A^=) \cup A^<$. Since $p \in L$ and $q \notin L$, we must have $u \in \phi(L)$ and $v \notin \phi(L)$. Then since u and v are both contained in $A^=$ but not $A^<$, we must have $u \in M$ and $v \notin M$. We will now finish the proof by showing that M is clopen.

We first show that M is closed. Let $x, y \in M$. We must show that $xy \in M$. There are two cases to consider:

Case (A1): $x, y \in (\phi(L) \cap A^=)$. We see that $\phi(L)\phi(L) = \phi(LL) \subseteq \phi(L)$, so $\phi(L)$ is closed. Then since $A^=$ is closed, $\phi(L) \cap A^=$ is the intersection of two closed languages, and hence closed. Thus $xy \in \phi(L) \cap A^= \subseteq M$.

Case (A2): One or more of x or y is not in $\phi(L) \cap A^=$. Without loss of generality, suppose $x \notin \phi(L) \cap A^=$. Then $x \in A^<$, so $|x|_a < \lambda |x|$. Furthermore, $y \in M \subseteq A^\leq$, so $|y|_a \leq \lambda |y|$. Adding these two inequalities yields $|x|_a + |y|_a < \lambda |x| + \lambda |y|$, so $|xy|_a < \lambda |xy|$ and thus $xy \in A^< \subseteq M$.

Lastly, we show that M is open. Let $z \in M$ and suppose $z = xy$ for some $x, y \in \Sigma^+$. We show that $x \in M$ or $y \in M$. Again, we have two cases to consider:

Case (B1): $z \in A^<$. Since $A^<$ is open, at least one of x or y is in $A^<$. Since $A^< \subseteq M$, we are done.

Case (B2): $z \in \phi(L) \cap A^{=}$. If either x or y is in $A^{<}$, then we are done, so assume otherwise. Then $|x|_a \geq \lambda|x|$ and $|y|_a \geq \lambda|y|$. But $|xy|_a = \lambda|xy|$, so we must have $|x|_a = \lambda|x|$ and $|y|_a = \lambda|y|$ and thus $x, y \in A^{=}$. Then $\lambda|x|$ and $\lambda|y|$ must be integers and hence n divides both $|x|$ and $|y|$. Then there exist $s, t \in \Delta^*$ such that $\phi(s) = x$ and $\phi(t) = y$. But since ϕ is a morphism, we must then have $\phi(st) = \phi(s)\phi(t) = xy = z$. But z in $\phi(L)$, so $st \in L$. Since L is open, we must then have either $s \in L$ or $t \in L$. Thus we must have either $x = \phi(s) \in \phi(L)$ or $y = \phi(t) \in \phi(L)$. Then one of x or y is in $\phi(L) \cap A^{=} \subseteq M$.

Thus M is both closed and open, and the result follows by induction. ■

Corollary 2. *Let $u, v \in \Sigma^+$. There exist non-intersecting finite open languages L and M with $u \in L$ and $v \in M$ if and only if u and v do not commute.*

Proof. As in the proof of Theorem 14, we note that if u and v commute, then there is some x such that $u = x^p$ and $v = x^q$, implying that every open language containing u or v must contain x, and thus there is no open language containing u but not v. If u and v do not commute, then by our theorem, let K be a clopen language containing u but not v. We then take $L = \{w \in K : |w| \leq |u|\}$ and $M = \{w \in K^- : |w| \leq |v|\}$. These are open by our Proposition 5 (b) since K and K^- are both open. ■

We can also use Theorem 14 to extend the topological notion of *connected components* to the setting of formal languages. We say that words $u, v \in \Sigma^+$ are *disconnected* if there exists a clopen partition separating u from v, and *connected* otherwise. We write $u \sim v$ if u and v are connected, and note that $\sim$ is an equivalence relation (indeed, this is the case when we consider the clopen partitions created by any closure operator; it need not be topological). Since Theorem 14 implies that $u \sim v$ if and only if $u = x^p$ and $v = x^q$ for some integers p and q, it follows that each connected component of Σ^+ consists of a primitive word and all of its powers. Connected components of other languages will simply consist of collections of words sharing a common primitive root.

Note that connected components must be closed, but they need not be clopen. In fact, the only clopen components of Σ^+ are the languages $\{a\}^+$ for each $a \in \Sigma$.

The following theorem holds for all closure operators that preserve opennness.

Theorem 15. *If $L, M \subseteq \Sigma^*$ are disjoint and open, then L^+ and M^+ are disjoint.*

Proof. If $L \cap M = \emptyset$, then $M \subseteq L^-$. Then by isotonicity, $M^+ \subseteq L^{-+} = L^-$ since L^- is closed. But then $L \subseteq M^{+-}$. Applying isotonicity again yields $L^+ \subseteq M^{+-+}$. But M^+ is the closure of an open language and is thus clopen, so M^{+-} is also clopen and thus $M^{+-+} = M^{+-}$. Hence $L^+ \subseteq M^{+-}$, and it follows that L^+ and M^+ are disjoint. ■

Corollary 3. *Let $L, M \subseteq \Sigma^*$ be closed and such that $L \cup M = \Sigma^*$. Then $L^{\oplus} \cup M^{\oplus} = \Sigma^*$.*

In our setting, it is not true that a single “point” x and a closed set S can be separated by two open sets. As a counterexample, consider $x = ab$ and $y = \{aa, bb\}^*$. Furthermore, it is not true that that arbitrary disjoint sets, even ones whose closures are disjoint, can be clopen separated. As an example, consider $\{ab\}^*$ and $\{aa, bb\}^*$.

7 Algorithms

We now consider the computational complexity of determining if a given language L is closed or open. Of course, the answer depends on how L is represented.

Theorem 16. *Given an n-state DFA $M = (Q, \Sigma, \delta, q_0, F)$ accepting the regular language L, we can determine in $O(n^2)$ time if L is closed or open.*

Proof. We prove the result when L is positive-closed. For Kleene-closed, we have the additional check $q_0 \in F$. For the open case, we start with a DFA for $\overline{L}$.

It is easy to verify that $L(M)L(M)$ can be accepted by an NFA with $2n$ states, and therefore the language $(L(M))^2 \setminus L(M)$ can be accepted by an NFA with $O(n^2)$ states. For details of the construction, see [3]. ■

From Proposition 5 (a), we know that L is not closed if and only if there exists a word $uv \notin L$ such that $u, v \in L$. We call such a word a *counterexample.*

Corollary 4. *If L is a regular language, accepted by a n-state DFA, that is not closed, then the smallest counterexample is of length $\leq n^2 + n - 1$.*

This $O(n^2)$ upper bound on the length of the shortest counterexample is matched by a corresponding $\Omega(n^2)$ lower bound:

Theorem 17. *There exists a class of DFA's M_n with $2n + 5$ states, having the following property: a shortest word $x \notin L(M_n)$ such that there exist $u, v \in L(M_n)$ with $x = uv$ is of length $n^2 + 2n + 2$.*

Proof. It is easier to describe DFA $M'_n = (Q, \Sigma, \delta, q_0, F)$ that accepts the complement of $L(M_n)$. In other words, we will show that a shortest word $x \in L(M'_n)$ such that there exist $u, v \notin L(M_n)$ with $x = uv$ is of length $n^2 + 2n + 2$. Let $Q = \{q_0, q_1, \ldots, q_n, r, p_0, p_1, \ldots, p_n, s, d\}$, let δ be given by Table 3, and let $F = \{q_0, q_1, \ldots, q_n, p_0, p_1, \ldots, p_n, s\}$. The case $n = 5$ is shown in Fig. 2.

First, we observe that $x = 10^{n-1}110^{n^2+n-1}1$ is accepted by M'_n, but neither $u = 10^{n-1}1$ nor $v = 10^{n^2+n-1}1$ is. Next, take any word x' accepted by M'_n. If the acceptance path does not pass through r, then by examining the DFA we see

Table 3. Transition function $\delta(q, a)$ of M'_n

$a\backslash q$	q_0	q_1	q_2	$\ldots$	q_{n-1}	q_n	r	p_0	p_1	$\ldots$	p_{n-1}	p_n	s	d
0	d	q_2	q_3	$\ldots$	q_n	q_1	d	p_1	p_2	$\ldots$	p_n	p_0	d	d
1	q_1	s	s	$\ldots$	s	r	p_0	d	d	$\ldots$	d	s	d	d

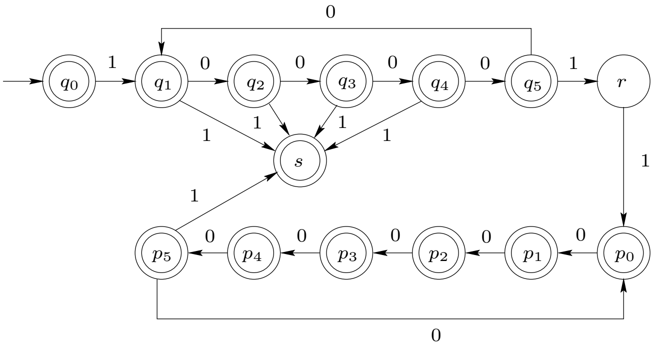


Fig. 2. Example of DFA M_n for $n = 5$. Unspecified transitions go to the dead state d.

that every prefix of x' is also accepted. Otherwise, the acceptance path passes through r. Again, we see that every prefix of x' is accepted, with the possible exception of the prefix ending at r. Thus either x' is of the form $10^{in+n-1}110^k$ for some $i, k \geq 0$, or x' is of the form $10^{in+n-1}110^{j(n+1)+n}1$ for some $i, j \geq 0$. In both cases the prefix ending at r is $10^{in+n-1}1$, so in the first case, the corresponding suffix is 10^k for some $k \geq 0$, and this suffix is accepted by M_n'. In the latter case the corresponding suffix is $10^{j(n+1)+n}1$. This is accepted unless $j(n+1)+n$ is of the form $in+n-1$. If $in+n-1 = j(n+1)+n$, then by taking both sides modulo n, we see that $j \equiv -1 \pmod{n}$. Thus $j \geq n-1$. Thus $|x'| \geq 1+n-1+1+1+(n-1)(n+1)+n+1 = n^2+2n+2$. ■

We now turn to the case where M is represented as an NFA or regular expression. For the following theorem, we actually require the word w exhibited in the theorem above to have length ≥ 2. However, this can easily be accomplished via a trivial modification of the proof given in [1], since the word w encodes a configuration of the Turing machine T.

Theorem 18. *The following problem is PSPACE-complete: given an NFA M, decide if $L(M)$ is closed.*

Proof. First, we observe that the problem is in PSPACE. We give a nondeterministic polynomial-space algorithm to decide if $L(M)$ is not closed, and use Savitch's theorem to conclude the result.

If M has n states, then there is an equivalent DFA M' with $N \leq 2^n$ states. From Corollary 4 we know that if $L = L(M) = L(M')$ is not closed, then there exist words u, v with $u, v \in L$ but $uv \notin L$, and $|uv| \leq N^2 + N - 1 = 2^{2n} + 2^n - 1$. We now guess u, processing it symbol-by-symbol, arriving in a set of states S of M. Next, we guess v, processing it symbol-by-symbol starting from both q_0 and S, respectively and ending in sets of states T and U. If U contains a state of F and T does not, then we have found $u, v \in L$ such that $uv \notin L$. While we guess u and v, we count the number of symbols guessed, and reject if that number is greater than $2^{2n} + 2^n - 1$.

To show that the problem is PSPACE-hard, we note that Δ^* is closed, but $\Delta^* \setminus \{w\}$ for w with $|w| \geq 2$ is not. With the aid of Lemma 10.2 of [1] we could use an algorithm solving the problem of whether a language is closed to solve the membership problem for polynomial-space bounded Turing machines. ■

If L is not closed and is accepted by an n-state NFA, then a minimal-length word uv, with $u, v \in L$ but $uv \notin L$, may be exponentially long. Such an example is given in [6], where it is shown that for some constant c, there exist NFA's with n states such that a shortest word not accepted is of length $> 2^{cn}$. We note also that the problem of deciding, for a given NFA M, whether $L(M)$ is open is PSPACE-complete. The proof is similar to that of Theorem 18.

Acknowledgments. This research was supported by the Natural Sciences and Engineering Research Council of Canada. We thank the anonymous referees for suggesting ways to shorten some of our proofs.

References

1. Aho, A., Hopcroft, J., Ullman, J.: The Design and Analysis of Computer Algorithms. Addison-Wesley, Reading (1974)
2. Brzozowski, J., Grant, E., Shallit, J.: Closures in formal languages and Kuratowski's theorem (preprint) (January 2009), http://arxiv.org/abs/0901.3761
3. Brzozowski, J., Grant, E., Shallit, J.: Closures in formal languages: concatenation, separation, and algorithms (January 2009), http://arxiv.org/abs/0901.3763
4. Burris, S.N., Sankappanavar, H.P.: A Course in Universal Algebra, 2nd edn., http://www.math.uwaterloo.ca/snburris/htdocs/ualg.html
5. Chagrov, A.V.: Kuratowski numbers. In: Application of Functional Analysis in Approximation Theory, Kalinin. Gos. Univ., Kalinin, pp. 186–190 (1982) (in Russian)
6. Ellul, K., Krawetz, B., Shallit, J., Wang, M.-w.: Regular expressions: new results and open problems. J. Autom. Lang. Combin. 10, 407–437 (2005)
7. Fife, J.H.: The Kuratowski closure-complement problem. Math. Mag. 64, 180–182 (1991)
8. Gardner, B.J., Jackson, M.: The Kuratowski closure-complement theorem. New Zealand J. Math. (to appear); Preprint available at http://www.latrobe.edu.au/mathstats/department/algebra-research-group/Papers/GJ_Kuratowski.pdf
9. Graham, R.L., Knuth, D.E., Motzkin, T.S.: Complements and transitive closures. Discrete Math. 2, 17–29 (1972)
10. Hammer, P.C.: Kuratowski's closure theorem. Nieuw Archief v. Wiskunde 7, 74–80 (1960)
11. Kuratowski, C.: Sur l'opération $\overline{A}$ de l'analysis situs. Fund. Math. 3, 182–199 (1922)
12. Lyndon, R.C., Schützenberger, M.P.: The equation $\mathrm{a}^{\mathrm{M}} = \mathrm{b}^{\mathrm{N}}\ \mathrm{c}^{\mathrm{P}}$ in a free group. Michigan Math. J. 9, 289–298 (1962)
13. Peleg, D.: A generalized closure and complement phenomenon. Discrete Math. 50, 285–293 (1984)

Rich and Periodic-Like Words*

Michelangelo Bucci, Aldo de Luca, and Alessandro De Luca

Dipartimento di Matematica e Applicazioni "R. Caccioppoli"
Università degli Studi di Napoli Federico II, via Cintia, Monte S. Angelo
80126 Napoli, Italy
{micbucci,aldo.deluca,alessandro.deluca}@unina.it

Abstract. In this paper we investigate the periodic structure of *rich words* (i.e., words having the highest possible number of palindromic factors), giving new results relating them with *periodic-like* words. In particular, some new characterizations of rich words and rich palindromes are given. We also prove that a periodic-like word is rich if and only if the square of its fractional root is also rich.

1 Introduction

In the study of the structural properties of finite or infinite words, a relevant role is played by palindromes, i.e., words which can be read without distinction from left to right or from right to left. Indeed, many important classes of words enjoy remarkable properties regarding their palindromic factors.

A well-known example was given in [1]: any factor w of an episturmian word has the maximum number of distinct palindromic factors (that is, $|w|+1$ counting the empty word). Recently, such a property was also found in different contexts (cf. [2,3]), so in [4] a more systematic study of *rich words* (words w having $|w|+1$ distinct palindromic factors) was initiated; more recent results on rich words can be found in [5,6,7].

In this paper we study rich words in relation with other structural properties of finite words, introduced by Carpi and the second author in [8,9] in the frame of a suitable classification of words with respect to their periods.

In the next section we recall some basic definitions and notation. In Sect. 3 we deal with periodic-like words and prove some new results which are useful for the sequel. In Sect. 4 we shall review some basic results on rich words. In the last section we give our main results relating periodic-like and rich words. In particular, we give a new characterization of rich palindromes, and we prove that a palindrome (or a periodic-like word) is rich if and only if the square of its fractional root is also rich.

* The work for this paper was partially supported by the Italian Ministry of Education (MIUR) under the project PRIN 2007 "Proprietà combinatorie e strutturali nella teoria dei linguaggi formali".

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 145–155, 2009.

2 Preliminaries

In the following, A will denote a finite *alphabet* and A^* the *free monoid* of words over A with the natural concatenation operation. The identity of A^* is the *empty word* ε. Any other word w of A^* can be uniquely written as $w = a_1 \cdots a_n$ with $a_i \in A$ for $i = 1, \ldots, n$. The integer n, denoted by $|w|$, is called the *length* of w; naturally one sets $|\varepsilon| = 0$.

A *factor* of $w \in A^*$ is any word u such that $w = rus$ for some $r, s \in A^*$. If $r = \varepsilon$ (resp. $s = \varepsilon$) then u is a *prefix* (resp. *suffix*) of w. If $|r| = |s|$, then u is a *median* factor of w.

Two words $u, v \in A^*$ are *conjugate* if there exist $\alpha, \beta \in A^*$ such that $u = \alpha\beta$ and $v = \beta\alpha$. It is easy to see that u is a conjugate of v if and only if $|u| = |v|$ and u is a factor of v^2.

The *reversal* of a word $w = a_1 \cdots a_n$, with $a_i \in A$ for $1 \leq i \leq n$, is the word $\tilde{w} = a_n \cdots a_1$; one assumes $\tilde{\varepsilon} = \varepsilon$. If $w = \tilde{w}$, then w is a *palindrome*. The set of all palindromes of A^* will be denoted by PAL. For any $w \in A^*$, $w^{(+)}$ denotes the *right palindromic closure* of w, that is, the shortest palindrome having w as a prefix. In a symmetric way, $w^{(-)}$ is the shortest palindrome having w as a suffix.

Let $w = a_1 a_2 \cdots a_n$ be a non-empty word with $a_1, \ldots, a_n \in A$. A *period* of w is any positive integer p such that for all $i, j = 1, \ldots, n$,

$$i \equiv j \pmod{p} \Longrightarrow a_i = a_j \ .$$

The minimal period of w is denoted by π_w; the word w is called *periodic* if $2\pi_w \leq |w|$. The *fractional root* of w is its prefix z_w of length π_w. For technical convenience, we also set $\pi_\varepsilon = 1$ and $z_\varepsilon = \varepsilon$. A word w is called *unbordered* if $w = z_w$.

A right-infinite word over the alphabet A, called *infinite word* for short, is a mapping $x : \mathbb{N}_+ \to A$, where $\mathbb{N}_+$ is the set of positive integers. One can represent x as

$$x = x_1 x_2 \cdots x_n \cdots \ ,$$

where for any $i > 0$, $x_i = x(i) \in A$. A (finite) *factor* of x is either the empty word or any sequence $u = x_i \cdots x_j$ with $i \leq j$, i.e., any block of consecutive letters of x. If $i = 1$, then u is a *prefix* of x. An infinite word x such that $x = uuu \cdots = u^\omega$ for some $u \in A^*$ is called *periodic*. The set of all infinite words over A is denoted by A^ω. We also set $A^\infty = A^* \cup A^\omega$.

Let $w \in A^\infty$. We denote respectively by $\mathrm{Fact}(w)$ and $\mathrm{Pref}(w)$ the sets of factors and prefixes of w. A factor u of w is called *right special* (resp. *left special*) if there exist two letters $x, y \in A$, $x \neq y$, such that $ux, uy \in \mathrm{Fact}(w)$ (resp. $xu, yu \in \mathrm{Fact}(w)$). Any pair $(\lambda, \mu) \in A^* \times A^\infty$ such that $w = \lambda u \mu$ is called an *occurrence* of u in w. An occurrence is *internal* if λ and μ are both non-empty. The factor u is *unioccurrent* (or *unrepeated*) in w if u has exactly one occurrence in w.

An infinite word $x \in A^\omega$ is called *episturmian* (see [1]) if $\mathrm{Fact}(x)$ is closed under reversal and x has at most one right special factor of each length. A

Sturmian word can be defined as an aperiodic episturmian word on a binary alphabet.

For all definitions and notation not explicitly given in the text, the reader is referred to [10,11].

3 Periodic-Like Words

Let w be a non-empty word of A^*. Following the notation in [8], we denote by h'_w (resp. k'_w) the longest repeated prefix (resp. suffix) of w. The word w is called *periodic-like* if h'_w is not right special in w. Conventionally, the empty word ε is also considered to be periodic-like. We denote by H_w (resp. K_w) the length of the shortest unrepeated prefix (resp. suffix) of w. The following [8] holds:

Lemma 3.1. *Let w be a non-empty periodic-like word. Then*

1. *h'_w has no internal occurrence in w,*
2. *$h'_w = k'_w$,*
3. *k'_w is not left special in w,*
4. *$\pi_w = |w| - H_w + 1 = |w| - K_w + 1$,*
5. *$w = z_w h'_w$.*

Moreover, one can prove [8] that condition 1, as well as condition 3, is equivalent to the condition of being periodic-like. If a palindrome w is periodic-like, then, by definition, h'_w is a palindrome.

Let u be a non-empty factor of a finite or infinite word w. We recall (see for instance [4]) that a *complete return* to u in w is any factor of w having exactly two occurrences of u, one as a prefix and one as a suffix. Trivially, one has that a factor v of w is a complete return to u if and only if v is a periodic-like word having $h'_v = u$.

Example 3.2. Let $w = aabcaa$. The longest repeated prefix $h'_w = aa$ is not right special, so that w is periodic-like. We have $k'_w = aa = h'_w$, $H_w = K_w = 3$, $\pi_w = 4$, and $z_w = aabc$.

For any non-empty word w, w^f (resp. w^ℓ) denotes the first (resp. last) letter of w.

Proposition 3.3. *If w is periodic-like and its longest proper median factor is a palindrome, then w is a palindrome.*

Proof. Suppose that $w = xuy$ with $u \in PAL$ and $x, y \in A$. Since w is periodic-like, h'_w is a prefix and a suffix of w, so that it begins with the letter x and terminates with the letter y. We can write $h'_w = x\beta$ where β is a prefix of the palindrome u. Therefore,

$$h'_w = x\beta = \tilde{\beta}y \ .$$

If $\beta = \varepsilon$, then $x = y$. If β is non-empty, then $\beta^\ell = y$ and $\tilde{\beta}^f = x$. Since $\beta^\ell = \tilde{\beta}^f$, one derives again $y = x$. Hence, w is a palindrome. □

We say that a word w is *strongly periodic-like* if all of its median factors are periodic-like.

Example 3.4. The word $w = aabbaa$ is strongly periodic-like. Indeed, all of its median factors are periodic-like. On the contrary, the word $aabcaa$ considered in Example 3.2 is periodic-like but not strongly, since its median factor bc is not periodic-like. Finally, the palindrome $abcbaccabcba$ is periodic-like but not strongly, as its median factor $v = cbaccabc$ has $h'_v = c$ which is right special in v, so that v is not periodic-like.

Proposition 3.5. *A strongly periodic-like word is a palindrome.*

Proof. Let w be a strongly periodic-like word. The proof is by induction on the length of w. The result is trivial when $|w| \leq 3$. Indeed, the only periodic-like words are the palindromes ε, x, xx, xyx with $x, y \in A$. Let us then write w as:

$$w = xuy$$

with $x, y \in A$. Since w is strongly periodic-like, u is strongly periodic-like and $|u| < |w|$. By induction $u \in PAL$. By Proposition 3.3 one derives that w is a palindrome. □

For a word $w \in A^*$, we denote by R_w the minimal integer p such that w has no right special factor of length p. We recall [12,13] that a word w is *trapezoidal* if $|w| = R_w + K_w$. A finite word is called *Sturmian* if it is a finite factor of a (standard) Sturmian word (cf. [11]). Notice that any finite Sturmian word is trapezoidal, but not conversely. For example, the word $aabb$ is trapezoidal, but not Sturmian.

Proposition 3.6. *A periodic-like trapezoidal word is Sturmian.*

Proof. If w is periodic-like, then by Lemma 3.1, $\pi_w = |w| - K_w + 1$. Since w is trapezoidal, one has $|w| = R_w + K_w$, so that $\pi_w = R_w + 1$. This implies that w is Sturmian (see [14, Proposition 28]). □

We observe that the converse of the preceding proposition does not hold. Indeed, even though every finite Sturmian word is trapezoidal, not every finite Sturmian word is periodic-like. For instance, $w = aab$ is Sturmian (and hence trapezoidal), but not periodic-like (since $h'_w = a$ is right special).

4 Rich Words

Let $w \in A^*$ and denote by S_w the number of palindromic factors of w (including the empty word). As proved in [1], $S_w \leq |w| + 1$. A word w is said *rich* if $S_w = |w| + 1$. We recall (cf. [4]) that the richness property is closed by factors, as well as under the operations of reversal and palindromic closures.

An infinite word is called rich if all its factors are rich. Sturmian and episturmian words (see, for instance, [15] for an overview), are well-studied families of infinite rich words; however, the richness property has been found in wider contexts (cf. [2,3]). A general investigation on rich words was recently carried on in [4,5,6,7].

The following characterization of rich words was first established in [1]:

Proposition 4.1. *A word $w \in A^\infty$ is rich if and only if every prefix p of w has a palindromic suffix which is unioccurrent in p.*

A characterization of rich words in terms of complete returns to their palindromic factors was given in [4, Theorem 2.14]:

Theorem 4.2. *A word $w \in A^\infty$ is rich if and only if for each palindromic factor u of w, every complete return to u in w is a palindrome.*

As a consequence, we can easily derive the following:

Proposition 4.3. *Let $w \in A^\infty$. The following conditions are equivalent:*

1. *w is rich.*
2. *For any factor v of w the longest palindromic prefix (or suffix) of v is unrepeated in v.*
3. *For any periodic-like factor v of w the longest palindromic prefix (or suffix) of v is unrepeated in v.*

Proof. 1) ⇒ 2). Since w and its factor v are rich, if the longest palindromic prefix (or suffix) α of v is repeated in v, then by Theorem 4.2 the complete return to α in v is a palindrome. This contradicts the maximality of the length of α.

2) ⇒ 3). Trivial.

3) ⇒ 1). In view of Theorem 4.2, it is sufficient to show that any complete return v to a palindrome in w is a palindrome. Such a v is then a periodic-like factor of w with $h'_v \in PAL$. Since the longest palindromic prefix (or suffix) α of v is unrepeated in v it follows that $|\alpha| > |h'_v|$. If $|\alpha| < |v|$, since h'_v is a palindromic suffix (or prefix) of α, one would derive that h'_v has an internal occurrence in v which is a contradiction. Hence, $\alpha = v$, so that v is palindrome. □

The following proposition summarizes two further useful results proved in [5].

Proposition 4.4. *Let $w \in A^\infty$ be a rich word. For any $v \in \mathrm{Fact}(w)$, the following holds:*

1. *Any factor of w beginning with v and ending with $\tilde{v}$, having no internal occurrences of v nor of $\tilde{v}$, is a palindrome.*
2. *If $v \notin PAL$, then $\tilde{v}$ is a unioccurrent factor of any complete return to v in w.*

5 Main Results

Let us observe that in general a periodic-like word is not rich. For instance, the word *aabcaa* is periodic-like but not rich. Conversely, there are rich words such as *abc* and *aabaaca* which are not periodic-like. However, in the palindromic case we have the following:

Theorem 5.1. *A word is a rich palindrome if and only if it is strongly periodic-like.*

Proof. Let w be a rich palindrome. The longest proper palindromic prefix α of w occurs only as a prefix and as a suffix of w. In fact, if α had an internal occurrence in w, there would exist a complete return to α, which would be a palindrome in view of Theorem 4.2, and thus a proper palindromic prefix of w longer than α, contradicting the maximality of $|\alpha|$. Thus $|\alpha| \leq |h'_w|$ so that h'_w has no internal occurrence in w, that is w is periodic-like (and moreover $h'_w = \alpha$). Since all median factors of w are also rich palindromes, the "only if" part follows.

Conversely, suppose that w is strongly periodic-like. By Proposition 3.5, w is a palindrome. If $w = \varepsilon$, then it is clearly rich, so let us suppose by induction that $w = aua$ with $a \in A$ and u a rich palindrome. Since $|u| = |w| - 2$ and $S_u = |u| + 1$, it suffices to show that h'_w is the only proper palindromic factor of w which does not occur in u. Indeed, since w is a periodic-like palindrome, h'_w is a palindrome, and in fact the longest proper palindromic prefix of w, as h'_w cannot occur in u by Lemma 3.1. Any palindromic factor v of w with $|v| < |h'_w|$ must be a factor of u. This is trivial if v is not a prefix of w. If v is a prefix of w, then v is a prefix and also a suffix of h'_w and hence a factor of u. □

Let us observe that a different characterization of rich palindromes in terms of *palindromic* and *factor complexity* was recently obtained in [6].

A rich word which is periodic-like (but not strongly) need not be a palindrome: for instance, the word *abacdcabac* is rich and periodic-like.

Corollary 5.2. *A palindrome is rich if and only if all its palindromic factors are periodic-like.*

Proof. If a palindrome is rich, then all its palindromic factors are rich. By Theorem 5.1 it follows that they are periodic-like. Conversely, if all palindromic factors of w are periodic-like, then all median factors of w will be periodic-like so that w is strongly periodic-like and by Theorem 5.1, w is a rich palindrome. □

We remark that there exist non-palindromic words whose palindromic factors are all rich (and then periodic-like), but are not themselves rich. This is the case, for instance, of the word $w = abcab$.

Corollary 5.3. *All palindromic factors of an episturmian word are periodic-like.*

Proof. Any factor of an episturmian word is rich (cf. [1]), so that the result follows from Theorem 5.1. □

Remark 5.4. From the preceding corollary one has in particular that all palindromic factors of Sturmian words are periodic-like. Moreover, in the case of *central* Sturmian words, i.e., the palindromic prefixes w of all standard Sturmian words, one has that these words are *semiperiodic* [12], i.e., $R_w < H_w$. We recall [9] that a semiperiodic word is periodic-like, whereas the converse is not in general true.

Remark 5.5. From the previous results one easily obtains that a trapezoidal palindrome is Sturmian [6]. Indeed, any trapezoidal word is rich [6] so that the result follows by Corollary 5.2 and Proposition 3.6.

Remark 5.6. Notice that the converse of Corollary 5.3 does not hold: indeed there exist periodic-like finite Sturmian and episturmian words that are not palindromes such as *abab* and *abacab*.

The following proposition holds:

Proposition 5.7. *A word $w \in A^\infty$ is rich if and only if every periodic-like factor v of w with $h'_v \in PAL$ is strongly periodic-like.*

Proof. If w is rich, then by Theorem 4.2 every periodic-like factor v of w with $h'_v \in PAL$ is a palindrome. Since v is rich, by Theorem 5.1 it is strongly periodic-like. Conversely, suppose that every periodic-like factor v of w with $h'_v \in PAL$ is strongly periodic-like. By Proposition 3.5, v is a palindrome and the result follows by Theorem 4.2. □

Proposition 5.8. *A word $w \in A^*$ is rich if and only if all palindromic factors of $w^{(+)}$ are periodic-like.*

Proof. If w is rich, then by [4, Proposition 2.6] the right palindrome closure $w^{(+)}$ is rich. By Corollary 5.2, all palindromes in $w^{(+)}$ are periodic-like. Conversely, if all palindromic factors of $w^{(+)}$ are periodic-like, one has by Corollary 5.2 that $w^{(+)}$ is rich. Since $w^{(+)}$ begins with w and the richness property is closed by factors one derives that w is rich. □

Corollary 5.9. *A word $w \in A^*$ is rich if and only if all palindromic factors of $w^{(-)}$ are periodic-like.*

Proof. It is sufficient to observe that $w^{(-)} = \tilde{w}^{(+)}$ and that the richness property is closed under reversal. □

We recall (cf. [16]) that a word w is called *symmetric* if $w \in PAL^2$. The following result was proved in [4, Theorem 3.1].

Proposition 5.10. *Let $w \in A^*$. Then the following conditions are equivalent:*

1. *w^2 is rich,*
2. *w^ω is rich,*
3. *w is symmetric and all of its conjugates are rich.*

A useful corollary of this proposition is:

Corollary 5.11. *Let $w \in A^*$ and let z_w be its fractional root. If z_w^2 is rich, then w is rich.*

Proof. If z_w^2 is rich, then by Proposition 5.10 the infinite word z_w^ω is also rich, so that its prefix w is rich as well. □

It is worth noting that if the fractional root of a word w is rich, in general w need not be rich itself. For example, the word $w = abca$ is not rich even if $z_w = abc$ is rich.

As an easy consequence of Corollary 5.11 we obtain:

Proposition 5.12. *Let w be a finite periodic word. Then w is rich if and only if z_w^2 is rich.*

Proof. Suppose w is rich. Then z_w^2 is rich, since it is a prefix of w by the definition of periodic word. The converse follows from Corollary 5.11. □

Theorem 5.13. *Let w be a palindrome. Then w is rich if and only if z_w^2 is rich.*

Proof. Since w is a rich palindrome, by Theorem 5.1 it is periodic-like, so that by Lemma 3.1 we can write:

$$w = z_w h'_w = h'_w \tilde{z}_w \ ,$$

where h'_w is the longest repeated prefix of w. Therefore, from the preceding equation, by the classic lemma of Lyndon and Schützenberger (cf. [10]) there exist words α and β and $n \geq 0$ such that:

$$z_w = \alpha\beta, \ \tilde{z}_w = \beta\alpha, \ h'_w = (\alpha\beta)^n\alpha \ .$$

Hence, as $w \in PAL$

$$w = (\alpha\beta)^{n+1}\alpha = (\tilde{\alpha}\tilde{\beta})^{n+1}\tilde{\alpha} \ .$$

Since $n \geq 0$ one has that $\alpha, \beta \in PAL$.

If $n > 0$, then z_w^2 is a prefix of w, so that it is rich. If $n = 0$, then $h'_w = \alpha$ and

$$w = h'_w \beta h'_w \ .$$

We will show that any prefix p of z_w^2 has a unioccurrent palindromic suffix. This is certainly true for $|p| \leq |w|$ by Proposition 4.1, since w is rich.

Suppose $|p| > |w|$. We can write $p = w\delta = h'_w\beta h'_w\delta$, with $\delta \in \text{Pref}\,(\beta)$. Since $\beta, h'_w \in PAL$, the prefix p has the palindromic suffix $\tilde{\delta} h'_w \delta$. In fact, this suffix is unioccurrent in p, since otherwise h'_w would have an internal occurrence in w, which is absurd in view of Lemma 3.1. Thus z_w^2 is rich by Proposition 4.1.

The converse follows from Corollary 5.11. □

We remark that if w is a non-palindromic rich word, in general z_w^2 may be not rich, as in the case of $w = abc = z_w$, which is rich whereas $z_w^2 = abcabc$ is not.

Corollary 5.14. *A word $w \in A^*$ is rich if and only if $z_{w^{(+)}}^2$ is rich.*

Proof. The proof is an immediate consequence of the preceding theorem together with the fact that a word w is rich if and only if $w^{(+)}$ is rich. □

We recall [16] that if a word w is a palindrome, then its fractional root z_w is symmetric. Moreover, $z_w = z_{w^{(+)}}$ if and only if z_w is symmetric. Thus a more general result than Theorem 5.13 is the following:

Theorem 5.15. *Let w be a word such that z_w is symmetric. Then w is rich if and only if z_w^2 is rich.*

Proof. It is enough to observe that $z_w \in PAL^2$ implies $z_w = z_{w^{(+)}}$, so that $z_w^2 = z_{w^{(+)}}^2$ and the result follows from Corollary 5.14. □

Example 5.16. Consider the rich word $w = abacdcabac \notin PAL$, whose root $z_w = abacdc \in PAL^2$. One easily verifies that z_w^2 is rich.

Corollary 5.17. *Let w be an unbordered symmetric word. Then w is rich if and only if w^2 is rich.*

Proof. Since w is unbordered and symmetric, we have $w = z_w \in PAL^2$. As $w^2 = z_w^2$, the result follows from Theorem 5.15. □

We remark that from the preceding corollary and Proposition 5.10, one has that if w is a symmetric and unbordered rich word, then all conjugates of w are rich.

Theorem 5.18. *Let $w \in A^*$ be a periodic-like word. Then w is rich if and only if z_w^2 is rich.*

Proof. If w is periodic and rich, then z_w^2 is rich by Proposition 5.12. Let us then assume that w is not periodic. Since w is periodic-like and $2\pi_w > |w|$, we can write

$$w = h'_w u h'_w$$

for some $u \neq \varepsilon$. Let us set $u = xu'$ with $x \in A$ and $u' \in A^*$.

We prove that wx is periodic-like and rich, with $z_{wx} = z_w$. We can write

$$wx = h'_w x u' h'_w x \ .$$

We have that $h'_{wx} = h'_w x$; indeed if $|h'_{wx}| > |h'_w x|$, then h'_w would have an internal occurrence in w, which is absurd since w is periodic-like. By Lemma 3.1 it follows that $z_{wx} = h'_w u = z_w$.

If $h'_w \in PAL$, then since $w = h'_w x u' h'_w$ is rich and is a complete return to h'_w, by Theorem 4.2 we obtain $w \in PAL$. This implies, by Theorem 5.13, that z_w^2 is rich, so that its prefix wx is also rich.

Now let us suppose $h'_w \notin PAL$. By Proposition 4.4, since w is rich and is a complete return to h'_w, the word $\tilde{h}'_w$ is an internal unioccurrent factor of w. Hence w has a prefix γ which begins with $h'_w x$ and ends with $\tilde{h}'_w$, and has no other occurrences of h'_w or $\tilde{h}'_w$. By Proposition 4.4, γ is a palindrome, so that it ends with $x\tilde{h}'_w$. We can then write

$$w = \xi x \tilde{h}'_w \xi'$$

for some $\xi, \xi' \in A^*$. The suffix $\tilde{h}'_w \xi'$ ends with h'_w and has no internal occurrences of h'_w nor of $\tilde{h}'_w$, so that by Proposition 4.4 it is a palindrome. Thus the word wx has the palindromic suffix

$$x\tilde{h}'_w \xi' x = x\tilde{\xi}' h'_w x \ ,$$

which is unioccurrent because otherwise h'_w would have an internal occurrence in w. Since w is rich, by Proposition 4.1 every prefix of w has a unioccurrent palindromic suffix; hence all prefixes of wx have a unioccurrent palindromic suffix, so that wx is rich by Proposition 4.1.

We have proved that wx is periodic-like and rich, with $z_{wx} = z_w$. By iterating this argument, we eventually obtain that $z_w^2 = wu$ is rich.

The converse follows from Corollary 5.11. □

Remark 5.19. Since a rich palindrome is periodic-like by Theorem 5.1, the preceding theorem is an extension of Theorem 5.13. However, Theorem 5.13 is used in the proof of Theorem 5.18.

As a consequence of Theorem 5.18 and of Proposition 5.10, one has that if w is a periodic-like rich word, then its fractional root z_w is symmetric.

In conclusion, we mention that the importance of considering rich periodic-like words is also due to the fact that, as proved in [8], any word w can be canonically decomposed in (overlapping) periodic-like factors $w_1, \ldots, w_n$, with the property that

$$\pi_w = \sum_{i=1}^{n} \pi_{w_i} \quad . \tag{1}$$

Thus any rich word w can be canonically decomposed in rich periodic-like factors w_i $(i = 1, \ldots, n)$ having symmetric roots and satisfying (1).

Example 5.20. Let w be the rich word *abaccabacabaadaab*. Following [8], the canonical decomposition of w in rich periodic-like words is given by (w_1, w_2, w_3) with

$$w_1 = abaccabac\,, \quad w_2 = abacaba\,, \quad w_3 = abaadaab \ ,$$

whose minimal periods are respectively 5, 4, and 6. We have $\pi_w = 5+4+6 = 15$, and the roots $z_{w_1} = abacc$, $z_{w_2} = abac$, and $z_{w_3} = abaada$ are all symmetric.

References

1. Droubay, X., Justin, J., Pirillo, G.: Episturmian words and some constructions of de Luca and Rauzy. Theoretical Computer Science 255, 539–553 (2001)
2. Allouche, J.P., Baake, M., Cassaigne, J., Damanik, D.: Palindrome complexity. Theoretical Computer Science 292, 9–31 (2003)
3. Ambrož, P., Frougny, C., Masáková, Z., Pelantová, E.: Palindromic complexity of infinite words associated with simple Parry numbers. Ann. Inst. Fourier (Grenoble) 56, 2131–2160 (2006)
4. Glen, A., Justin, J., Widmer, S., Zamboni, L.Q.: Palindromic richness. European Journal of Combinatorics 30, 510–531 (2009)
5. Bucci, M., De Luca, A., Glen, A., Zamboni, L.Q.: A connection between palindromic and factor complexity using return words. Advances in Applied Mathematics 42, 60–74 (2009)
6. de Luca, A., Glen, A., Zamboni, L.Q.: Rich, Sturmian, and trapezoidal words. Theoretical Computer Science 407, 569–573 (2008)

7. Bucci, M., De Luca, A., Glen, A., Zamboni, L.Q.: A new characteristic property of rich words. In: Theoretical Computer Science (to appear, 2009) doi:10.1016/j.tcs.2008.11.001
8. Carpi, A., de Luca, A.: Periodic-like words, periodicity, and boxes. Acta Informatica 37, 597–618 (2001)
9. Carpi, A., de Luca, A.: Semiperiodic words and root-conjugacy. Theoretical Computer Science 292, 111–130 (2003)
10. Lothaire, M.: Combinatorics on Words. Addison-Wesley, Reading (1983); Reprinted by Cambridge University Press, Cambridge UK (1997)
11. Lothaire, M.: Algebraic Combinatorics on Words. Cambridge University Press, Cambridge (2002)
12. de Luca, A.: On the combinatorics of finite words. Theoretical Computer Science 218, 13–39 (1999)
13. D'Alessandro, F.: A combinatorial problem on trapezoidal words. Theoretical Computer Science 273, 11–33 (2002)
14. de Luca, A., De Luca, A.: Combinatorial properties of Sturmian palindromes. International Journal of Foundations of Computer Science 17, 557–574 (2006)
15. Berstel, J.: Sturmian and Episturmian Words (a survey of some recent results). In: Bozapalidis, S., Rahonis, G. (eds.) CAI 2007. LNCS, vol. 4728, pp. 23–47. Springer, Heidelberg (2007)
16. de Luca, A., De Luca, A.: Pseudopalindrome closure operators in free monoids. Theoretical Computer Science 362, 282–300 (2006)

Traces of Control-Flow Graphs[⋆]

Simone Campanoni[⋆⋆] and Stefano Crespi Reghizzi[⋆⋆⋆]

Politecnico di Milano, Dipartimento Elettronica e Informazione - DEI
P.za Leonardo da Vinci 32, I–20133, Milano
simone.campanoni@mail.polimi.it, stefano.crespireghizzi@polimi.it

Abstract. This is a new applied development of trace theory to compilation. Trace theory allows to commute independent program instructions, but overlooks the differences between control and data dependencies. Control(C)-dependences, unlike data-dependences, are determined by the Control Flow Graph, modelled as a local DFA. To ensure semantic equivalence, partial commutation must preserve C-dependences. New properties are proved for C-dependences and corresponding traces. Any local language is star-connected with respect to C-dependences, hence this trace language family is recognizable. Local languages unambiguously represent traces. Within the family of local languages with the same C-dependences, we construct the language such that instructions are maximally anticipated. This language differs from the Foata-Cartier normal form. Future directions for application of trace theory to program optimization are outlined.

1 Introduction

This is a research on the application of trace theory to compilation. Optimizing compilers transform a program in many ways. Transformations which only change the order of execution of instructions are known as rescheduling. Inside compilers, the Control Flow Graph CFG representation carries the essential information needed for performing program transformations. CFG nodes represent the instructions (such as assignments and predicates), and the arcs the predecessor-successor relation. A CFG is conveniently viewed as a Deterministic Finite Automaton (DFA) of the local [3] type. Program instructions interfere in different ways, termed *dependences.*

> The set of dependences for a program may be viewed as inducing a partial ordering on the statements and predicates in the program that must be followed to preserve the semantics of the original program. Dependences arise as the result of two separate effects. First, a [Data-]dependence exists between two statements whenever a variable

[⋆] Preliminary presentation at ESF-AutoMathA Workshop on Developments and New Tracks in Trace Theory, Cremona 2008.

[⋆⋆] Partially supported by STMicroelectronics.

[⋆⋆⋆] Partially supported by COFIN “Linguaggi Formali” and CNR-IEIIT.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 156–169, 2009.

> appearing in one statement may have an incorrect value if the two statements are reversed. . . . Second, a [Control(C)-]dependence exists between a statement and the predicate whose value immediately controls the execution of the statement [7].

Mazurkiewicz's theory quite abstractly represents the program statements as letters of an alphabet, and D-dependences by means of a binary relation. This abstraction is realistic because a data-dependence, say of the Read-After-Write type, is solely determined by the variables used or defined in the two statements, and not by their position in the CFG. This allows to find a program (i.e. a model in the logical sense) for any arbitrarily given data-dependence relation over the alphabet.

On the other hand a C-dependence is a structural property of the program CFG. If C-dependences are arbitrarily assigned, discrepancies may arise between the real program and its idealization by means of trace theory. More precisely, a C-dependence is generally neither symmetric nor reflexive; and, for a given partially commutative alphabet and CFG, it may well be that no program exists with the corresponding C-dependence relation.

Related work. A few formal studies of C-dependences exist such as [10], but not oriented towards program transformation. We know of just one work [5] on locally defined traces, investigating the word problem for certain program loops. In looser sense, some similarity of objectives may be seen between our work and the much more established line [8] investigating the application of asynchronous automata to concurrent programs.

As we did not find any previous work useful to apply trace theory for program optimization, we investigated how C-dependences constrain commutation. The next simple example introduces the problem.

$$\begin{array}{c} a: \text{ read } (x, y); \\ b: \text{ if } x > 0 \text{ goto } c \text{ else goto } d; \\ c: x = x - 1; \text{ goto } e; \qquad d: x = x + 1; \text{ goto } e; \\ e: \text{ print } (y); \end{array}$$

The runs are just $\{abce, abde\}$. Since instruction e is data-dependent only on a, among the runs obtained by permuting independent instructions such as c and e, we have $\{abec, abde\}$. Clearly the two sets represent the same trace language. But strings of the latter set, when viewed as CFG paths, imply the existence of path $abdec$, which violates the program semantics since both successors of predicate b are executed! The inconsistency comes from overlooking an essential constraint: instruction rescheduling must ensure that the original and the transformed program have the same C-dependences.

This paper sets a new rigorous framework for such program transformations, combining and extending some basic results of trace and compilation theories. The paper proceeds as follows. Sect. 2 contains basic definitions from compiler theory, and states simple but elsewhere non-available properties of post-dominance and C-dependence relations. Sect. 3 specializes trace theory for C-dependences. Any local language is star-connected with respect to

C-dependences, hence this trace language family is recognizable. Moreover CFG local languages unambiguously represent traces, and identity of traces implies identity of C-dependences. Sect. 4 is a significant application: within the family of local languages with the same C-dependences, we define and construct the language such that instructions are maximally anticipated. This language differs from the Foata-Cartier normal form, which in general is not a local language. Sect. 5 lists future directions for application of trace theory to program parallelization.

2 Basic Definitions and Properties

For the basic concepts of formal language and trace theory we refer primarily to [6], for compiler theory to e.g. [2,1,9]. Let Σ be a finite alphabet. The set of all strings over Σ is denoted by Σ^*, including the empty string, denoted by ε. For any string $x \in \Sigma^*$, $|x|$ denotes the length of x, $x(i)$, with $1 \leq i \leq |x|$, is the i-th character of x, $alph(x)$ denotes the set of letters present in x, and $\pi_\Delta(x)$ denotes the projection of x on a set $\Delta \subseteq \Sigma$ of letters.

In compilation and software engineering a program is often represented by a control flow graph CFG, a single entry (i), single exit (t) directed graph $G = (\Sigma, E)$. Let $pred(b), succ(b)$ denote the predecessors and successors of node b. The following customary hypotheses will be tacitly assumed.

1. Each node has at most two successors. A node with two successors is a *predicate* (or conditional instruction).
2. The successor of a node cannot be the node itself, i.e. the CFG has no self-loops.
3. For any node b there exists a path from the i to b, and a path from node b to t.

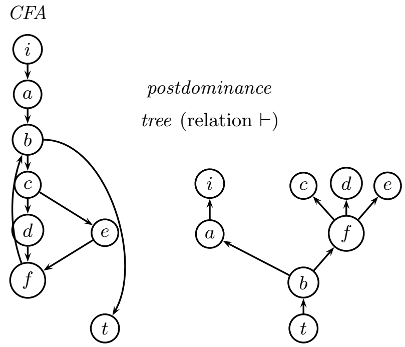
The definition of CFG is next rephrased within automata theory, using a restricted type of DFA known as *local automaton* [3].

Definition 1. *A* <u>*Control Flow Automaton*</u> *representing a CFG* $G = (\Sigma, E)$ *is a DFA* $A = (Q, \Sigma, \delta, q_0, F)$ *where the state set is* $Q = \Sigma$*, the initial and final states are* $q_0 = i$*,* $F = \{t\}$ *, the graph of the transition function* δ *is such that* $\delta(p, b) = q$ *if, and only if,* $p = a$*,* $q = b$ *and* $a \rightarrow b \in E$*. A recognizes a language* $L(A)$*. The corresponding language family is termed* <u>*CFG family*</u> *and denoted by* $\mathcal{CFG}$*.*

A CFA defines a local language [3], and $\mathcal{CFG}$ is strictly included in the family of local regular languages, because of the above restrictions on CFG. In particular, a language in $\mathcal{CFG}$ is never empty and may not contain the empty string.

It is known that local automata are convenient visualized identifying the states with terminals and omitting the transitions labels, as shown in figure 1.

program
i entry
a statement
b while condition
c if condition
d then statement
e else statement
f statement end
t exit



D_3	D_{3_2}	D_2
$(b,b)_c$	(b,b)	(b,b)
$(c,b)_c$	(c,b)	(c,b), (b,c)
$(d,c)_d$	(d,c)	(d,c), (c,d)
$(e,c)_e$	(e,c)	(e,c), (c,e)
$(f,b)_c$	(f,b)	(f,b), (b,f)
		(i,i), (a,a), (c,c), (d,d), (e,e), (f,f), (t,t)

Fig. 1. Program, CFA, strict post-dominance tree, and C-dependence relations

In compiler theory two properties of a CFG play a major role: predominance and postdominance. An instruction a post-dominates instruction b if, and only if, after executing instruction b, instruction a is always executed.

Definition 2. *Let A be a CFA, and let $a \neq b$. The strict postdominance relation $\vdash_s \subseteq \Sigma \times \Sigma$ is*

$$a \vdash_s b \textit{ if, and only if, for any } x \in L(A): \ \pi_{\{a,b\}}(x) \in \{(a,b)^* a \cup \epsilon\}$$

Then the postdominance relation $\vdash$ is obtained adding to $\vdash_s$ the identity relation. The immediate postdominance relation $\vdash_i$ is

$$a \vdash_i b \textit{ iff } a \vdash_s b \textit{ and } a \textit{ does not postdominate any other dominator of } b$$

Similarly, instruction a strictly *predominates* instruction b if, and only if, before executing instruction b, instruction a is always executed. The notation for predominance is $a \dashv_s b$. Postdominance and predominance are partial order reflexive relations and more precisely tree partial orders.

The concept of an instruction depending on a predicate has been formalized within compiler and software engineering research. Slightly different formulations exist (e.g. [10]) and we follow [7,2,9]).

For a CFA A or $\mathcal{CFG}$ language L, let the sub-alphabet $\Sigma_2 \subset \Sigma$ contain the letters having two successors, and let $\Sigma_1 = \Sigma \setminus \Sigma_2$. The letters in Σ_2 represent predicates.

Definition 3. *For a $\mathcal{CFG}$ language $L \subseteq \Sigma^+$ the ternary C-dependence relation $D_3 \subset \Sigma \times \Sigma_2 \times \Sigma$, pronounced as a is control dependent on b via c, is defined as*

$$(a,b)_c \in D_3(L) \textit{ if, and only if, } (a \vdash c) \wedge \neg(a \vdash_s b) \wedge c \in succ(b).$$

By erasing the third argument, we obtain the binary intermediate relation

$$D_{3_2}(L) = \{(a,b) \mid (a,b)_c \in D_3(L) \textit{ for some } c \in \Sigma\}$$

Then closing $D_{3_2}(L)$ by means of commutation and adding the identity relation, we obtain the binary C-dependence relation

$$(a,b) \in D_2(L) = \{(a,b) \mid (a,b) \in D_{3_2}(L) \vee (b,a) \in D_{3_2}(L) \vee (a=b)\}$$

Examples are shown in Figures 1 and 2. Notice that in the ternary relation a may coincide with c or a may coincide with b, but b necessarily differs from c because the successor relation is irreflexive. Intuitively, a is control dependent on b via c when predicate b has a successor c such that, if c is executed then also a is surely executed, but, if the other successor of b is taken, it is not certain that a will be later executed. The intermediate binary relation is less informative; it does not say which of the successors of the predicate is always followed by a. Finally the binary relation is symmetric and reflexive, to be consistent with the usual assumptions of trace theory.

Some properties of C-dependences are next stated, which are later needed. Let A be a CFA and $L \subseteq \Sigma^*$ be its language.

Statement 1. *The C-dependence relation $D_3(L)$ is empty iff $\Sigma_2 = \emptyset$.*

In other words, there no C-dependences if, and only if, the CFA graph has no bifurcation. Notice that the "only if" part of the statement is not obvious.

Statement 2. *Let $b \in \Sigma_2$, with $succ(b) = \{a, d\}$. Then at least one of $(a,b)_a$ or $(d,b)_d$ is in $D_3(L)$.*

Proof. By contradiction, assume

$$\exists a \in \Sigma \text{ such that } |succ(a)| > 1 \text{ and } \forall b \in succ(a) \text{ it is } (b,a)_b \notin D_3(L).$$

Since by definition of CFA, $\forall b \in succ(a)$ it is $b \neq a$, it follows that $\forall b \in succ(a)$, b postdominates a. Let now $\{b_1, b_2\} = succ(a)$; but if $b_1 \vdash a \wedge b_2 \vdash a$, then either $b_1 \vdash b_2$ or $b_2 \vdash b_1$. Considering the case $b_1 \vdash b_2$ (the other case is analogous), any string $w \in L$ containing b_2 can be factorized as $w = y_1 b_2 y_2 b_1 y_3 t$, with b_2 not occurring in y_2 or in y_3. But since $b_2 \in succ(a)$, there exists a string $w_1 \in L$ such that $w_1 = x_1 a b_2 x_2 t$, with b_1 not occurring in x_2. Hence b_1 does not postdominate a, a contradiction. Moreover, Figure 1 provides an example where one successor (t) of a predicate (b) is not control dependent on the predicate. □

Thus (at least) one of the successors of a predicate is control dependent on it.

For a CFA, a *circuit* $O = a_1, a_2, \ldots, a_n$ is a sequence of states $a_i \in \Sigma$, such that for all positions but the last, a_{i+1} is in $succ(a_i)$ and a_1 is in $succ(a_n)$. A node a_i present in O, having a successor $b \in succ(a_i)$ not in O, is called an *exit from circuit* O. An *iterative factor* of a language L is a string w such that $xw^*z \subseteq L$, for some strings x and z. A circuit or iterative factor is *simple* if all letters are distinct. Clearly for a $\mathcal{CFG}$ language a simple iterative factor is a simple circuit of the CFA graph.

The next statement, informally presented in [7], characterizes the cases of cyclicity of the C-dependence relation. The property immediately follows from Lemma 1 of Section 3.

Statement 3. *The binary intermediate relation $D_{3_2}(L)$ contains a cycle if, and only if, the graph of A contains a circuit. Moreover, when a CFA circuit has more than one exit, the exit nodes exhibit mutual C-dependences.*

For instance, consider a program loop such as the one in fig.1. Node b is the the loop exit, and it is control dependent on itself. Notice that this statement does not imply that a set of letters that make a loop within the graph of A, make a loop within D_{3_2} as well.

The next statement relates C-dependence and pre-, post-dominance relations.

Statement 4. *Let $a, b \in \Sigma$. If $\nexists c, d \in \Sigma$ such that $(a, c) \in D_{3_2}(L)$ and $(b, d) \in D_{3_2}(L)$, then the following two conditions are met: $a \dashv b \vee b \dashv a$ and $a \vdash b \vee b \vdash a$.*

3 Control Flow and Traces

This central section studies traces and C-dependences.

Let $D \subseteq \Sigma \times \Sigma$ be a symmetric and reflexive *dependence* relation, and its complement be denoted by I and named the *independence* relation. The dependence alphabet is the pair (Σ, D). The equivalence relation over Σ^* induced by I is denoted by $\sim_I$. The free partially commutative monoid, i.e. the quotient of Σ^* by the congruence $\sim_I$ is denoted by $M(\Sigma, I)$. For a string $x \in \Sigma^*$, the equivalence class of x under $\sim_I$ is called a *trace* and denoted by $[x]_I$. The mapping from strings to traces is denoted by φ: $\varphi(x) = [x]_I$. For a string language $X \subseteq \Sigma^*$, the mapping $\varphi(L)) = \{[x]_I \mid x \in X\}$ produces a *trace language*, also denoted by $[X]_I = \varphi(L)$. We use interchangeably I and its complement D, if no confusion arises. The subscript I is dropped if the (in)dependence relation is understood.

A trace or string is *D-connected* if its letters induce a connected subgraph of the dependence graph; a language is D-connected if every sentence is so.

Given a trace language T over a trace monoid $M(\Sigma, D)$, the family of all languages $L \subseteq \Sigma^*$ such that $[L]_I = T$ is denoted by $\mathcal{L}(T)$. Any language in $\mathcal{L}(T)$ is a *representative* of T.

For a subset $\Delta \subseteq \Sigma$ and a binary relation $D \subseteq \Sigma \times \Sigma$, the relation (or graph) *induced by* Δ is $D_{|\Delta} = D \cap \Delta^2$.

We refer to [6,4] for the notions of recognizable $\mathcal{R}ec$, rational $\mathcal{R}at$, and unambiguous trace language, and for the Foata-Cartier Normal Form.

Traces of $\mathcal{CFG}$ Languages

We consider a control-flow automaton A_0, recognizing the language $L_0 = L(A_0) \subseteq \Sigma^*$, which is, as we know, a local regular language and more specifically a member of the $\mathcal{CFG}$ language family. Let $D_3(A_0) = D_3(L_0)$ be the C-dependence relation and D_2 its symmetric and reflexive embedding. Since we do not consider data-dependences (except in Sect. 5), we may take the independence relation to be $I = (\Sigma \times \Sigma) \setminus D_2$.

A regular expression is *D-star-connected*, if the star is used over D-connected languages only. For a CFA, the definition becomes: if, for every circuit of the DFA graph, the state labels are D_2-connected.

Definition 4. *Let L_0 be a language in $\mathcal{CFG}$. The trace language $[L_0]_{D_2(L_0)}$ is called the trace language C-defined by L_0. The family of CFG trace languages is*

$$\mathcal{T}_{CFG} = \{T \mid T \textit{ is a trace language C-defined by } L_0, \textit{ for some } L_0 \textit{ in } \mathcal{CFG}\}$$

The next technical Lemma will be used to prove that, for any CFG language, the graph of the D_{3_2} relation is connected for every iterative factor.

Lemma 1. *Let $L \subseteq \Sigma^*$ be a language in $\mathcal{CFG}$ and $D_{3_2} = D_{3_2}(L)$ be the intermediate binary C-dependence relation. Let w be a simple iterative factor with $W = alph(w)$. Then for every pair of letters $a, e \in W$ such that e is an exit from w, the graph $D_{3_2}{}_{|W}$ contains a (not necessarily directed) path, termed a D-path, between a and e.*

Proof. The proof is by induction. Let $PATHS_k$, $k \geq 2$, be the set of CFG paths of length k contained in circuit w, and denote with $EPATHS_k$ the paths ending in an exit node. We say $EPATHS_k$ has the D-connection property, if, for every path z in $EPATHS_k$, the graph $D_{3_2}{}_{|alph(z)}$ is connected[1].

Induction base. For circuit w, let ab be a path in $EPATHS_2$. Since b exits from w, from the definition of control-dependence it follows that $(a, b) \in D_{3_2}$. Thus $EPATHS_2$ has the D-connection property.

Inductive hypothesis. From the induction base, we may assume that for circuit w with $|w| = k$, the set $EPATHS_j$ has the D-connection property, for all $2 \leq j \leq k - 1$.

Induction step. We prove that $EPATHS_k$ has the D-connection property. Let $b_1 b_2 \ldots b_{k-1} b_k$ be a path in $EPATHS_k$. Since the path $x = b_2 \ldots b_{k-1} b_k$ is in $EPATHS_{k-1}$, it has the D-connection property. It suffices to show that b_1 is D-connected to a letter in $alph(x)$, and we prove it for b_2. Two cases arise.

b_2 **postdominates** b_1**.** Since b_2 is control-dependent on some letter in $\{b_2, \ldots, b_{k-1} b_k\}$, from the definition of control-dependence it follows b_1 is dependent on the same letter.

b_2 **does not post-dominate** b_1**.** Then b_1 is a predicate with b_2 as one of its successors. From Statement 2 it follows that $(b_2, b_1)_{b_1} \in D_3$. □

Clearly the property remains true for non-simple iterative paths. The next trace language family inclusion follows immediately.

Theorem 1. *The $\mathcal{T}_{CFG}$ family is strictly included within the family $\mathcal{R}ec$ of recognizable trace languages.*

Proof. Inclusion follows from the fact [6] that a trace language T is recognizable, if, and only if, there exists a regular language $X \subseteq \Sigma^*$ such that every iterative factor of X is D-connected and $T = [X]$. Moreover, the inclusion is strict: the (finite) language $R = \{abc\}$ with the dependence relation $D = \{(a, b)\}$ is recognizable, but D differs from the binary control relation $D_2(R)$, which is empty (Statement 1). □

[1] We prefer to use the intermediate binary relation D_{3_2} instead of D_2 because it makes more perspicuous the arguments based on the properties of control-dependences.

Control Equivalent Automata

We have argued in the introduction that a CFG language, though equivalent modulo the C-independence relation to a given language, may not correspond to a semantically equivalent program. This fact is precisely stated in the next theorem.

Theorem 2. *Let L' and L'' be languages in $\mathcal{CFG}$ over the same ranked alphabet $\Sigma = \Sigma_1 \cup \Sigma_2$. If the trace languages C-defined by the two languages are identical, then the binary C-dependence relations are equal, in formula*

$$\textit{if } [L']_{D_2(L')} = [L'']_{D_2(L'')} \textit{ then } D_2(L') = D_2(L'')$$

but the converse implication does not hold.

In other words, for a given partition of the alphabet into predicates (Σ_2) and non-predicates, a trace language uniquely determines the C-dependence relation for all CFG representatives. On the other hand, two CFG languages having equal D_2, may represent different trace languages and therefore, they are not semantically equivalent when viewed as programs. For this reason, the theorem calls for a new stronger concept of equivalence.

Definition 5. *Let $L_0 = L(A_0)$ be a language in $\mathcal{CFG}$ and let $T_0 = [L_0]_{D_2(L_0)}$ be the C-defined trace language. The family $\mathcal{L}_C(L_0) \subseteq \mathcal{CFG}$ includes the languages L, such that $D_3(L) = D_3(L_0)$ and $[L]_{D_2(L_0)} = T_0$. The corresponding family of CFG automata is denoted by $\mathcal{A}_C(L_0)$. The languages (or automata) in these families are termed C-equivalent.*

For a given CFG language, we are interested in studying C-equivalent languages, having desirable properties. With the terminology of compilers, they can be viewed as obtained by legal program transformations.

To conclude this part, we formalize a fact anticipated in the introduction: a regular language, which represents the same trace as a CFG language L_0, may not even be in $\mathcal{CFG}$.

Statement 5. *Let L_0 be a language in $\mathcal{CFG}$ and T_0 its C-defined trace language. The language family $\mathcal{L}_C(L_0)$ is strictly included within the family $\{R \mid R \text{ is a regular language and } [R]_{D_3(L_0)} = T_0\}$.*

Proof. Weak inclusion is obvious. Strictness of inclusion follows from the example in Sect. 1. The set $L_0 = \{abce, abde\}$ is in $\mathcal{CFG}$, with $D_2(L_0) = \{(b,c),(b,d)\}$ (omitting the identity relation). The language $R = \{aebc, abde\}$ defines modulo $D_2(L_0)$ the same trace language C-defined by L_0, but it is not a local language, hence not a CFG language. □

Unambiguity

A natural question to ask concerns ambiguity. We recall a rational trace language T is unambiguous if, there exists a regular language X such that $T = [X]_I$ and for every trace $t \in T$, X contains exactly one representative for t. In that case we say that language X *unambiguously defines* trace language T. Since all recognizable trace languages are unambiguous, from Theor. 1 the same holds for the $\mathcal{T}_{CFG}$ family. The next statement solves the question whether every $\mathcal{CFG}$ language unambiguously defines its trace language.

Theorem 3. *Consider a language L in $\mathcal{CFG}$, the C-defined trace language T, and the family $\mathcal{L}_C(L)$ of CFG languages which C-define T. Every language in this family unambiguously defines T.*

Proof. It is based on Lemma 2 and Lemma 3. We show that for any distinct strings $w_1, w_2 \in L$ it is $[w_1]_{D_2(L)} \neq [w_2]_{D_2(L)}$. Assume by contradiction that $[w_1]_{D_2(L)} = [w_2]_{D_2(L)}$.

If $D_3(L) = \emptyset$, from statement 1 it follows that each letter has at most one successor, and, since the first letter must be i, $|L| = 1$, hence $w_1 = w_2$, a contradiction.

It remains to analyze the case $D_3(L) \neq \emptyset$. Consider the longest common prefix xa of the two strings $w_1 = xaby_{11} \qquad w_2 = xacy_{21}$ with $b \neq c$ and $y_{11}, y_{21} \in \Sigma^*$.

Since $[w_1]_{D_2(L)} = [w_2]_{D_2(L)}$ implies $\pi_b(w_1) = \pi_b(w_2) \wedge \pi_c(w_1) = \pi_c(w_2)$, letter b must occur in y_{21} and letter c must occur in y_{11}, hence the factorizations

$$w_1 = xaby_{12}cy_{13} \qquad w_2 = xacy_{22}by_{23}$$

where $y_{12}, y_{13}, y_{22}, y_{23} \in \Sigma^* \wedge c \notin y_{12} \wedge b \notin y_{22}$. Moreover, since w_1 and w_2 are in the same trace, b and c are independent letters (i.e. $(b, c) \notin D_{3_2}(L)$).

From Lemma 3, it follows that $\forall d \in y_{12}, (c, d) \notin D_{3_2}(L)$ and $\forall d \in y_{22}, (b, d) \notin D_{3_2}(L)$. Moreover, from Lemma 2, we have $c \vdash b$ and $b \vdash c$. Since the post-dominance relation makes a tree over Σ, and $c \vdash b$ and $b \vdash c$, then $c = b$. Contradiction found. □

Lemma 2. *Let $L \in \mathcal{CFG}$ contain a sentence of the form $xyaz$, for a letter a, and some strings $x, z \in \Sigma^*, y \in \Sigma^+$. Then*

$$\exists c, d \in ya \textit{ such that } (a, c)_d \in D_3 \textit{ if, and only if, } \exists e \in ya \textit{ such that } \neg(a \vdash e)$$

Proof. We provide the proof in two rounds.

$\underline{\exists c, d \in ya \mid (a, c)_d \in D_3 \Rightarrow \exists e \in ya \mid \neg(a \vdash e)}$ By contradiction, we suppose that:

$$(\exists c, d \in ya \mid (a, c)_d \in D_3) \wedge (\nexists e \in ya \mid \neg(a \vdash e))$$

Then it follows $\forall e \in ya, (a \vdash e)$. Since $(a, c)_d \in D_3$ follows that $\neg(a \vdash c)$ by definition of D_3. Since $c \in ya \wedge \forall e \in ya, (a \vdash e)$ then $a \vdash c$. Contradiction found.

$\underline{\exists c, d \in ya \mid (a, c)_d \in D_3 \Leftarrow \exists e \in ya \mid \neg(a \vdash e)}$ We provide the proof by induction.

Induction base. For the string ya, let the length of y be one and then $y \in \Sigma$. Since $a \vdash a$ by definition of the pre-dominance relation, then the Lemma becomes $\neg(a \vdash y) \Rightarrow (a, y)_a \in D_3$ which is true by definition of D_3.

Inductive hypothesis. From the induction base, we may assume that for any y with length equal to $n-1$ the Lemma holds.

Induction step. We prove that for any y with length n, the Lemma holds. We can write the string as xdy_1az where $y = dy_1$. If $\neg(a \vdash y_1(1))$, then by the Inductive hypothesis, the Lemma holds. Then it suffices to consider the case $a \vdash y_1(1)$. Two cases arise:

- $(\neg(a \vdash d))$: by definition of D_3 it follows $(a,d)_{y_1(1)} \in D_3$ and then the Lemma holds.
- $(a \vdash d)$: then $\nexists e \in ya \mid \neg(a \vdash e)$ and then the Lemma holds.

□

Lemma 3. *Let L_0 be in $\mathcal{CFG}$, and let $w_1, w_2 \in L_0$ be sentences representing the same trace $t = [w_1]_{D_2} = [w_2]_{D_2}$, such that for some letters $a \neq b \in \Sigma$, $w_1 = xax_{11}bx_{12}$ and $w_2 = xbx_{21}ax_{22}$, where $x, x_{11}, x_{12}, x_{21}$ and x_{22} are possibly empty strings. Then*

- $\forall e \in alph(x_{11})$ *and* $\forall f \in \Sigma$ *it is* $(b,e)_f \notin D_3(L_0)$;
- $\forall e \in alph(x_{21})$ *and* $\forall f \in \Sigma$ *it is* $(a,e)_f \notin D_3(L_0)$.

Comment: consider a program represented by a CFA. Theor. 3 says that it is impossible for a program to have two runs that are a permutation of each other. Moreover this remains impossible for any program that is C-equivalent to the original one.

4 Maximally Parallel Program

In this section we show how to transform a CFA into a semantically equivalent one, and especially into the one with maximal parallelism. For the latter we discuss its relation with the Foata-Cartier normal form of traces.

Ordering by Degree of Parallelism

To compare the degree of parallelism of programs, we introduce two different binary relations over string languages that define the same trace language. We need some definitions. A clique of the independence graph is a set of mutually independent letters. The set of such cliques is denoted by Σ_I. For a string $s \in \Sigma^*$ the *clique decomposition* is $s = s_1 s_2 ... s_n$ where each s_i is a clique, and each s_j, $1 \leq j < n$ contains a letter such that there exists a dependent letter in s_{j+1}. The string decomposition must be computed from left to right, thus ensuring its uniqueness for any given string. The *height* of the string is $height(s) = n$.

Let L_1 and L_2 be string languages. We define an order relation $\geq_P$ on strings $s_1 \in L_1, s_2 \in L_2$ representing the same trace. We say string s_1 is *more parallel* than s_2, written $s_1 \geq_P s_2$, if for their clique decompositions $s_1 = s_{1,1}s_{1,2}...s_{1,n}$ and $s_2 = s_{2,1}s_{2,2}...s_{2,m}$ it is $n \leq m$. This relation induces a partial order on languages: L_1 is more parallel than L_2, written $L_1 \geq_P L_2$, if for each string $s_1 \in L_1$ and $s_2 \in L_2$ with $[s_1]_{D_2(L_1)} = [s_2]_{D_2(L_2)}$, it is $s_1 \geq_P s_2$. The strict relation $>_P$ is defined in the obvious way, using existential quantifier.

Of two strings with equal height, one may place a certain letter in an earlier clique than the other. This concept of greediness or anticipation is captured by another order relation $\geq_G$ on strings $s_1 \in L_1, s_2 \in L_2$ representing the same trace and having equal height. String s_1 is *greedier* than s_2, written $s_1 \geq_G s_2$, if for the clique decompositions $s_1 = s_{1,1}s_{1,2}...s_{1,n}$ and $s_2 = s_{2,1}s_{2,2}...s_{2,n}$ the following condition holds. Let $sign(|s_{1,i}| - |s_{2,i}|)$ be a symbol in $\{0, n, p\}$. Then the sequence of signs is in $0^*p(n \mid p)^*$. The $\geq_G$ naturally induces a partial order on string languages.

We are especially interested in a language that is more parallel and greedy than any other C-equivalent language.

Computing C-Equivalent Languages

We move into application, to present an algorithm for computing the set of C-equivalent automata (vs Def. 5). After proving correctness and completeness of the algorithm, we show how to construct the most parallel and greedy automaton.

Definition 6. *Let A be a CFA with $L(A) \subseteq \Sigma^*$, and let D_3 be the ternary C-dependence relation. The following transformation of the graph of A is termed "moving a letter b to a letter a". Let j be a letter such that:*

$$j \vdash_i b \wedge \forall p \in pred(j) \mid b \dashv p \qquad (1)$$

The edges E' of the CFG (Σ, E) of A become as following:

- $E_{del} = \{(p, d) \mid (d = a \vee d = b \vee d = j) \wedge (p, d) \in E\}\}$
- $E_{add} = \{(p, b) \mid (p, a) \in E\} \cup \{(p, j) \mid (p, b) \in E\} \cup \{(p, a) \mid (p, j) \in E\}$
- $E' = (E \setminus E_{del}) \cup E_{add}$

Notice that a becomes the immediate postdominator of b; if $b \in \Sigma_2$, the move can shift other nodes which are transitive successors of b. Such a transformation is called a *legal move* if the new automaton recognizes a language C-equivalent to $L(A)$.

Consider a CFA A and its graph. For each non-initial node a of the graph, we define the *guest set*, $Guest(a) \subseteq \Sigma \setminus \{i, t\}$, which includes the letters b such that:
$\forall c, d \in \Sigma, (a, c)_d \in D_3 \Leftrightarrow (b, c)_d \in D_3$; $\forall c \in \Sigma, (a, c) \in D_{3_2} \Rightarrow (a \dashv c \Leftrightarrow b \dashv c)$ and $\exists j$ such that condition 1 is met. The sets are computed from C-dependence and pre-dominance relations. Since the latter can be computed in polynomial time, the computation of guest sets is in the complexity class P.

The next technical Lemma will be used to prove that we can construct any C-equivalent automaton by moving some letter from its current position in the graph of A, to some node which has the letter as guest.

Lemma 4. *Let L_1 be a $\mathcal{CFG}$ language, the move of b to a is legal if, and only if, $b \in Guest(a)$.*

Theorem 4. *Let $\mathcal{A}$ be the class of automata computed by applying one or more legal moves to A, and $L_0 = L(A)$. Then $\mathcal{A}$ coincides with the class $\mathcal{L}_C(L_0)$.*

The following statement characterizes the pre- and post-dominance properties of guest letters, and will be used to compute the maximally greedy and parallel automaton.

Statement 6. *Let $a, b \in \Sigma$. If $b \in Guest(a)$, i.e. a is a host of b, then the following two relations hold: $b \dashv a \vee a \dashv b$ and $b \vdash a \vee a \vdash b$.*

Maximally Parallel and Greedy Control Flow Automaton

For a given CFA automaton A, we define the *most parallel and greedy* automaton, A_{max}, as the CFA obtained by the following transformation.
For every letter b, legally move b to a letter a such that no other letter c, which pre-dominates a, has b as guest, in formula: $\nexists c \in \Sigma$ such that $b \in Guest(c) \wedge c \dashv a$.

Statement 7. *Let $L = L(A)$ be in $\mathcal{CFG}$ and A_{max} be defined as above. Then $L(A_{max})$ is the most parallel and greedy language C-equivalent to L, i.e. it is: $\forall L' \in \mathcal{L}_C(L)$, $L(A_{max}) >_P L'$ or $L(A_{max}) \geq_G L'$.*

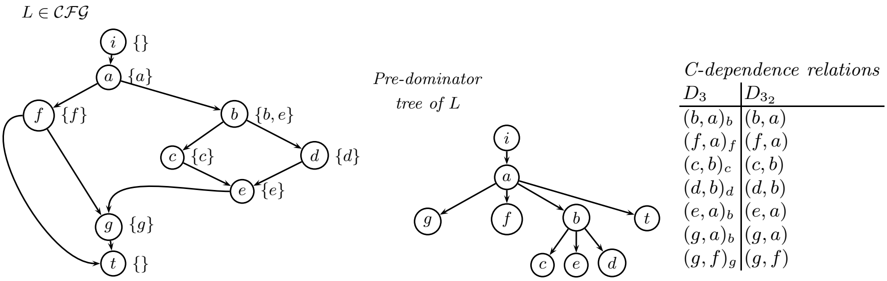
Proof. Since each move applied is legal, i.e. $b \in Guest(a)$, from Theor 4 it follows that the automaton is C-equivalent to A. The proof of maximal greediness is based on Stat. 6.

Foata-Cartier Normal form vs. Maximally Parallel Program

It is known that for a trace $[x]$, the clique decomposition associated to the Foata-Cartier normal form gives the most parallel and greedy form of x. The next example proves that the set of such Foata-Cartier decompositions, in general, is not C-equivalent to the given CFG language. Therefore it differs from $L(A_{max})$, which has been proved to be the most parallel and greedy language which preserves C-equivalence.

For $L \in \mathcal{CFG}$ we denote as A_{Foata} the minimum DFA that recognizes the Foata-Cartier normal forms of the strings in L. More precisely, since there is more than one automaton A_{Foata}, depending on the serialization of the letters in an independence clique, we may assume that serialization complies with the lexicographic order of the terminal alphabet.

Example 1. Let L be the *CFG* language recognized by the CFA A in Figure 2. C-dependence and dominance relations and guest sets are also shown. Let the letters be lexically ordered as listed: $\Sigma = \{i, a, b, c, d, e, t\}$. The Foata-Cartier automaton and A_{max} are in Figure 3.



D_3	D_{3_2}
$(b,a)_b$	(b,a)
$(f,a)_f$	(f,a)
$(c,b)_c$	(c,b)
$(d,b)_d$	(d,b)
$(e,a)_b$	(e,a)
$(g,a)_b$	(g,a)
$(g,f)_g$	(g,f)

Fig. 2. CFA with relations and guest sets

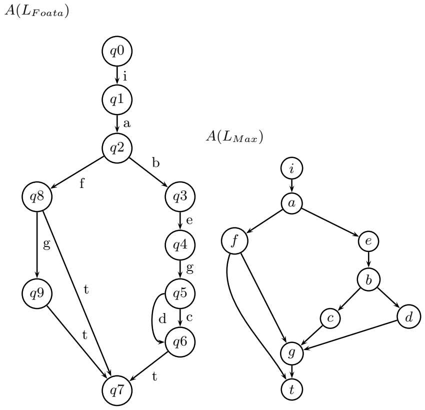


Fig. 3. Foata-Cartier automaton and maximally parallel and greedy automaton

The I-cliques present in the Foata-Cartier strings differ from the cliques in $L(A_{max})$. Clearly A_{Foata} is not C-equivalent to A, since it is not a local automaton. Moreover, this example shows that $L(A_{Foata}) \geq_P L(A_{max})$.

We summarize the above discussions and example in the following statement.

Statement 8. *The automaton A_{Foata} defines the maximally parallel and greedy language $L(A_{Foata})$. The automaton A_{max} defines the maximally parallel and greedy C-equivalent language.*

Technical remark: to achieve the Foata-Cartier parallelism, copies of the same instruction have to be differentiated in order to re-obtain the local property of the automaton. This is an ordinary transformation for compilers, that replicates the same instructions at different addresses.

5 Conclusion

We have shown that program transformations consisting in instruction rescheduling can be conveniently studied and actually implemented using results from trace theory, combined with concepts and methods of compiler theory. Control-dependence and pre- post-dominance relations play an essential role in that. The automata constructions described are efficient and have a potential for compilation.

Of course data-dependences too must be considered for realistic applications. It is an easy job to superimpose data- onto control constraints. Another straightforward development is to study a program transformation with the aim to achieve uniform parallelism at all steps, instead of maximal greediness.

We believe that this effort for expressing program transformations by means of a suitably enriched trace theory should be continued to cover other parallelizing transformations done by compilers, such as speculative execution, loop unrolling or software pipelining. Such developments are likely to need other algebraic concepts, in particular partially inverse monoids.

Lastly, we observe that we have proved several properties for the traces represented by CFG languages, which are a subclass of local languages. It is not known whether such properties still hold for the latter.

Acknowledgement. we would like to thank Massimiliano Goldwurm for critical reading and suggestions.

References

1. Aho, A.V., Lam, M.S., Sethi, R., Ullman, J.D. (eds.): Compilers: principles, techniques, and tools, 2nd edn. Pearson/Addison Wesley, Boston (2007)
2. Appel, A.W., Palsberg, J.: Modern compiler implementation in Java, 2nd edn. Cambridge University Press, Cambridge (2002)
3. Berstel, J., Pin, J.-E.: Local languages and the Berry-Sethi algorithm. Theor. Comput. Sci. 155(2), 439–446 (1996)
4. Bertoni, A., Goldwurm, M., Mauri, G., Sabadini, N.: Counting techniques for inclusion, equivalence and membership problems. In: Diekert, V., Rozenberg, G. (eds.) Counting techniques for inclusion, equivalence and membership problems. The Book of Traces, ch. 5, pp. 131–163. World Scientific, Singapore (1995)
5. Breveglieri, L., Crespi Reghizzi, S., Savelli, A.: Efficient word recognition of certain locally defined trace languages. In: 5th Int. Conf. on Words (WORDS 2005), Montreal, Canada (2005)
6. Diekert, V., Métivier, Y.: Partial commutation and traces. In: Rozenberg, G., Salomaa, A. (eds.) Handbook on Formal Languages, vol. III (1997)
7. Ferrante, J., Ottenstein, K.J., Warren, J.D.: The program dependence graph and its use in optimization. ACM Trans. Program. Lang. Syst. 9(3), 319–349 (1987)
8. Klarlund, N., Mukund, M., Sohoni, M.: Determinizing asynchronous automata. In: Shamir, E., Abiteboul, S. (eds.) ICALP 1994. LNCS, vol. 820, pp. 130–141. Springer, Heidelberg (1994)
9. Muchnick, S.S.: Advanced compiler design and implementation. Morgan Kaufmann Publishers Inc., San Francisco (1997)
10. Pingali, K., Bilardi, G.: Optimal control dependence computation and the Roman chariots problem. ACM Transactions on Programming Languages and Systems 19(3), 462–491 (1997)

Left and Right Synchronous Relations

Olivier Carton

LIAFA, Université Paris Diderot & CNRS
http://www.liafa.jussieu.fr/~carton

Abstract. In this paper, we study rational relations that are both left and right synchronous. We show that these relations are boolean combinations of almost length preserving relations, length comparing relations and recognizable relations.

1 Introduction

We consider left (and right) synchronous rational relations. These relations are realized by letter to letter transducers that read both words synchronously from the left (or right), the shortest word being padded with an extra symbol at its right (or left) end. These relations are intensively used in the theory of automatic structures. Let us recall that a relational structure has an automatic presentation if the elements of its domain can be named by finite strings in such a way that the coded relations are realized by left (or right) synchronous transducers [4]. The closure properties of left (and right) synchronous relations imply that each automatic structure has a decidable first-order theory.

The main result of this paper is a characterization of rational relations that are both left and right synchronous. This natural question was raised by Sakarovitch [5, Problem 6.2 (p. 659)]. It is easy to see that some simple classes of rational relations only contain both left and right synchronous relations. For instance, any recognisable relation is obviously both left and right synchronous. Let us recall that a relation is recognizable if it is a finite union of product $K_i \times L_i$ where the K_i and L_i are rational sets of words. Length preserving relations are also both left and right synchronous. A relation is length preserving if it consists of pairs (u, v) such that u and v have the same length. We say that a relation R is almost length preserving if all the differences $|u| - |v|$ for (u, v) in R are bounded by some constant M. These relations were already considered in [3]. Finally, each relation of the form $\{(u, v) \mid |u| \leq |v| + M\}$ for some constant M is also both left and right synchronous. The main result is that any both left and right synchronous relation is a boolean combination of such simple relations.

The proof of the main result is obtained through an intermediate characterization of both left and right synchronous relations. This characterization states that a relation is left and right synchronous if and only if it fulfils the following two conditions. First, it coincides with a recognizable relation for pairs (u, v) of words with a great difference of lengths, that is $\big||u| - |v|\big|$ greater that some fixed threshold. Second, it coincides with an almost length preserving relation for pairs (u, v) of words with a small difference.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 170–182, 2009.

The question of the synchronous relations goes back to the paper of Elgot and Mezei [2] about rational relations realized by finite automata, and to the result of Eilenberg and Schützenberger [1], which states that a length preserving rational relation of $A^* \times B^*$ is a rational subset of $(A \times B)^*$, or, equivalently, is realized by a synchronous automaton.

The paper is organized as follows. Sect. 2 introduces the basic definitions of rational relations and transducers. The left and/or right synchronous relations are defined in Sect. 3 and the main result is stated in Sect. 4. Sect. 5 provides the intermediate characterization that makes the proof easier. The proof of the main result is finally completed in Sect. 6.

2 Preliminaries

In what follows, A and B denote finite alphabets. The free monoid A^* is the set of finite *words* or sequences of letters from A. The *empty word* is denoted by ε. The *length* of a word $u \in A^*$ is denoted by $|u|$. A word u is a *prefix* of a word v if there exists a word w such that $v = uw$.

In this paper, we study relations, that is, subsets of the product monoid $A^* \times B^*$.

A *transducer* (also known as a *two-tape automaton*) is a non-deterministic automaton whose transitions are labeled by pairs of words. A *transducer* over the monoid $A^* \times B^*$ is composed of a finite set Q of *states*, a finite set $E \subset Q \times A^* \times B^* \times Q$ of *transitions* and two sets $I, F \subseteq Q$ of *initial* and *final* states. A transition $\tau = (p, u, v, q)$ from p to q is denoted by $p \xrightarrow{u|v} q$.

A *path* in a transducer $\mathcal{T}$ is a sequence

$$q_0 \xrightarrow{u_1|v_1} q_1 \xrightarrow{u_2|v_2} \cdots \xrightarrow{u_n|v_n} q_n$$

of consecutive transitions. The *label* of this path is the pair (u, v) where its *input label* u is the word $u_1 u_2 \cdots u_n$ and its *output label* v is the word $v_1 v_2 \cdots v_n$. Such a path is denoted $q_0 \xrightarrow{u|v} q_n$. This path is *accepting* if q_0 is initial and q_n is final. The set *accepted* by the transducer is the set of labels of its accepting paths, which is a relation $R \subseteq A^* \times B^*$. We say that the relation R is *accepted* by the transducer $\mathcal{T}$.

A subset of $A^* \times B^*$ is *rational* if it can be obtained from some finite subsets using union, concatenation and star iteration. It is a consequence of Kleene's theorem that a subset of $A^* \times B^*$ is a rational relation if and only if it is the relation accepted by some transducer.

By boolean operations, we mean union, intersection and complementation. Therefore, we say that a class $\mathcal{C}$ of relations contained in $A^* \times B^*$ is *closed under boolean operations* if for any relation R and R' in $\mathcal{C}$, the three relations $R \cup R'$, $R \cap R'$ and $(A^* \times B^*) \setminus R$ are also in $\mathcal{C}$. We also say that a relation R is a *boolean combination* of relations in $\mathcal{C}$ if R belongs to the smallest class containing $\mathcal{C}$ and closed under boolean operations. For any two relations $R, R' \subseteq A^* \times B^*$, the product RR' is defined by $RR' = \{(uu', vv') \mid (u, v) \in R \text{ and } (u'v') \in R'\}$.

3 Left and Right Synchronous Relations

A relation $R \subseteq A^* \times B^*$ such that $|u| = |v|$ holds for any pair $(u, v) \in R$ is called *length preserving.*

A transducer $\mathcal{T}$ is *synchronous* if for each transition $p \xrightarrow{u|v} q$, the lengths of the input and output labels satisfy $|u| = |v| = 1$. It is straightforward that a relation accepted by a synchronous transducer is length preserving. By the result of Eilenberg and Schütenberger [1], the converse also holds. Any rational length preserving relation can be accepted by a synchronous transducer.

A synchronous transducer is actually an automaton over the product alphabet $A \times B$. As such, it can be made deterministic using the usual subset construction for finite automata. Therefore, we may always assume that synchronous transducers are deterministic.

Let # be a padding symbol that does not belong to the alphabets A and B. The *left padding* of a pair (u, v) of words over A and B is the pair $(u, v\#^{|u|-|v|})$ if $|v| \geq |u|$ and the pair $(u\#^{|v|-|u|}, v)$ otherwise. The left padding is called so because the two words are left aligned. The *right padding* of (u, v) is defined similarly but the shorter word is padded on the left. The left and right padding of (u, v) are respectively denoted $\overrightarrow{(u, v)}$ and $\overleftarrow{(u, v)}$. For a relation $R \subseteq A^* \times B^*$, we respectively denote by $\overrightarrow{R}$ and $\overleftarrow{R}$ the following two relations

$$\overrightarrow{R} = \{\overrightarrow{(u, v)} \mid (u, v) \in R\} \quad \text{and} \quad \overleftarrow{R} = \{\overleftarrow{(u, v)} \mid (u, v) \in R\}.$$

A relation $R \subseteq A^* \times B^*$ is called *left synchronous* (respectively *right synchronous*) if the relation $\overrightarrow{R}$ (respectively $\overleftarrow{R}$) can be accepted by a synchronous transducer over the alphabets $A \cup \{\#\}$ and $B \cup \{\#\}$.

Since a synchronous transducer can be assumed to be deterministic, it follows that the class of left (respectively right) synchronous relations is closed under boolean operations.

The transducers are pictured in figures by graphs with labeled edges. Each transition $p \xrightarrow{u|v} q$ is represented by an edge from p to q labeled by $u|v$. Initial states are marked by a small incoming arrow and final states by a small outgoing arrow.

Example 1. Let R be the set of pairs (u, v) such that u is a prefix of v. This relation is left synchronous since the relation $\overrightarrow{R}$ is accepted by the transducer of Fig. 1. It can be easily proved that this relation is not right synchronous.

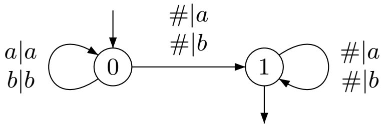


Fig. 1. Left synchronous transducer for the prefix relation

Example 2. Let A be the alphabet $\{a, b\}$ and let $R \subset A^* \times A^*$ be the relation defined by

$$R = \{(u, u) \mid u \in A^*\} \cup \{(u, v) \mid |u| > |v| \text{ and } |u| \equiv |v| \mod 2\}.$$

This relation is left and right synchonous. It is left synchonous since $\overrightarrow{R}$ is accepted by the transducer of Fig. 2. It is also right synchronous since a similar transducer accepts $\overleftarrow{R}$.

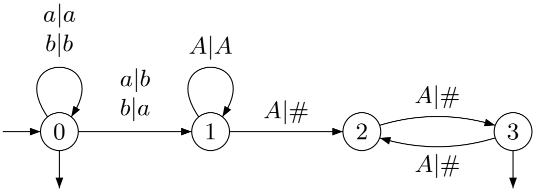


Fig. 2. Left synchonous transducer for the relation R of Example 2

The aim of this paper is to characterize relations that are both left and right synchronous.

4 Characterization

We now recall the definition of a class of very simple rational relations. A relation $R \subseteq A^* \times B^*$ is *recognizable* if there are two families $K_1, \ldots, K_n$ and $L_1, \ldots, L_n$ of rational subsets of A^* and B^* such that $R = \bigcup_{i=1}^{n} K_i \times L_i$.

Example 3. Let R be the relation $\{(u, v) \mid |u| \equiv |v| \mod 2\}$. This relation is recognizable since it is equal to the following finite union of direct products of rational languages.

$$R = (K_0 \times K_0) \cup (K_1 \times K_1) \quad \text{where} \quad K_i = \{u \mid |u| \equiv i \mod 2\}.$$

For each integer $k \geq 0$, we define three relations G_k, H_k and J_k as follows.

$$\begin{aligned} G_k &= \{(u, v) \in A^* \times B^* \mid |v| - |u| \leq -k\}, \\ H_k &= \{(u, v) \in A^* \times B^* \mid -k \leq |v| - |u| \leq k\}, \\ J_k &= \{(u, v) \in A^* \times B^* \mid k \leq |v| - |u|\}. \end{aligned}$$

For each integer $k \geq 0$, the equality $G_k \cup H_k \cup J_k = A^* \times B^*$ holds. Note however that the intersections are not empty since $G_k \cap H_k = \{(u, v) \mid |v| - |u| = -k\}$ and $H_k \cap J_k = \{(u, v) \mid |v| - |u| = k\}$. If $k < k'$, the three inclusions $G_k \supset G_{k'}$, $H_k \subset H_{k'}$ and $J_k \supset J_{k'}$ are strict.

Note that a relation R is length preserving if R is contained in H_0. We say that a relation $R \subseteq A^* \times B^*$ is *almost length preserving* if R is contained in H_k for some integer $k \geq 0$. If both relations R and R' are almost length preserving, the relations $R \cup R'$ and $R \cap R'$ are also rational and almost length preserving. Furthermore if R is contained in H_k, the relation $H_k \setminus R$ is also rational and almost length preserving.

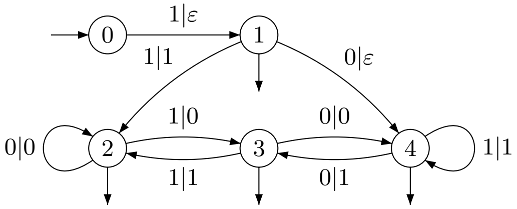


Fig. 3. Transducer for the division by 3 in base 2

Example 4. Let A be the alphabet $\{0,1\}$. Let $R \subseteq A^* \times A^*$ be the set of pairs (u,v) such that u is the binary expansion of some integer n and v is the binary expansion of $\lfloor n/3 \rfloor$. This relation R is not length preserving since $(11,1)$ is in R but it is almost length preserving since $1 \leq |u| - |v| \leq 2$ holds for any pair (u,v) of R. This relation is accepted by the transducer pictured in Fig. 3.

We say that a relation R is *length comparing* if $R = G_k$ or $R = J_k$ for some integer $k \geq 0$.

The following theorem is the main result of the paper. It gives a characterization of relations that are both left and right synchronous.

Theorem 1. *A relation is both left and right synchronous if and only if it is a boolean combination of recognizable relations, almost length preserving relations and length comparing relations.*

As already pointed out in Sect. 3, the class of left synchronous relations is closed under boolean operations. It follows that the class of left and right synchronous relations is also closed under boolean operations.

It is straightforward that recognizable relations, almost length preserving relations and length comparing relations are both left and right synchronous.

It follows from the previous remarks that a boolean combination of recognizable relations, almost length preserving relations and length comparing relations is left and right synchronous. The rest of the paper is devoted to the proof of the converse which is actually the interesting point. We show that any left and right synchronous relation can be decomposed as a boolean combination of recognizable relations, almost length preserving relations and length comparing relations. We can already point out that this boolean combination will involve unions and intersections but no complementation.

The proof of the converse is carried out in the following way. In the next section, we introduce a family of properties $\mathcal{P}_k$. As stated in Proposition 1, these properties characterize the boolean combinations of recognizable relations, almost length preserving relations and length comparing relations. Indeed, a relation is such a combination if and only if it has Property $\mathcal{P}_k$ for some k. This characterization is an intermediate step in the proof of the converse. In the last section, we finally prove that any left and right synchronous relation has Property $\mathcal{P}_k$ for some k.

5 An Intermediate Characterization

We now introduce a family $(\mathcal{P}_k)_{k\geq 0}$ of properties. It will be proved in Proposition 1 that these properties characterize the boolean combinations of recognizable relations, almost length preserving relations and length comparing relations.

Let k be a non-negative integer. We say that a relation $R \subseteq A^* \times B^*$ has the Property $\mathcal{P}_k$ if the following three properties are satisfied.

i) the relation $R \cap H_k$ is rational,
ii) there is a recognizable relation $K \subseteq A^* \times B^*$ such that $R \cap G_k = K \cap G_k$,
iii) there is a recognizable relation $K' \subseteq A^* \times B^*$ such that $R \cap J_k = K' \cap J_k$.

We say that a relation R has Property $\mathcal{P}$ if it has the Property $\mathcal{P}_k$ for some $k \geq 0$.

We will see that Property (i) is satisfied by any left or right synchronous relation but it is not satisfied by all rational relations as the following example shows.

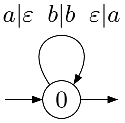


Fig. 4. Transducer for $((a,\varepsilon)+(b,b)+(\varepsilon,a))^*$

Example 5. Let A be the alphabet $\{a,b\}$. Let $R \subseteq A^* \times A^*$ be the set of pairs (u,v) such that $|u|_b = |v|_b$ where $|w|_b$ denotes the number of occurrences of the letter b in the word w. This relation is accepted by the transducer of Fig. 4. It can be easily shown that, for any integer $k \geq 0$, the intersection $R \cap H_k$ is not rational.

Lemma 1. *If a relation $R \subseteq A^* \times B^*$ has the Property $\mathcal{P}_k$ for some $k \geq 0$, then, it also the Property $\mathcal{P}_{k'}$ for any $k' \geq k$.*

Proof. Obviously Properties (ii) and (iii) are still satisfied for any $k' \geq k$ since $G_{k'} \subseteq G_k$ and $J_{k'} \subseteq J_k$. For Property (i), the intersection $R \cap H_{k+1}$ can be decomposed as the union $R_1 \cup R_2 \cup R_3$ where the relations R_1, R_2 and R_3 are defined as follows.

$$\begin{aligned} R_1 &= R \cap H_k, \\ R_2 &= K \cap \{(u,v) \mid |v| - |u| = -k-1\}, \\ R_3 &= K' \cap \{(u,v) \mid |v| - |u| = k+1\}. \end{aligned}$$

Relation R_1 is rational since R has Property $\mathcal{P}_k$. Relations R_2 and R_3 are both rational as intersections of a rational relation with a recognizable relation.

The following lemma is very easy to prove.

Lemma 2. *For any fixed k, the class of relations having Property $\mathcal{P}_k$ is closed under boolean operations.*

Corollary 1. *The class of relations having Property $\mathcal{P}$ is closed under boolean operations.*

Proof. A boolean combination of relations having Property $\mathcal{P}$ involves finitely many relations. By lemma 1, there is an integer k such that each relation involved in the boolean combination has Property $\mathcal{P}_k$. By the previous lemma, the boolean combination has also Property $\mathcal{P}_k$.

Proposition 1. *A relation $R \subseteq A^* \times B^*$ is a boolean combination of recognizable relations, almost length preserving relations and length comparing relations if and only if it has Property $\mathcal{P}$.*

Proof. It is easy to check that recognizable relations have Property $\mathcal{P}_k$ for any integer $k \geq 0$. A length comparing relation G_k or J_k has Property $\mathcal{P}_j$ whenever $j \geq k$. An almost length preserving relation R has Property $\mathcal{P}_k$ as soon as R is contained in H_k. By the previous corollary, the class of relations having Property $\mathcal{P}$ is closed under boolean operations. Therefore any boolean combination of recognizable relations, almost length preserving relations and length comparing relations has Property $\mathcal{P}$.

If a relation R has Property $\mathcal{P}_k$ for some $k \geq 0$, it can be decomposed as the union $R_1 \cup R_2 \cup R_3$ where the relations R_1, R_2 and R_3 are defined as follows.

$$\begin{aligned} R_1 &= R \cap G_k = K \cap G_k, \\ R_2 &= R \cap H_k \\ R_3 &= R \cap J_k = K' \cap J_k \end{aligned}$$

By definition, the relation R_2 is rational and almost length preserving. Both relations R_1 and R_3 are a intersection of a recognizable relation and a length comparing relation. It follows that R is a boolean combination of recognizable relations, almost length preserving relations and length comparing relations.

6 Proof of the Main Result

In this section we prove that any left and right synchronous relation has Property $\mathcal{P}_k$ for some integer k. This result combined with Proposition 1 completes the proof of Theorem 1. It shows that any left and right synchronous relation is a boolean combination of recognizable relations, almost length preserving relations and length comparing relations. We first recall some elementary results on congruences of finite index.

6.1 Congruences

Let A be an alphabet. A *congruence on* A^* is an equivalence relation on A^* which is compatible with concatenation. This means that if $x \sim x'$ and $y \sim y'$ then $xy \sim x'y'$. For an equivalence relation $\sim$, we denote by $[w]_\sim$ the equivalence class of a word w. If the relation $\sim$ is a congruence on A^*, the quotient $A^*/\!\sim$ is naturally endowed with a structure of monoid defined by $[u]_\sim \cdot [v]_\sim = [uv]_\sim$. In the proof we use the following results on congruences of finite index, that is, with finitely many classes. It is a kind of pumping lemma which is a direct consequence of the pigeon-hole principle.

Lemma 3. *Let $\sim$ be a congruence on A^* of finite index k. Any word w such that $|w| \geq k$ can be factorized $w = w_1w_2w_3$ such that $w_2 \neq \varepsilon$ and $w_1w_2^nw_3 \sim w$ for any $n \geq 0$.*

Proof. Let w be the word $a_1 \cdots a_m$ where $m \geq k$. Let us define a sequence $s_0, \ldots, s_m$ of classes of $\sim$. Let s_0 be the class $[\varepsilon]_\sim$ of the empty word and let $s_i = [a_1 \cdots a_i]_\sim$ for $1 \leq i \leq m$. By the the pigeon-hole principle, there are two integers $0 \leq i < j \leq k$ such that $s_i = s_j$. The words $w_1 = a_1 \cdots a_i$, $w_2 = a_{i+1} \cdots a_j$ and $w_3 = a_{j+1} \cdots a_m$ have the required properties. Indeed since $w_1w_2 \sim w_1$, one has $w_1w_2^n \sim w_1$ for any $n \geq 0$. It follows that $w_1w_2^nw_3 \sim w_1w_3 \sim w$ for any $n \geq 0$.

Lemma 4. *Let $\sim$ be a congruence on A^* of finite index. There are two integers $\ell \geq 0$ and $p \geq 1$ such that for any word $w \in A^*$ and any integer $m \geq \ell$, $w^{m+p} \sim w^m$ holds.*

Proof. Since the congruence $\sim$ has finite index, there are for each class s of $\sim$ two integers ℓ_s and p_s such that $s^{\ell_s+p_s} = s^{\ell_s}$. By multiplying this last equality by $s^{m-\ell_s}$, one gets that, for any $m \geq \ell_s$, $s^{m+p_s} = s^m$ holds. Let ℓ be the sum of the ℓ_s and p be the product of p_s where s ranges over the classes of $\sim$. This yields that $s^{m+p} = s^m$ holds for any class s of $\sim$ and for any integer $m \geq \ell$. Since $\sim$ is a congruence, $[w^n]_\sim$ is equal to $[w]_\sim^n$ and the result follows easily.

Let $w = w_1w_2w_3$ be a factorization of a word w such that $w_2 \neq \varepsilon$. For each integer m, let us define an integer n_m and two words $p(m)$ and $s(m)$ as follows. We set $n_m = 0$ if $m < |w_3|$ and $n_m = \lfloor (m - |w_3|)/|w_2| \rfloor$ otherwise. The integer n_m is thus the least non-negative integer n such that the length of $w_2^nw_3$ is greater than m. The words $p(m)$ and $s(m)$ are then the prefix and the suffix of respective length $n_m|w_2| + |w_1w_3| - m$ and m of the word $w_1w_2^{n_m}w_3$. By definition, the product $p(m)s(m)$ is equal to the word $w_1w_2^{n_m}w_3$.

Lemma 5. *Let $\sim$ be a congruence of finite index and let $w = w_1w_2w_3$ be a factorization such that $w_2 \neq \varepsilon$. Let $p(m)$ and $s(m)$ be the words defined as above for each integer m. There are two integers $\ell \geq 0$ and $q \geq 1$ such that both relations $p(m + q) = p(m)$ and $s(m + q) \sim s(m)$ hold for any $m \geq \ell$.*

Proof. For $m \geq |w_3|$, $n_{m+|w_2|}$ is equal to $n_m + 1$. It follows that the equality $p(m + |w_2|) = p(m)$ holds for each $m \geq |w_3|$.

By the previous lemma, there are two integers $\ell_0 \geq 0$ and $q_0 \geq 1$ such that for any word w_2 and any integer $n \geq \ell_0$, $w_2^{n+p_0} \sim w_2^n$. Let ℓ and q be defined by $\ell = k + \ell_0|w_2|$ and $q = q_0|w_2|$. We claim that $s(m+q) \sim s(m)$ holds for any $m \geq \ell$.

For $m \geq |w_3|$, each word $s(m)$ can be factorized $s'(m)w_3$ where the word $s'(m)$ is the suffix of length $m - |w_3|$ of the word $w_2^{n_m}$. For each $m \geq |w_3|$, the word $s'(m+|w_2|)$ is equal to $s'(m)w_2$ and $s'(m+i|w_2|)$ is then equal to $s'(m)w_2^i$. The following computation proves the claim.

$$\begin{aligned}
s(m+q) &\sim s'(m+q)w_3 \\
&\sim s'(m' + (\ell_0 + q_0)|w_2|)w_3 && \text{where } m' = m - \ell_0|w_2| \geq |w_3| \\
&\sim s'(m')w_2^{\ell_0+q_0}w_3 \\
&\sim s'(m')w_2^{\ell_0}w_3 && \text{by definition of } \ell_0 \text{ and } q_0 \\
&\sim s'(m' + \ell_0|w_2|)w_3 \\
&\sim s(m)
\end{aligned}$$

Since $\ell \geq |w_3|$ and q is a multiple of $|w_2|$, the equality $p(m+q) = p(m)$ also holds for any $m \geq \ell$. This completes the proof of the lemma.

6.2 Reductions

The following lemmas states that any left or right synchronous relation has Property (i) of the Property $\mathcal{P}_k$ for any $k \geq 0$. It is the first step in the proof that a left and right synchronous relation has Property $\mathcal{P}$.

Lemma 6. *For any left (respectively right) synchronous relation $R \subseteq A^* \times B^*$ and for any integer k, the relation $R \cap H_k$ is rational.*

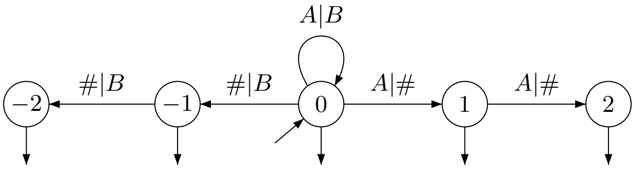


Fig. 5. Synchronous transducer accepting H_2

Proof. Since the class of left (respectively right) synchronous relations is closed under intersection (actually under all boolean operations), it suffices to prove that the relation H_k is left (respectively right) synchronous. We give a synchronous transducer that accepts the relation $\overrightarrow{H_k}$. The set Q of states is the set $\{-k, \ldots, 0, \ldots, k\}$ of integers between $-k$ and k. The initial state is 0 and all states are final. The set of transitions is the set E defined by

$$\begin{aligned} E = \{0 \xrightarrow{a|b} 0 \mid a \in A \text{ and } b \in B\} \\ \cup \{i \xrightarrow{a|\#} i+1 \mid a \in A \text{ and } 0 \leq i < k\} \\ \cup \{i \xrightarrow{\#|b} i-1 \mid b \in B \text{ and } -k < i \leq 0\}. \end{aligned}$$

The transducer for $k = 2$ is pictured in Fig. 5.

The following lemma allows us to only consider pairs of the form (uw, v) for some fixed word w.

Lemma 7. *Let k be an integer and $R \subseteq A^* \times B^*$ be a relation. There is a recognizable relation K such that $R \cap G_k = K \cap G_k$ if and only if for each word $w \in A^*$ of length k, there is a recognizable relation K_w such that*

$$\{(u, v) \mid (uw, v) \in R\} \cap G_0 = K_w \cap G_0.$$

Proof. Suppose first that there exists some recognizable relation K such that $R \cap G_k = K \cap G_k$. Let w be a fixed word of length k over A. Let K_w be the relation defined by

$$K_w = \{(u, v) \mid (uw, v) \in K\}.$$

If K is recognizable, then K_w is also recognizable. Furthermore, the required equality holds by definition of K_w.

Suppose conversely that for each word w of length k, there exists a recognizable relation K_w with the required property. Define the relation K as follows.

$$K = \bigcup_{|w|=k} K_w(w, \varepsilon).$$

Since each relation K_w is recognizable, each relation $K_w(w, \varepsilon)$ is also recognizable. The relation K is then recognizable as a finite union of recognizable relations. The equality $R \cap G_k = K \cap G_k$ follows from the definition of K.

The following lemma allows us to only consider pairs (u, v) where v belongs to some fixed class of a congruence of finite index.

Lemma 8. *Let $\approx$ be a congruence on B^* of finite index and $R \subseteq A^* \times B^*$ be a relation. There is a recognizable relation K such that $R \cap G_0 = K \cap G_0$ if and only if for each class S of $\approx$, there is a recognizable relation K_S such that*

$$R \cap (A^* \times S) \cap G_0 = K_S \cap G_0.$$

Proof. Since $\approx$ is a congruence of finite index, each class S is a rational language of B^* and $A^* \times S$ is a recognizable relation. If there is a recognizable relation K such that $R \cap G_0 = K \cap G_0$, then $K_S = K \cap (A^* \times S)$ is also recognizable relation which obviously has the required property.

If there is conversely a relation K_S for each class S, the relation $K = \bigcup_S K_S$ is still recognizable since there are finitely many S and it satisfies $R \cap G_0 = K \cap G_0$.

We now finally come to core of the proof which shows a left and right synchronous relation has Property $\mathcal{P}$. By Lemma 6, a left synchronous relation has Property (i) of the Property $\mathcal{P}_k$ for any $k \geq 0$. It remains to show that a left and right synchronous relation satisfies properties (ii) and (iii) of Property $\mathcal{P}_k$ for some $k \geq 0$. We only prove it for Property (ii). The result for Property (iii) follows by symmetry.

Let R be a left and right synchronous relation. There is then a transducer $\mathcal{L}$ accepting the relation $\overrightarrow{R}$ and a transducer $\mathcal{R}$ accepting $\overleftarrow{R}$. Without loss of generality, we can assume that the transducer $\mathcal{R}$ is deterministic.

We first define an equivalence relation $\sim$ on A^*. Two words w and w' satisfy $w \sim w'$ if for any pair (p, q) of states of $\mathcal{L}$, there is in $\mathcal{L}$ a path from p to q labeled by $(w, \#^{|w|})$ if and only if there is in $\mathcal{L}$ a path from p to q labeled by $(w', \#^{|w'|})$.

The equivalence relation $\sim$ has finite index since its number of classes is bounded by 2^{n^2} where n is the number of states of $\mathcal{L}$. It can be easily verified that the relation $\sim$ is actually a congruence on A^*. The quotient $A^*/\!\sim$ is then a finite monoid.

The following lemma captures the main property of the congruence $\sim$.

Lemma 9. *Let $u, w, w' \in A^*$ and $v \in B^*$ be words such that $|u| \geq |v|$ and $w \sim w'$. The pair (uw, v) belongs to R if and only if the pair (uw', v) belongs to R.*

Proof. Suppose that (uw, v) belongs to R. Since R is accepted by $\mathcal{L}$, there is an accepting path $i \xrightarrow{uw|v\#^k} f$ where $k = |uw| - |v|$ in the transducer $\mathcal{L}$. Since $|u| \geq |v|$, this path can be decomposed

$$i \xrightarrow{u|v\#^{|u|-|v|}} q \xrightarrow{w|\#^{|w|}} f.$$

Since $w \sim w'$, there is also a path $q \xrightarrow{w'|\#^{|w'|}} f$ which yields the accepting path

$$i \xrightarrow{u|v\#^{|u|-|v|}} q \xrightarrow{w'|\#^{|w'|}} f$$

labeled by the pair (uw', v). This shows that (uw', v) is also accepted by $\mathcal{L}$ and belongs then to R. By symmetry, if (uw', v) belongs to R, then (uw, v) belongs then to R.

From now on, the integer k is fixed to the index of the congruence $\sim$. We claim that the relation R has Property $\mathcal{P}_k$ for this integer k. By Lemma 7, it suffices to prove that for each word w of length k, there is a recognizable relation K_w such that $\{(u, v) \mid (uw, v) \in R\} \cap G_0 = K_w \cap G_0$. From now on, we fix a word w of length k.

By Lemma 3, the word w can be factorized $w = w_1 w_2 w_3$ such that $w_2 \neq \varepsilon$ and $w_1 w_2^n w_3 \sim w$ for any $n \geq 0$. Given this factorization $w = w_1 w_2 w_3$ with $w_2 \neq \varepsilon$, the words $p(m)$ and $s(m)$ are defined as in Sect. 6.1. Lemma 5 gives then two integers ℓ and q such that both relations $p(m+q) = p(m)$ and $s(m+q) \sim s(m)$ hold for any $m \geq \ell$.

We introduce a congruence $\approx$ on B^* as the intersection of two congruences $\approx_1$ and $\approx_2$ that we now define.

The congruence $\approx_1$ is defined as follows. Two words v and v' satisfy $v \approx_1 v'$ if $v = v'$ or $|v| \geq \ell$, $|v'| \geq \ell$ and $|v| \equiv |v'| \mod q$. Note that if the length of v is less than k, the class of v is reduced to the singleton $\{v\}$. This is easy to verify that the relation $\approx_1$ is indeed a congruence of finite index. Its number of classes is equal to $n + q$ where n is the number of words over B of length less than k. This number n is equal to k if $|B| = 1$ and to $(|B|^k - 1)/(|B| - 1)$ otherwise.

The congruence $\approx_2$ is defined as follows. Two words v and v' satisfy $v \approx_2 v'$ if for any pair (p, q) of states of $\mathcal{R}$ and any class S of $\sim$, there is a word $u \in S$ such that $|u| = |v|$ and there is in $\mathcal{R}$ a path from p to q labeled by (u, v) if and only if there is a word $u' \in S$ such that $|u'| = |v'|$ and there is in $\mathcal{R}$ a path from p to q labeled by (u', v'). The equivalence relation $\approx_2$ has finite index since its number of classes is bounded by $2^{n^2 k}$ where n is the number of states of $\mathcal{R}$ and k is the index of $\sim$. It can be easily verified that the relation $\approx_2$ is actually a congruence on B^*.

The relation $\approx$ is finally defined by $v \approx v'$ if and only if $v \approx_1 v'$ and $v \approx_2 v'$ for any $v, v' \in B^*$. As an intersection of two congruences of finite index, the relation $\approx$ is also a congruence of finite index.

By Lemma 8, it suffices to prove that for each class S of $\approx$, there is a recognizable relation K_S such that

$$\{(u, v) \mid (uw, v) \in R\} \cap (A^* \times S) \cap G_0 = K_S \cap G_0.$$

From now on, we fix a class S of $\approx$. We claim that there is a recognizable relation of the form $T \times S$ which has the required property. Note that the result is trivial if S contains a single word v. In that case, the recognizable K_S can be chosen equal to $T \times \{v\}$ where T is the set $\{u \mid (uw, v) \in R\}$ which is obviously rational. Therefore, we may assume that S is not a singleton class. By definition of $\approx_1$, any two words v and v' in S satisfy $|v| \geq \ell$, $|v'| \geq \ell$ and $|v| \equiv |v'| \mod q$.

In order to prove the claim, we show that for any $v, v' \in S$, the two sets $\{u \mid (uw, v) \in R\}$ and $\{u \mid (uw, v') \in R\}$ are equal. It suffices then to take K_S equal to $\{u \mid (uw, v) \in R\} \times S$ which is actually independent of the choice of v in S.

By definition of the factorization $w = w_1 w_2 w_3$, one has $w_1 w_2^n w_2 \sim w$ for any $n \geq 0$. By Lemma 9, for any pair (u, v) such that $|u| \geq |v|$, the pair (uw, v) belongs to R is and only if the pair $(uw_1 w_2^n w_3, v)$ belongs to R. The idea is now to choose n large enough such that the length of $w_2^n w_3$ is greater than the length of v. Recall that we have defined in Sect. 6.1 two words $p(m)$ and $s(m)$ for each integer $m \geq 0$. Recall also that the length of $s(m)$ is equal to m and that the product $p(m)s(m)$ is equal to $w_1 w_2^n w_3$ for some $n \geq 0$.

Furthermore, the two integers ℓ and q given by Lemma 5 are such that $p(m + q) = p(m)$ and $s(m + q) \sim s(m)$ hold for any $m \geq \ell$. For each word $v \in B^*$, let us denote by $p(v)$ and $s(v)$ the words $p(|v|)$ and $s(|v|)$. The two words v and $s(v)$ have the same length. Since two words v and v' in S satisfy $|v| \geq \ell$, $|v'| \geq \ell$

and $|v| \equiv |v'| \mod q$, both relations $p(v) = p(v')$ and $s(v) \sim s(v')$ hold for any $v, v' \in S$.

Since $p(v)s(v)$ is equal to $w_1 w_2^n w_3$ for some $n \geq 0$, the pair (uw, v) belongs to R if and only the pair $(up(v)s(v), v)$ belongs to R. This is true if and only if there is a path labeled by this pair in the transducer $\mathcal{R}$. Suppose that such a path exists. Since v and $s(v)$ have the same length, this path can be factorized

$$i \xrightarrow{u|\#^{|u|}} q \xrightarrow{p(v)|\#^{|p(v)|}} q' \xrightarrow{s(v)|v} f$$

where i is an initial state and f is a final state.

Since $p(v) = p(v')$ for any $v' \in S$, there is also a path $q \xrightarrow{p(v')|\#^{|p(v')|}} q'$ in the transducer $\mathcal{R}$. Furthermore, since $v \approx_2 v'$, there is also a path $q' \xrightarrow{u'|v'} f$ in $\mathcal{R}$ where $|u'| = |v'|$ and $u' \sim s(v)$. This shows that $(up(v')u', v')$ belongs also to R. By Lemma 9, $(up(v')s(v'), v')$ belongs also to R since $|u| \geq |v|$ and $u' \sim s(v) \sim s(v')$.

We have shown that if the pair $(up(v)s(v), v)$ belongs to R, then the pair $(up(v')s(v'), v')$ belongs also to R. This proves the claim and completes the proof that any left and right synchronous relation has Property $\mathcal{P}$.

References

1. Eilenberg, S.: Automata, Languages and Machines, vol. A. Academic Press, New York (1972)
2. Elgot, C.C., Mezei, J.E.: On relations defined by generalized finite automata. IBM Journal Res. and Dev. 9, 47–68 (1965)
3. Frougny, C., Sakarovitch, J.: Synchronized relations of finite words. Theoret. Comput. Sci. 108, 45–82 (1993)
4. Khoussainov, B., Rubin, S.: Automatic structures: Overview and future directions. Journal of Automata, Languages and Combinatorics 8(2), 287–301 (2003)
5. Sakarovitch, J.: Éléments de théorie des automates. Vuibert (2003)

An Extension of the Lyndon Schützenberger Result to Pseudoperiodic Words*

Elena Czeizler**, Eugen Czeizler**, Lila Kari, and Shinnosuke Seki

Department of Computer Science, University of Western Ontario,
London, Ontario, N6A 5B7, Canada
{elena,eugen}.czeizler@abo.fi, {lila,sseki}@csd.uwo.ca

Abstract. One of the particularities of information encoded as DNA strands is that a string u contains basically the same information as its Watson-Crick complement, denoted here as $\theta(u)$. Thus, any expression consisting of repetitions of u and $\theta(u)$ can be considered in some sense periodic. In this paper we give a generalization of Lyndon and Schützenberger's classical result about equations of the form $u^l = v^n w^m$, to cases where both sides involve repetitions of words as well as their complements. Our main results show that, for such extended equations, if $l \geq 5, n, m \geq 3$, then all three words involved can be expressed in terms of a common word t and its complement $\theta(t)$. Moreover, if $l \geq 5$, then $n = m = 3$ is an optimal bound. We also obtain a complete characterization of all possible overlaps between two expressions that involve only some word u and its complement $\theta(u)$.

1 Introduction

This paper is a theoretical study of *pseudoperiodic words*, notion motivated by the properties of information encoded as DNA strands for DNA computing purposes. Informally, a word is pseudoperiodic if it consists of repeated occurrences of another word and/or the image of that word under an antimorphic involution. The notion of antimorphic involution is the mathematical formalization of the Watson-Crick complementarity of DNA single strands, as detailed below.

DNA, in its single-stranded form, is a linear chain made up of four different types of units, called nucleotides, and can thus be viewed to a first approximation as a word over the four-letter alphabet $\{A, C, G, T\}$. A DNA single strand has an orientation, with one end known as the 5' end, and the other as the 3' end, based on their chemical properties. By convention, a word over the DNA alphabet represents the corresponding DNA single strand in the 5'-3' orientation. Another crucial feature is the Watson-Crick (WK) complementarity: A is complementary to T, and G is to C. Two complementary DNA single strands with opposite

* This research was supported by The Natural Sciences and Engineering Council of Canada Discovery Grant and Canada Research Chair Award to L.K.

** Current affiliation: Department of IT, Åbo Akademi University, Turku 20520, Finland.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 183–194, 2009.

orientation will bind to each other by bonds between their individual bases to form a helical DNA double strand. The Watson-Crick complementarity operation is a fundamental bio-operation in DNA Computing experiments [1]. In this paper we investigate the consequences of Watson-Crick complementarity on the notion of periodicity of words.

Periodicity properties of words are among the main theoretical tools used in pattern-matching algorithms, see e.g. [2] and [3]. Recall that a word u is called *periodic* if there exists another word v, shorter than u, such that u is a prefix of v^i for some $i \geq 2$. Moreover, the way in which a word can be decomposed, and whether two words are powers of a common word are two questions which have been widely investigated in language theory, see, e.g., [4] and [5]. However, when dealing with DNA strands, note that a string u encodes the same information as its complement, $\theta(u)$, where θ denotes the WK complementarity function or its mathematical formalization as an antimorphic involution. In this context, e.g., the word $u^m\theta(u)^n$ can be considered periodic, since it consists of repetitions of the same information unit. (Other generalizations of the notion of periodicity include, e.g., the "weak periodicity" of [6] whereby a word is called *weakly periodic* if it consists of repetitions of words with the same Parikh vector. This type of period was called *abelian period* in [7].) In [8] the Fine and Wilf Theorem – one of the fundamental periodicity results on words, see e.g. [4] and [5] – was extended to deal with expressions involving both a word and its image under an antimorphic involution.

Here we extend another central periodicity result, due to Lyndon and Schützenberger, [9]. (See also [10] and Chap. 5 from [5] for some shorter proofs and [11] and [12] for some other generalizations.) The original result states that, if the concatenation of two periodic words v^n and w^m can be expressed in terms of a third period u, i.e., $u^l = v^n w^m$, for some $n, m, l \geq 2$, then all three words u, v, and w can be expressed in terms of a common word t, i.e., $u, v, w \in \{t\}^*$.

In our generalization, we consider repetitions involving both a word and its image under θ, i.e., the equation $\alpha(u, \theta(u)) = \beta(v, \theta(v)) \cdot \gamma(w, \theta(w))$ where $\alpha(u, \theta(u)) \in \{u, \theta(u)\}^l$, $\beta(v, \theta(v)) \in \{v, \theta(v)\}^n$, and $\gamma(w, \theta(w)) \in \{w, \theta(w)\}^m$ with $l, n, m \geq 2$. A conclusion of our main results is that, whenever $l \geq 5$, $n, m \geq 3$ we have $u, v, w \in \{t, \theta(t)\}^+$ for some word t, i.e., all three words can be expressed using a common word t and its image $\theta(t)$. Moreover, we provide examples showing that, for any $l \geq 5$, $n = m = 3$ is an optimal bound. In the case when $l = 3$ or $l = 4$, the problem of finding optimal bounds remains open. Our proofs are not generalizations of the methods used in the classical case, since one of the main properties used therein, i.e., the fact that the conjugate of a primitive word is still primitive, cannot be used here.

In our search for these bounds, we also obtain a characterization of all possible overlaps of two expressions $\alpha(v, \theta(v)), \beta(v, \theta(v)) \in \{v, \theta(v)\}^+$. In particular, we show that, contrary to the classical case (when the two expressions involve only a word v, but not its image under θ), the equality $\alpha(v, \theta(v)) \cdot x = y \cdot \beta(v, \theta(v))$ with x and y shorter than v, does not always force a decomposition of v of the form $v \in \{t, \theta(t)\}^+$ for some word t.

The paper is organized as follows. In Sect. 2, we fix our terminology and recall some known results. In Sect. 3, we provide the characterization of all possible overlaps of the form $\alpha(v, \theta(v)) \cdot x = y \cdot \beta(v, \theta(v))$ with $\alpha(v, \theta(v)), \beta(v, \theta(v)) \in \{v, \theta(v)\}^+$ and x, y shorter than v. Finally, in Sect. 4 we provide our extension of Lyndon and Schützenberger's result.

2 Preliminaries

Let Σ be a finite alphabet. We denote by Σ^* the set of all finite words over Σ, by ϵ the empty word, and by Σ^+ the set of all nonempty finite words. The catenation of two words $u, v \in \Sigma^*$ is denoted by either uv or $u \cdot v$. The *length* of a word $w \in \Sigma^*$, denoted by $|w|$, is the number of letters occurring in it. We say that u is a *factor* (a *prefix*, a *suffix*, resp.) of v, if $v = t_1ut_2$ ($v = ut_2$, $v = t_1u$, resp.) for some $t_1, t_2 \in \Sigma^*$. We denote by $\mathrm{Pref}(v)$ (resp. $\mathrm{Suff}(v)$) the set of all prefixes (resp. suffixes) of the word v. We say that two words u and v overlap if $ux = yv$ for some $x, y \in \Sigma^*$ with $|x| < |v|$. An integer $p \geq 1$ is a *period* of a word $w = a_1 \ldots a_n$, with $a_i \in \Sigma$ for all $1 \leq i \leq n$, if $a_i = a_{i+p}$ for all $1 \leq i \leq n - p$.

A word $w \in \Sigma^+$ is called *primitive* if it cannot be written as a power of another word; that is, if $w = u^n$ then $n = 1$ and $w = u$. For a word $w \in \Sigma^+$, the shortest $u \in \Sigma^+$ such that $w = u^n$ for some $n \geq 1$ is called the *primitive root* of the word w and is denoted by $\rho(w)$. The following is a well-known property of primitive words, see, e.g., [4], [5].

Proposition 1. *Let $u \in \Sigma^+$ be a primitive word. If $u^2 = xuy$, then either $x = \epsilon$ or $y = \epsilon$.*

A mapping $\theta : \Sigma^* \to \Sigma^*$ is called an *antimorphism* if for any words $u, v \in \Sigma^*$, $\theta(uv) = \theta(v)\theta(u)$. Moreover, a mapping $\theta : \Sigma^* \to \Sigma^*$ is called an *involution* if, for all words $u \in \Sigma^*$, $\theta(\theta(u)) = u$. An antimorphic involution is a mathematical formalization of the WK complementarity of DNA single strands. Throughout this paper we will assume that θ is an antimorphic involution on a given alphabet Σ. A word $w \in \Sigma^*$ is called a *θ-palindrome*, or a *pseudopalindrome* if θ is not specified, if $w = \theta(w)$ (see [13] and [14]).

The notions of periodic and primitive words were extended in [8] in the following way. A word $w \in \Sigma^+$ is θ-periodic if $w = w_1 \ldots w_k$ for some $k \geq 2$ and words $t, w_1, \ldots, w_k \in \Sigma^+$ such that $w_i \in \{t, \theta(t)\}$ for all $1 \leq i \leq k$. Following [14], in less precise terms, a word which is θ-periodic with respect to a given but unspecified involutory morphism θ will be also called *pseudoperiodic*. The word t in the definition of a θ-periodic word w is called a θ-period of w. We call a word $w \in \Sigma^+$ *θ-primitive* if it is not θ-periodic. The set of θ-primitive words is strictly included in the set of primitive ones, see [8]; for instance, if we take $a \neq b$ and $\theta(a) = b$, $\theta(b) = a$, then the word ab is primitive, but not θ-primitive. We define *the θ-primitive root of* w, denoted by $\rho_\theta(w)$, as the shortest word t such that $w = w_1 \ldots w_k$ for some $k \geq 1$, $w_i \in \{t, \theta(t)\}$ for all $1 \leq i \leq k$, and $w_1 = t$. Note that if w is θ-primitive, then $\rho_\theta(w) = w$.

We say that two words u and v *commute* if $uv = vu$. We can characterize the commutation of two words in terms of primitive roots, see, e.g., [4], [5].

Theorem 1. *For $u, v \in \Sigma^*$, the following conditions are equivalent: i) u and v commute; ii) u and v satisfy a nontrivial relation, i.e., a nontrivial equation over two variables without constants; iii) u and v have the same primitive root.*

Two words u and v are said to be *conjugate* if there exist words x and y such that $u = xy$ and $v = yx$. In other words, v can be obtained via a cyclic permutation of u. The next known result, see, e.g., [4], [5], characterizes the conjugacy of two words.

Theorem 2. *Let $u, v \in \Sigma^+$. Then, the following conditions are equivalent: i) u and v are conjugate; ii) there exists a word z such that $uz = zv$; moreover, this holds if and only if $u = pq$, $v = qp$, and $z = (pq)^i p$, for some $p, q \in \Sigma^*$ and $i \geq 0$; iii) the primitive roots of u and v are conjugate.*

The following periodicity result is due to Lyndon and Schützenberger, [9].

Theorem 3. *If words u, v, w satisfy the relation $u^l = v^n w^m$ for some positive integers $l, n, m \geq 2$, then they are all powers of a common word, i.e., there exists a word t such that $u, v, w \in \{t\}^*$.*

The Fine and Wilf theorem, in its form for words, see [4], [5], illustrates another fundamental periodicity property. It states that if two words $u, v \in \Sigma^*$, with $n = |u|$, $m = |v|$, $d = \gcd(n, m)$, are such that if two powers u^i and v^j have a common prefix of length at least $n+m-d$, then u and v are powers of a common word, where $\gcd(n, m)$ denotes, as usual, the *greatest common divisor of n and m*. Moreover, the bound $n + m - d$ is optimal. The original result of Fine and Wilf, [15], was formulated for sequences of real numbers.

This theorem was extended in [8] for the case when instead of dealing with powers of two words u^i and v^j, we look at expressions over $\{u, \theta(u)\}$ and $\{v, \theta(v)\}$, respectively. Its weaker version, which will be very useful, is presented as well.

Theorem 4 ([8]). *Let $u, v \in \Sigma^+$ be two distinct words with $|u| > |v|$. If there exist two expressions $\alpha(u, \theta(u)) \in u\{u, \theta(u)\}^*$ and $\beta(v, \theta(v)) \in v\{v, \theta(v)\}^*$ having a common prefix of length at least $2|u|+|v|-gcd(|u|, |v|)$, then $\rho_\theta(u) = \rho_\theta(v)$. Moreover, the bound $2|u| + |v| - gcd(|u|, |v|)$ is optimal.*

Theorem 5 ([8]). *Let $u, v \in \Sigma^+$, $\alpha(u, \theta(u)) \in u\{u, \theta(u)\}^*$, and $\beta(v, \theta(v)) \in v\{v, \theta(v)\}^*$ such that $\alpha(u, \theta(u)) = \beta(v, \theta(v))$. Then $\rho_\theta(u) = \rho_\theta(v)$.*

The next two results, also from [8], will be very useful in our considerations.

Lemma 1 ([8]). *For $u, v \in \Sigma^*$, if $uv = \theta(uv)$ and $vu = \theta(vu)$, then there exists a word $t \in \Sigma^+$ such that $u, v \in \{t, \theta(t)\}^*$.*

Lemma 2 ([8]). *Let $v \in \Sigma^+$ be a θ-primitive word. Then, $\theta(v)vx = yv\theta(v)$ for some words $x, y \in \Sigma^*$ with $|x|, |y| < |v|$, if and only if $v = \theta(v)$ and $x = y = \epsilon$. Similarly, $v\theta(v)v = xv^2y$ for some $x, y \in \Sigma^*$ if and only if $v = \theta(v)$ and either $x = \epsilon$ or $y = \epsilon$.*

The following result will prove very useful in our future considerations.

Lemma 3. *Let $u \in \Sigma^+$ such that $u = xz = zy$ for some $x, y, z \in \Sigma^+$ with $x = \theta(x)$ and $y = \theta(y)$. Then $x, y, z, u \in \{t, \theta(t)\}^*$ for some $t \in \Sigma^+$.*

Proof. The equation $u = xz = zy$ implies that $x = pq$, $y = qp$, and $z = (pq)^j p$ for some $p, q \in \Sigma^*$ and $j \geq 0$. Since $x = \theta(x)$ and $y = \theta(y)$, we have $pq = \theta(pq)$ and $qp = \theta(qp)$. Then, Lemma 1 implies that there exists a word $t \in \Sigma^+$ such that $p, q \in \{t, \theta(t)\}^*$. □

3 Overlaps between θ-Primitive Words

It is well known that a primitive word v cannot occur nontrivially inside v^2, see Proposition 1. Thus, two expressions v^i and v^j, with $i, j \geq 1$, cannot overlap nontrivially on a sequence longer than $|v|$. A natural question is whether we can have some nontrivial overlaps between two expressions $\alpha(v, \theta(v)), \beta(v, \theta(v)) \in \{v, \theta(v)\}^+$, when $v \in \Sigma^+$ is a θ-primitive word. In this section we completely characterize all such nontrivial overlaps, and, moreover, in each case we also give the set of all solutions of the corresponding equation.

We begin our analysis by giving two intermediate results.

Theorem 6. *Let $v \in \Sigma^+$ be a θ-primitive word and $\alpha(v, \theta(v)), \beta(v, \theta(v)) \in \{v, \theta(v)\}^+$ such that $\alpha(v, \theta(v)) \cdot x = y \cdot \beta(v, \theta(v))$, with $x, y \in \Sigma^+$, $|x|, |y| < |v|$. Then, v^2 and $\theta(v)^2$ cannot occur simultaneously neither in $\alpha(v, \theta(v))$ nor in $\beta(v, \theta(v))$.*

Proof. Suppose that both v^2 and $\theta(v)^2$ occur in $\alpha(v, \theta(v))$; the case when they both occur in $\beta(v, \theta(v))$ is symmetric. Moreover, since θ is an involution, we can suppose without loss of generality that v^2 occurs before $\theta(v)^2$, thus implying that $v^2\theta(v)$ is a factor in $\alpha(v, \theta(v))$. Since v (resp. $\theta(v)$) is primitive, the border between any two consecutive v's (resp. $\theta(v)$'s) falls inside a $\theta(v)$ (resp. v), see Fig. 1. Thus, $v^2\theta(v)$ overlaps either with $\theta(v)v^2$ or with $\theta(v)v\theta(v)$ or with $\theta(v)^2v$.

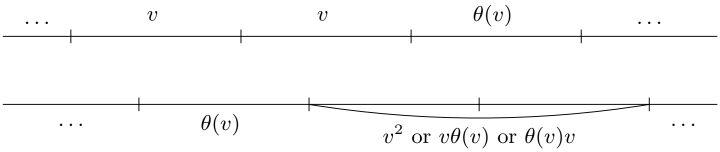


Fig. 1. The case when $v^2\theta(v)$ is a factor in $\alpha(v, \theta(v))$

In all three cases the nontrivial overlap between $v\theta(v)$ and $\theta(v)v$ contradicts the θ-primitivity of v, see Lemma 2. □

Theorem 7. *For a θ-primitive word $v \in \Sigma^+$, let $\alpha(v, \theta(v)), \beta(v, \theta(v)) \in \{v, \theta(v)\}^+$ such that $\alpha(v, \theta(v)) \cdot x = y \cdot \beta(v, \theta(v))$ for some $x, y \in \Sigma^+$ with $|x|, |y| < |v|$. Then, for any $i \geq 1$, neither $v\theta(v)^i v$ nor $\theta(v)v^i\theta(v)$ can occur either in $\alpha(v, \theta(v))$ or in $\beta(v, \theta(v))$.*

Proof. Suppose that $v\theta(v)^i v$ occurs in $\alpha(v,\theta(v))$ for some $i \geq 1$. We assumed that $x, y \in \Sigma^+$ and $|x|, |y| < |v|$ so that the factor $v\theta(v)^i v$ contains as a proper factor $\gamma(v,\theta(v)) \in \{v,\theta(v)\}^{i+1}$, i.e., there exist some $p, q \in \Sigma^+$ such that $v\theta(v)^i v = p\gamma(v,\theta(v))q$. Due to Lemma 2 and $\theta(v)$ being primitive, $\gamma(v,\theta(v)) = v^{i+1}$. Now we have $v\theta(v)^i v = pv^{i+1}q$ and hence $v\theta(v)v = pv^2q$. However, this contradicts Lemma 2. All the other cases can be proved similarly. □

As an immediate consequence of the previous two theorems, for a given θ-primitive word v, if $\alpha(v,\theta(v))\cdot x = y\cdot\beta(v,\theta(v))$ with $x, y \in \Sigma^+$, $|x|, |y| < |v|$, then $\alpha(v,\theta(v))$ and $\beta(v,\theta(v))$ can be only of the following types v^i, $v^i\theta(v)$, $v\theta(v)^i$, $\theta(v)^i$, $\theta(v)^i v$, or $\theta(v)v^i$ for some $i \geq 1$. The next result refines this characterization further. However, due to space limitations, we omit its proof.

Theorem 8. *Let $v \in \Sigma^+$ be a θ-primitive word. Then, the only possible proper overlaps of the form $\alpha(v,\theta(v)) \cdot x = y \cdot \beta(v,\theta(v))$ with $\alpha(v,\theta(v)), \beta(v,\theta(v)) \in \{v,\theta(v)\}^+$, $x, y \in \Sigma^+$ and $|x|, |y| < |v|$ are given in Table 1 (modulo a substitution of v by $\theta(v)$) together with the characterization of their sets of solutions.*

Table 1. Characterization of possible proper overlaps of the form $\alpha(v,\theta(v)) \cdot x = y \cdot \beta(v,\theta(v))$. For the last three equations, $n \geq 0$, $m \geq 1$, $r, t \in \Sigma^+$ such that $r = \theta(r)$, $t = \theta(t)$, and rt is primitive. Note that the 4th and 5th equations are the same up to the antimorphic involution θ.

Equation	Solution
$v^i x = y\theta(v)^i, i \geq 1$	$v = yp, x = \theta(y), p = \theta(p)$, and whenever $i \geq 2$, $y = \theta(y)$
$vx = yv$	$v = (pq)^{j+1}p$, $x = qp$, $y = pq$ for some $p, q \in \Sigma^+$, $j \geq 0$
$v\theta(v)x = yv\theta(v)$,	$v = (pq)^{j+1}p$, $x = \theta(pq)$, $y = pq$, with $j \geq 0$, $qp = \theta(qp)$
$v^{i+1}x = y\theta(v)^i v$, $i \geq 1$	$v = r(tr)^{n+m}r(tr)^n$, $x = (tr)^m r(tr)^n$, $y = r(tr)^{n+m}$
$v\theta(v)^i x = yv^{i+1}$, $i \geq 1$	$v = (rt)^n r(rt)^{m+n} r$, $y = (rt)^n r(rt)^m$, $x = (rt)^{m+n} r$
$v\theta(v)^i x = yv^i\theta(v)$, $i \geq 2$	$v = (rt)^n r(rt)^{m+n} r$, $y = (rt)^n r(rt)^m$, $x = (tr)^m r(tr)^n$

4 An Extension of Lyndon and Schützenberger's Result

For $u, v, w \in \Sigma^+$ and $\ell, n, m \geq 2$, let us consider some expressions $\alpha(u,\theta(u)) \in \{u,\theta(u)\}^\ell$, $\beta(v,\theta(v)) \in \{v,\theta(v)\}^n$, and $\gamma(w,\theta(w)) \in \{w,\theta(w)\}^m$ satisfying the equation $\alpha(u,\theta(u)) = \beta(v,\theta(v)) \cdot \gamma(w,\theta(w))$. We say that the triple (l, n, m) *imposes θ-periodicity on u, v, w*, (or shortly, *imposes θ-periodicity*), if the equation $\alpha(u,\theta(u)) = \beta(v,\theta(v)) \cdot \gamma(w,\theta(w))$ admits only solutions of the form $u, v, w \in \{t,\theta(t)\}^*$ for some word $t \in \Sigma^+$. Note that, if (l, n, m) imposes θ-periodicity, then so does (l, m, n), and vice versa.

In the classical case of the equation $u^l = v^n w^m$, Lyndon and Schützenberger's result (Theorem 3) states that any triple (l, n, m) with $l, n, m \geq 2$ imposes classical periodicity on u, v, w, with the same period. In this section we extend this result by considering the more general equation $\alpha(u,\theta(u)) = \beta(v,\theta(v)) \cdot$

Table 2. Result summary for the extended Lyndon-Schützenberger equation

l	n	m	θ-periodicity	
≥ 6	≥ 3	≥ 3	YES	Theorem 9
5	≥ 5	≥ 5	YES	Theorem 10
5	4	≥ 4	YES	Theorem 12
5	3	≥ 3	YES	Theorem 13
≥ 3	2	≥ 1	NO	Examples 1 and 2

$\gamma(w, \theta(w))$. Note that the fact that a certain triple (l, n, m) imposes θ-periodicity does not imply that (l', n', m') imposes θ-periodicity for $l' > l$ or $n' > n$ or $m' > m$.

The results of this section are summarized in Table 2. Overall, combining all the results from this section we obtain that $l \geq 5$, $n \geq 3$, $m \geq 3$ imposes θ-periodicity on u, v, and w. In contrast, for $l \geq 3$, once either $n = 2$ or $m = 2$, (l, n, m) does not always impose θ-periodicity, see Examples 1 and 2. Therefore, when $l \geq 5$, $(l, 3, 3)$ is the optimal bound. In the case when $l = 2$, $l = 3$, or $l = 4$, the problem of finding optimal bounds is still open.

Example 1. Let $\Sigma = \{a, b\}$ and $\theta : \Sigma^* \to \Sigma^*$ be the mirror image defined as $\theta(a) = a$, $\theta(b) = b$, and $\theta(w_1 \dots w_n) = w_n \dots w_1$, where $w_i \in \{a, b\}$ for all $1 \leq i \leq n$. Take now $u = a^k b^2 a^{2k}$, $v = \theta(u)^l a^{2k} b^2 = (a^{2k} b^2 a^k)^l a^{2k} b^2$, and $w = a^2$, for some $k, l \geq 1$. Then, although $\theta(u)^{l+1} u^{l+1} = v^2 w^k$, there is no word $t \in \Sigma^+$ with $u, v, w \in \{t, \theta(t)\}^+$, i.e., for any $k, l \geq 1$, the triple of numerical parameters $(2l + 2, 2, k)$ is not enough to impose θ-periodicity.

Example 2. Consider again $\Sigma = \{a, b\}$ and $\theta : \Sigma^* \to \Sigma^*$ be the mirror image defined in the previous example and take $u = b^2(aba)^k$, $v = u^l b = (b^2(aba)^k)^l b$, and $w = aba$ for some $k, l \geq 1$. Then, although $u^{2l+1} = v\theta(v)w^k$, there is no word $t \in \Sigma^+$ with $u, v, w \in \{t, \theta(t)\}^+$, i.e., for any $k, l \geq 1$, $(2l + 1, 2, k)$ is not enough to impose θ-periodicity.

The next two results analyze the cases when we have triples (l, n, m) with $l \geq 6$ and $n, m \geq 3$ and respectively $(5, n, m)$ with $n, m \geq 5$. The proof of Theorem 10 employs similar techniques as in Theorem 9, so we omit it here.

Theorem 9. *Let $u, v, w \in \Sigma^+$, $n, m \geq 3$, $l \geq 6$, $u_i \in \{u, \theta(u)\}$ for $1 \leq i \leq l$, $v_j \in \{v, \theta(v)\}$ for $1 \leq j \leq n$, and $w_k \in \{w, \theta(w)\}$ for $1 \leq k \leq m$. If $u_1 \dots u_l = v_1 \dots v_n \, w_1 \dots w_m$, then there exists a word $t \in \Sigma^+$ such that $u, v, w \in \{t, \theta(t)\}^+$.*

Proof. Let us suppose that $|v_1 \dots v_n| \geq |w_1 \dots w_m|$; the other case is symmetric and can be solved similarly. Then, $|v_1 \dots v_n| \geq \frac{1}{2}|u_1 \dots u_l| \geq 3|u|$, since $l \geq 6$. Since $n \geq 3$, this means that $u_1 \dots u_l$ and $v_1 \dots v_n$ share a common prefix of length larger than both $3|u|$ and $3|v|$. Thus, we can apply Theorem 4, to obtain that $u, v \in \{t, \theta(t)\}^+$ for some θ-primitive word $t \in \Sigma^+$. Moreover, since $u_1 \dots u_l = v_1 \dots v_n \, w_1 \dots w_m$, this implies $w_1 \dots w_m \in \{t, \theta(t)\}^*$. Since t is θ-primitive, Theorem 5 implies that also $w \in \{t, \theta(t)\}^+$. □

Theorem 10. *Let $u, v, w \in \Sigma^+$, $n, m \geq 5$, $u_i \in \{u, \theta(u)\}$ for $1 \leq i \leq 5$, $v_j \in \{v, \theta(v)\}$ for $1 \leq j \leq n$, and $w_k \in \{w, \theta(w)\}$ for $1 \leq k \leq m$. If $u_1u_2u_3u_4u_5 = v_1 \dots v_n\, w_1 \dots w_m$, then there exists a word $t \in \Sigma^+$ such that $u, v, w \in \{t, \theta(t)\}^+$.*

The triple $(5, n, m)$ also turns out to impose θ-periodicity for any $n \geq 4$ and $m \geq 7$.

Theorem 11. *Let $u, v, w \in \Sigma^+$, $n \geq 4$, $m \geq 7$, $u_i \in \{u, \theta(u)\}$ for $1 \leq i \leq 5$, $v_j \in \{v, \theta(v)\}$ for $1 \leq j \leq n$, and $w_k \in \{w, \theta(w)\}$ for $1 \leq k \leq m$. If $u_1u_2u_3u_4u_5 = v_1 \dots v_n\, w_1 \dots w_m$, then there exists a word $t \in \Sigma^+$ such that $u, v, w \in \{t, \theta(t)\}^+$.*

Proof. Unless the border between v_n and w_1 falls inside u_3, Theorem 4 concludes the existence of such t. So, assume that the border falls inside u_3. If the border between u_2 and u_3 falls inside some v_i except v_n, then, due to Theorem 4, we obtain $u, v, w \in \{t, \theta(t)\}^+$ for some $t \in \Sigma^+$. Otherwise, we have that $(n-1)|v| < 2|u|$, which means $|v| < \frac{2}{n-1}|u| \leq \frac{2}{3}|u|$. Similarly, if the border between u_3 and u_4 does not fall inside w_1, we reach the existence of such t; otherwise $|w| < \frac{2}{m-1}|u| \leq \frac{1}{3}|u|$. Under the condition that v_n and w_1 straddle these respective borders, the equation cannot hold because v and w are too short. □

We already know from Example 2 that for any $m \geq 1$, the triple $(5, 2, m)$ is not enough to impose θ-periodicity. So, we investigate next what would be the optimal bound for the extension of the Lyndon and Schützenberger result when the first parameter is 5. Note that, without loss of generality, we can assume $n \leq m$. Then, due to Theorem 10, all we have to investigate are the cases $(5, 3, m)$ for $m \geq 3$ and $(5, 4, m)$ for $m \geq 4$. The next intermediate lemma will be useful in the analysis of these cases.

Lemma 4. *Let $u \in \Sigma^+$ such that $u = xy$ and $y \in \mathrm{Pref}(u)$ for some θ-palindrome words $x, y \in \Sigma^+$. If $|y| \geq |x|$, then $\rho(x) = \rho(y) = \rho(u)$.*

Proof. We have $u = xy = yz$ for some $z \in \Sigma^+$ of the same length as x. The length condition implies that $x \in \mathrm{Pref}(y)$. Since $x = \theta(x)$ and $y = \theta(y)$, this means that $x \in \mathrm{Suff}(y)$ and hence $z = x$. So we have $u = xy = yx$, and hence x, y, and u share their primitive root. □

Unlike in the case of the original Lyndon-Schützenberger equation, the investigation of our extension entails the consideration of an enormous amount of cases since for each variable u_i, v_j, w_k we have two possible values. However, in almost all cases, it is enough to consider the common prefix between $u_1 \dots u_l$ and $v_1 \dots v_n$ or the common suffix between $u_1 \dots u_l$ and $w_1 \dots w_m$ to prove that either the equation imposes θ-periodicity or the equation cannot hold.

Note that for the $(5, 3, m)$ or $(5, 4, m)$ extensions of the Lyndon-Schützenberger equation, we only have to consider the case when the border between v_n and w_1 is inside u_3 because otherwise Theorem 4 imposes θ-periodicity. Also even if the border is inside u_3, if $m|w| \geq 2|u| + |w|$, then we reach the same conclusion. Moreover, we can assume that w is θ-primitive since otherwise we would just increase the value of the parameter m. These observations justify the assumptions which will be made in the following propositions.

Proposition 2. *Let $u, v \in \Sigma^+$ such that v is a θ-primitive word, $u_1, u_2, u_3 \in \{u, \theta(u)\}$, and $v_1, \cdots, v_{2m+1} \in \{v, \theta(v)\}$ for some $m \geq 1$. If $v_1 \cdots v_{2m+1}$ is a proper prefix of $u_1u_2u_3$ and $2m|v| < 2|u| < (2m+1)|v|$, then $u_2 \neq u_1$ and $v_1 = \cdots = v_{2m+1}$. Moreover, $v_1 = yp$ and $u_1u_2 = (yp)^{2m}y$ for some $y, p \in \Sigma^*$ such that $y = \theta(y)$ and $p = \theta(p)$.*

Proposition 3. *Let $u, v \in \Sigma^+$ such that v is θ-primitive, $u_1, u_2, u_3 \in \{u, \theta(u)\}$, and $v_1, \cdots, v_{2m} \in \{v, \theta(v)\}$ for some $m \geq 2$. If $v_1 \cdots v_{2m} \in \mathrm{Pref}(u_1u_2u_3)$ and $(2m-1)|v| < 2|u| < 2m|v|$, then either $u_1 \neq u_2$ and $v_1 = \cdots = v_{2m}$, with $v_1 = yp$ and $u_1u_2 = (yp)^{2m-1}y$ for some $y, p \in \Sigma^*$ such that $y = \theta(y)$ and $p = \theta(p)$, or $u_1 = u_2$, $v_1 = \cdots = v_m$, and $v_{m+1} = \cdots = v_{2m} = \theta(v_1)$, with $u_1 = [r(tr)^i(rt)^{i+j}r]^{m-1}r(tr)^i(rt)^j$ and $v_1 = r(tr)^i(rt)^{i+j}r$ for some $i \geq 0$, $j \geq 1$, and $r, t \in \Sigma^*$ such that $r = \theta(r)$, $t = \theta(t)$, and rt is primitive.*

These propositions show that if we suppose v to be θ-primitive, then the values of u_1, u_2, u_4, and u_5 determine the values of $v_1, \ldots, v_n$ and $w_1, \ldots, w_m$ uniquely, modulo a substitution of v by $\theta(v)$, or of w by $\theta(w)$. Thus, they decrease significantly the number of cases to be considered. Furthermore, the value of u_3 may put an additional useful restriction on v or w.

Lemma 5. *Let $u, v \in \Sigma^+$ such that v is a θ-primitive word, $u_1, u_2, u_3 \in \{u, \theta(u)\}$, and $v_1, \cdots, v_n \in \{v, \theta(v)\}$ for some $n \geq 3$. If $v_1 \cdots v_n \in \mathrm{Pref}(u_1u_2u_3)$, $u_1 \neq u_2$, $u_1 = u_3$, and $(n-1)|v| < 2|u| < n|v|$, then $|v| < \frac{4}{2n-1}|u|$.*

Proof. Since θ is an involution, we may assume without loss of generality that $u_1 = u$ and $v_1 = v$. Propositions 2 and 3 imply that $v_1 = \cdots = v_n = v$. Hence $u\theta(u) = v^{n-1}x$ for some $x \in \mathrm{Pref}(v)$. Since $u\theta(u)$ is a θ-palindrome, $v^{n-1}x = \theta(x)\theta(v)^{n-1}$ and this implies that $x = \theta(x)$ and $v = yx$ for some nonempty θ-palindrome y (see Fig. 2).

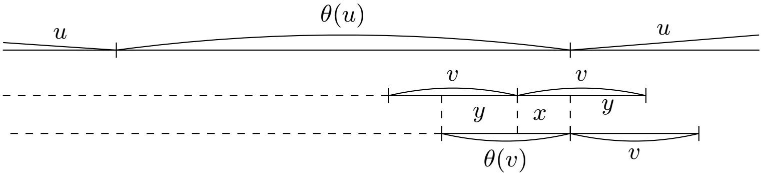


Fig. 2. Since u begins with v, y is a prefix of v

Since $v \in \mathrm{Pref}(u)$, we obtain that $y \in \mathrm{Pref}(v)$. If $|x| \leq |y|$, then Lemma 4 leads to a contradiction with the θ-primitivity of v. Thus $|y| < |x|$, which implies that $|y| < \frac{1}{2}|v|$. This means that $|v| < \frac{4}{2n-1}|u|$ because $|y| = n|v| - 2|u|$. □

All we did so far in studying the extended Lyndon-Schützenberger equation $u_1 \ldots u_5 = v_1 \ldots v_n \; w_1 \ldots w_m$ was to consider either the common prefix of $v_1 \ldots v_n$ and $u_1 \ldots u_5$, or the common suffix of $w_1 \ldots w_m$ and $u_1 \ldots u_5$. Next, we combine them together and consider the whole equation. The following lemma proves to be useful for our considerations.

Lemma 6. *Let $u, v \in \Sigma^+$ such that v is a θ-primitive word, $u_1, u_2, u_3 \in \{u, \theta(u)\}$ and $v_1, \cdots, v_n \in \{v, \theta(v)\}$ for some $n \geq 3$. If $v_1 \cdots v_n = u_1 u_2 z$ for some $z \in \mathrm{Pref}(u_3)$, $u_1 = u_2$, and $(n-1)|v| < 2|u|$, then $v_1 = xyx$ and $z = x^2$ for some $x, y \in \Sigma^+$ such that $x = \theta(x)$ and $yx = \theta(yx)$.*

Proof. Just as before, we can assume that $u_1 = u_2 = u$ and $v_1 = v$. Propositions 2 and 3 imply that n must be even, so let $n = 2m$ for some $m \geq 2$, and $u = \{r(tr)^i (rt)^{i+j} r\}^{m-1} r(tr)^i (rt)^{i+j}$ and $v = r(tr)^i (rt)^{i+j} r$ for some $r, t \in \Sigma^*$ such that $r = \theta(r)$, $t = \theta(t)$, $i \geq 0$, and $j \geq 1$. By taking $x = r(tr)^i$ and $y = (rt)^j$, we complete the proof. □

Next, we prove that the triple $(5, 4, m)$ imposes θ-periodicity for any $m \geq 4$.

Theorem 12. *Let $u, v, w \in \Sigma^+$, $u_1, u_2, u_3, u_4, u_5 \in \{u, \theta(u)\}$, $v_1, v_2, v_3, v_4 \in \{v, \theta(v)\}$, and $w_1, \cdots, w_m \in \{w, \theta(w)\}$ for some $m \geq 4$. If these words satisfy $u_1 u_2 u_3 u_4 u_5 = v_1 v_2 v_3 v_4\ w_1 \cdots w_m$, then u is not θ-primitive and $u, v, w \in \{t, \theta(t)\}^+$ for some $t \in \Sigma^+$.*

Proof. First note that we can assume that w is θ-primitive, since otherwise we would just increase the numerical parameter m. If u is not θ-primitive, that is, $u \in \{p, \theta(p)\}^k$ for some θ-primitive word $p \in \Sigma^+$ and $k \geq 2$, then the equation can be rewritten as $p_1 p_2 \cdots p_{5k} = v_1 v_2 v_3 v_4 w_1 \ldots w_m$, where $p_i \in \{p, \theta(p)\}$ for $1 \leq i \leq 5k$. But then, due to Theorem 9, we obtain that $v, w \in \{p, \theta(p)\}^+$. Furthermore, we can assume that also v is θ-primitive. Indeed, if it is not, then $v \in \{q, \theta(q)\}^j$ for some θ-primitive word q and $j \geq 2$. Then, the equation becomes $u_1 \ldots u_5 = q_1 \ldots q_{4j} w_1 w_2 \ldots w_m$, where $q_i \in \{q, \theta(q)\}$ for $1 \leq i \leq 4j$. But this implies that $u, w \in \{q, \theta(q)\}^+$ due to Theorem 11. Since u and w are assumed to be θ-primitive, $u, w \in \{q, \theta(q)\}$ and we have $5|q| < 4j|q| + m|q|$, which contradicts the fact that u, v, and w satisfy the equation $u_1 \ldots u_5 = q_1 \ldots q_{4j} w_1 w_2 \ldots w_m$. Even when v is θ-primitive, if $m \geq 7$ then the same argument leads to the same contradiction.

Now we will show that if u, v, and w are θ-primitive, then the equation cannot hold for $m \leq 6$. Since θ is an involution, we can assume that $u_1 = u$, $v_1 = v$, and $w_1 = w$. Let us start by supposing that u, v, and w satisfy $u_1 u_2 u_3 u_4 u_5 = v_1 v_2 v_3 v_4\ w_1 \cdots w_m$. Now, we have several cases depending on where the border between v_4 and w_1 is located. If it is left to or on the border between u_2 and u_3, then Theorem 4 implies that $u, w \in \{t, \theta(t)\}^+$ for some θ-primitive word $t \in \Sigma^+$, which further implies that also $v \in \{t, \theta(t)\}^+$. In fact, $u, v, w \in \{t, \theta(t)\}$ because they are θ-primitive. Then $|u_1 \ldots u_5| = 5|t|$, while $|v_1 v_2 v_3 v_4 w_1 \ldots w_m| = (4 + m)|t|$ with $m \geq 4$, which is a contradiction. The case when the border between v_4 and w_1 is right to or on the border between u_3 and u_4 will lead the contradiction along the same argument.

So let us suppose that $|u_1 u_2| < |v_1 v_2 v_3 v_4| < |u_1 u_2 u_3|$. Note that under this supposition, $|v|, |w| < |u|$. If $m|w| \geq 2|u| + |w| - 1$, then $u_3 u_4 u_5$ and $w_1 \ldots w_m$ share a suffix long enough to impose the θ-periodicity onto u and w due to Theorem 4. However, as explained before, this leads to a contradiction. This argument also applies to $u_1 u_2 u_3$ and $v_1 v_2 v_3 v_4$. As a result, it is enough to consider the case when $3|v| < 2|u| < 4|v|$ and $(m-1)|w| < 2|u| < m|w|$.

There are 16 cases to be considered depending on the values of u_2, u_3, u_4, and u_5. Note that once these values are determined, the values of v_1, v_2, v_3, v_4 and $w_1, \cdots, w_m$ are set uniquely due to Propositions 2 and 3. We number these cases from 0 to 15 by regarding $u_2u_3u_4u_5$ as the 4-bit number based on the conversion $u \to 0$ and $\theta(u) \to 1$. For example, case 5 is $u_2u_3u_4u_5 = u\theta(u)u\theta(u)$.

First, we consider the case 2, that is, $uuu\theta(u)u = v_1 \cdots v_4\ w_1 \cdots w_m$. Since $3|v| < 2|u| < 4|v|$, $|v| < \frac{2}{3}|u|$. Moreover, Lemma 5 implies that $|w| < \frac{4}{2m-1}|u|$. Then $5|u| - (4|v| + m|w|) > 0$ which contradicts the fact that u, v, and w satisfy the given equation. The same arguments work for the cases when either $u_1u_2u_3 = u\theta(u)u$ (i.e., cases 8, 9, 10, 11), or $u_3u_4u_5 = u\theta(u)u$ (i.e., cases 2, 10), or $u_3u_4u_5 = \theta(u)u\theta(u)$ (i.e., cases 5, 13).

Secondly we consider the case 1, that is, $uuuu\theta(u) = v_1 \cdots v_4w_1 \cdots w_m$. Let $uux = v_1 \cdots v_4$, $yu\theta(u) = w_1 \cdots w_m$ for some $x, y \in \Sigma^+$ such that $u = xy$. We immediately obtain now, due to Lemma 6, that $x = \theta(x)$. Since $x \in \text{Pref}(u_3)$, this means that $x \in \text{Suff}(u_5)$, which implies that $w_m \in \text{Suff}(x)$ or $x \in \text{Suff}(w_m)$. In both cases, we obtain that $u_3u_4u_5$ and $w_mw_1w_2 \ldots w_m$ share a common suffix of length at least $2|u| + |w| - 1$. Then we employ Theorem 4 to lead a contradiction. Among the cases left to be investigated, the only one where we cannot apply this technique is case 0.

Now, case 0 is $u_1 = u_2 = u_3 = u_4 = u_5 = u$. Applying Propositions 2 and 3, we have that $m = 2k$ for some $k \geq 2$, $w_1 = \cdots = w_k = w$, $w_{k+1} = \cdots = w_{2k} = \theta(w)$, $v_1 = v_2 = v$, and $v_3 = v_4 = \theta(v)$. Note that $k \in \{2, 3\}$ since $4 \leq m \leq 6$. Then, Lemma 6 implies that $u = xyxxy = (y'x'x')^{k-1}y'x' = x^2x'^2$, $v = xyx$, and $\theta(w) = x'y'x'$ for some $x, y, x', y' \in \Sigma^+$ with $x = \theta(x)$, $yx = \theta(yx)$, $x' = \theta(x')$, and $x'y' = \theta(x'y')$.

When $k = 2$, i.e., $xyxxy = y'x'x'y'x'$, we have three subcases depending on the lengths of xy and $y'x'$. If $|xy| < |y'x'|$, then by looking at the two sides of the equality $xyxxy = y'x'x'y'x'$, we obtain $y'x' = xyz = \theta(z)xy$ and $x = zx'\theta(z)$ for some $z \in \Sigma^+$. Substituting $x = zx'\theta(z)$ into $xyz = \theta(z)xy$ we get $z = \theta(z)$, and hence $y'x' = xyz = zxy$. Thus, $y'x', xy, z \in \{p\}^+$ for some primitive word p. Let $z = p^i$ and $y'x' = p^j$ for some $i, j \geq 1$. Then $y'x' = zxy$ and $x = zx'z$ imply that $p^j = p^{2i}x'p^iy$. Since p is primitive, we obtain that $\rho(x') = p$, which contradicts the θ-primitivity of $\theta(w) = x'y'x'$. For the case when $|xy| > |y'x'|$ we can use similar arguments to reach a contradiction. Finally, if $|xy| = |y'x'|$, then $x = x'$, which is a contradiction with the θ-primitivity of u since $u = xxx'x'$.

When $k = 3$, i.e., $u = xyxxy = (y'x'x')^2y'x'$, we first note that $|xy| > |y'x'|$ and $|xyx| > |y'x'x'|$. If $|xy| \geq |y'x'x'|$, then, by the Fine and Wilf theorem, $\rho(xyx) = \rho(y'x'x')$. Since xyx is strictly longer than $y'x'x'$, this means that $v = xyx$ is not primitive, which is a contradiction. Otherwise, i.e., $|y'x'| < |xy| < |y'x'x'|$, let $xy = y'x'z$ for some $z \in \text{Pref}(x')$. Since $x' = \theta(x')$, the equation $xyxxy = (y'x'x')^2y'x'$ also implies that $xy = \theta(z)y'x'$. Moreover, since $xy = y'x'z = \theta(z)y'x'$ and $\theta(z) \in \text{Suff}(x')$, we obtain $z = \theta(z)$. Thus $xy, y'x', z \in \{q\}^+$ for some primitive word $q \in \Sigma^+$, which, just as above, contradicts the θ-primitivity of $\theta(w)$. □

The next result shows that also the triple $(5, 3, m)$ imposes θ-periodicity for any $m \geq 3$. However, due to space limitations, we omit its proof.

Theorem 13. *Let $u, v, w \in \Sigma^+$, $u_1, u_2, u_3, u_4, u_5 \in \{u, \theta(u)\}$, $v_1, v_2, v_3 \in \{v, \theta(v)\}$, and $w_1, \cdots, w_m \in \{w, \theta(w)\}$ with $m \geq 3$. If these words verify the equation $u_1u_2u_3u_4u_5 = v_1v_2v_3\ w_1 \cdots w_m$, then u is not θ-primitive and $u, v, w \in \{t, \theta(t)\}^+$ for some $t \in \Sigma^+$.*

Acknowledgement

The authors are indebted to the anonymous reviewers for their valuable comments on a previous version of this paper.

References

1. Adleman, L.M.: Molecular computation of solutions to combinatorial problems. Science 266(5187), 1021–1024 (1994)
2. Crochemore, M., Hancart, C., Lecroq, T.: Algorithms on Strings. Cambridge University Press, Cambridge (2007)
3. Crochemore, M., Rytter, W.: Jewels of Stringology. World Scientific, Singapore (2002)
4. Choffrut, C., Karhumäki, J.: Combinatorics of words. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 1, pp. 329–438. Springer, Heidelberg (1997)
5. Lothaire, M.: Combinatorics on Words. Encyclopedia of Mathematics and its Applications, vol. 17. Addison-Wesley, Reading (1983)
6. Cummings, L.J., Smyth, W.F.: Weak repetitions in strings. Journal of Combinatorial Mathematics and Combinatorial Computing 24, 33–48 (1997)
7. Constantinescu, S., Ilie, L.: Fine and Wilf's theorem for Abelian periods. Bulletin of the EATCS 89, 167–170 (2006)
8. Czeizler, E., Kari, L., Seki, S.: On a special class of primitive words. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 265–277. Springer, Heidelberg (2008)
9. Lyndon, R.C., Schützenberger, M.P.: The equation $a^m = b^n c^p$ in a free group. Michigan Mathematical Journal 9, 289–298 (1962)
10. Harju, T., Nowotka, D.: The equation $x^i = y^j z^k$ in a free semigroup. Semigroup Forum 68, 488–490 (2004)
11. Harju, T., Nowotka, D.: On the equation $x^k = z_1^{k_1} z_2^{k_2} \ldots z_n^{k_n}$ in a free semigroup. Theoretical Computer Science 330(1), 117–121 (2005)
12. Lentin, A.: Sur l'équation $a^m = b^n c^p d^q$ dans un monoïde libre. Comptes Rendus de l'Académie des Sciences Paris 260, 3242–3244 (1965)
13. Kari, L., Mahalingam, K.: Watson-Crick palindromes. To appear in Natural Computing
14. de Luca, A., de Luca, A.: Pseudopalindrome closure operators in free monoids. Theoretical Computer Science 362, 282–300 (2006)
15. Fine, N.J., Wilf, H.S.: Uniqueness theorem for periodic functions. Proceedings of the American Mathematical Society 16(1), 109–114 (1965)

Asymptotic Cellular Complexity

Bruno Durand and Victor Poupet

Laboratoire d'informatique fondamentale de Marseille (LIF)
Aix-Marseille Université, CNRS
{bruno.durand,victor.poupet}@lif.univ-mrs.fr

Abstract. We show here how to construct a cellular automaton whose asymptotic set (the set of configurations it converges to) is maximally complex: it contains only configurations of maximal Kolmogorov complexity. This cellular automaton hence exhibits the most complex possible asymptotic behavior.

1 Introduction

Attractors are one of the key concepts in the study of dynamical systems. Discrete complex systems have the same concern and their asymptotic behavior has been the subject matter of many publications in theoretical computer science.

Cellular automata form the simplest of these complex systems and can produce complex behaviors from a finitely defined local transition rule. Many different classifications have been proposed, several of which based on their asymptotic behavior [2,4,8,12].

Litterature usually considers *limit sets* as attractors; they have many interesting properties (topological and algorithmic). However they contain some transient configurations. In this article we work both on limit sets and on a tighter definition: the *asymptotic set*. It was recently proved by Cervelle [3] that some cellular automata have linearly complex asymptotic sets but the theoretical bound on the complexity of two-dimensional cellular automata is quadratic. We show here that an optimal quadratic bound can indeed be achieved.

Our construction is inspired by the proof of a complexity result on tilings of the plane [5]. Nevertheless, there is a major difference: tilings are "static" models and only linear complexity can be obtained. Taking advantage of the dynamical aspect of cellular automata gives an improvement in the complexity to n^2.

The article is organized as follows: Section 2 recalls some definitions and properties about cellular automata, asymptotic behavior and Kolmogorov complexity. Section 3 states the main theorem and proves it by detailing the construction of a cellular automaton of high complexity. Section 4 discusses possible extensions and improvements of the result.

2 Definitions

Definition 1 (Cellular Automaton). *A cellular automaton (CA) is a discrete dynamical system defined by a quadruple $\mathcal{A} = (d, \mathcal{Q}, V, \delta)$ where $d \in \mathbb{N}$ is*

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 195–206, 2009.

the dimension of the CA, $\mathcal{Q}$ is a finite set called set of states*, $V \subseteq \mathbb{Z}^d$ is a finite set called* neighborhood *and $\delta : \mathcal{Q}^V \to \mathcal{Q}$ is the local transition function of the automaton.*

For a given automaton $\mathcal{A} = (d, \mathcal{Q}, V = \{v_1, \ldots, v_n\}, \delta)$, we call *configuration* any mapping $\mathfrak{C} : \mathbb{Z}^d \to \mathcal{Q}$. Elements of $\mathbb{Z}^d$ are called *cells*. Given a configuration $\mathfrak{C}$, we say that a cell c is in state q if $\mathfrak{C}(c) = q$. From the local function δ we define a global function $\Delta : \mathcal{Q}^{\mathbb{Z}^d} \to \mathcal{Q}^{\mathbb{Z}^d}$ associating any configuration C with the configuration $\mathfrak{C}'$ such that

$$\forall x \in \mathbb{Z}^d, \mathfrak{C}'(x) = \delta(\mathfrak{C}(x + v_1), \ldots, \mathfrak{C}(x + v_n))$$

In this article we mainly consider two-dimensional CA ($d = 2$). We work on the Moore neighborhood (9 nearest neighbors), although the choice of the neighborhood has little relevance to our construction.

Definition 2 (Spreading State). *A CA $\mathcal{A} = (d, \mathcal{Q}, V, \delta)$ is said to have a* spreading state *q_s if, as soon as one of the neighbors of a cell is in state q_s at time t, then it becomes in state q_s at time $(t + 1)$.*

Definition 3 (Asymptotic Set). *The* asymptotic set *of a CA $\mathcal{A}$ is the union of the accumulation points*[1] *of all its orbits:*

$$\mathfrak{U}(\mathcal{A}) = \bigcup_{\mathfrak{C} \in \mathcal{Q}^{\mathbb{Z}^d}} \bigcap_{n \in \mathbb{N}} \overline{\{\Delta^i(\mathfrak{C})\}_{i \geq n}}$$

Informally, a configuration $\mathfrak{C}$ is in the asymptotic set of $\mathcal{A}$ if there is a configuration $\mathfrak{C}_0$ such that the orbit of $\mathfrak{C}_0$ has infinitely many configurations that coincide with $\mathfrak{C}$ on arbitrarily large areas around the origin.

This definition is different from the standard definition of a *limit set* $\Lambda(\mathcal{A}) = \bigcap_{n \in \mathbb{N}} \Delta^n(\mathcal{Q}^{\mathbb{Z}^d})$. The limit set has nice topological properties (in particular it is a closed set) but contains, in addition to all the asymptotic configurations, some that disappear after some time[2]. The asymptotic set more accurately captures the asymptotic behavior of the automaton.

In the sequel, we use the following two lemmas (the proofs are very straightforward from the definitions):

Lemma 1. *There is always a uniform configuration in the asymptotic set of a cellular automaton. Moreover, $\Delta(\mathfrak{U}) = \mathfrak{U}$.*

Lemma 2. *If a cellular automaton has a spreading state □, the uniformly □ configuration is in the asymptotic set of the automaton. Moreover, the □ state does not appear in any other configuration of the asymptotic set.*

[1] We consider the usual topology induced by the Cantor distance: the distance between two configurations $\mathfrak{C}_1$ and $\mathfrak{C}_2$ is 2^{-k} where $k = \min\{\|x\| \mid \mathfrak{C}_1(x) \neq \mathfrak{C}_2(x)\}$.

[2] Consider for instance a one-dimensional CA with two states 0 and 1 for which each cell takes the maximal of the states in its neighborhood. Finite segments of 0 disappear over time but configurations of the form ${}^{\omega}10^n1^{\omega}$ have pre-images of any order and are hence in the limit set.

Definition 4 (Kolmogorov Complexity). *Given a recursive function f, the Kolmogorov complexity relative to f of a string $x \in \{0,1\}^*$ is defined as $K_f(x) = \min\{|y| \mid f(y) = x\}$ (it is infinite if the set $\{|y| \mid f(y) = x\}$ is empty).*

The idea is then to drop the recursive function f. This can be done by using the founding Kolmogorov theorem: there exists a recursive function U (called additively optimal) such that for any recursive function f, there is a constant $c_f \in \mathbb{N}$ such that for any string $x \in \{0,1\}^*$ we have $K_U(x) \leq K_f(x) + c_f$. The Kolmogorov complexity of a string x is then denoted as $K(x) = K_U(x)$ for some additively optimal U.

Informally, the Kolmogorov complexity of an object is its shortest possible description (after choosing the description language). In the following we will talk about Kolmogorov complexity of square patterns over a finite alphabet (not necessarily $\{0,1\}$). It is defined by choosing an encoding of these patterns into words over $\{0,1\}$ and considering the Kolmogorov complexity of the resulting encoding. The chosen encoding does not matter since it only affects the result up to an additive constant.

We will use the following well known properties of the Kolmogorov complexity:

Proposition 1. *There exists a constant c such that any string x has complexity less than $(|x|+c)$ and any square pattern of dimension $n \times n$ over a finite alphabet $\mathcal{Q}$ has complexity at most $(\log_2(|\mathcal{Q}|).n^2 + c)$.*

For all $\rho \in \mathbb{R}$ it is possible to enumerate all strings x such that $K(x) \leq \rho.|x|$.

Definition 5 (ρ-complexity). *Given a constant $\rho > 0$, a square pattern of size $n \times n$ on an alphabet $\mathcal{Q}$ is said to be ρ-*complex *if its Kolmogorov complexity is greater than $\rho.n^2$.*

A configuration $\mathfrak{C} \in \mathcal{Q}^{\mathbb{Z}^2}$ of the plane is ρ-complex if there exists a constant n_0 such that all square patterns of size $n \times n$ for $n \geq n_0$ are ρ-complex.

*Square patterns and configurations that are not ρ-complex are said to be ρ-*simple*.*

3 Main Theorem

This whole section is dedicated to the proof of the following theorem:

Theorem 1. *For any constant $0 < \rho < 1$ there exists a CA $\mathcal{A}$ whose asymptotic set contains infinitely many ρ-complex configurations and only one ρ-simple configuration.*

For practical and readability reasons, we will not give here a complete formal proof of the theorem. We will instead describe the construction of a cellular automaton that verifies the announced property and show how to solve the different difficulties that arise. A good understanding of the main ideas should be enough to convince the reader that a complete formal proof can indeed be derived.

Intuitively, for this constructed CA, all configurations of the asymptotic set are complex, except the unavoidable constant one. Its limit set however has more than these configurations as it can also have "hybrid" configurations that are a reunion of complex areas (areas that can be extended into ρ-complex configurations) and uniform areas.

3.1 Layered Structure

About Layers. The automaton will be described as a finite superposition of "layers". Saying that a cellular automaton has n layers means that its set of states is the cartesian product of n finite sets of "sub-states" ($\mathcal{Q} = \mathcal{Q}_1 \times \mathcal{Q}_2 \times \ldots \times \mathcal{Q}_n$). Layers evolve simultaneously. Some layers can work independently of the others (the sub-state of a cell only depends on the corresponding sub-states of its neighbors) while some will evolve differently according to what lies on the other layers.

Layers are mainly used to simplify the description of a cellular automaton: we sometimes need to perform different computations on the same set of cells, in which case we use a layer for each of the computations. If there are only finitely many computations to perform on each cell (and the bound is uniform) the set of states of the automaton thus constructed by layers remains finite.

We use three main layers and a spreading state $\square$. The set of states of the automaton is therefore $\mathcal{Q} = \{\square\} \cup (\mathcal{Q}_B \times \mathcal{Q}_R \times \mathcal{Q}_C)$ where $\mathcal{Q}_B$, $\mathcal{Q}_R$ and $\mathcal{Q}_C$ are the finite sets of sub-states corresponding to each layer.

The Bitmap Layer. The first layer is a very simple one. It contains only one bit ($\mathcal{Q}_B = \{0, 1\}$) on each cell and this bit does not change when the automaton evolves (except if the $\square$ state propagates over all layers of course). It is nonetheless very important to our automaton because it is where the complexity will be. The whole automaton will then be designed to search for ρ-simple square patterns on this layer. If one is found then a $\square$ state will appear and the configuration will not be in the asymptotic set.

Conversely, if none of the patterns is ρ-simple then no $\square$ state will appear and there will be a configuration in the asymptotic set whose bitmap layer is the one considered. Such a configuration is ρ-complex no matter what information lies on the other layers.

The Robinson Layer. The second layer is a Robinson tiling. This means that each cell of the automaton is associated to one of the Wang tiles of the considered set. $\mathcal{Q}_R$ is therefore a set of Wang tiles. This layer does not change over time either, but each cell checks locally the correctness of the tiling. If a cell sees an error in its surrounding, a $\square$ state is generated and, as seen before, the configuration is not in the asymptotic set. The asymptotic set can hence only contain configurations for which the Robinson layer contains a valid tiling of the plane. Because this layer does not change over time, we will use the properties of the Robinson tiling to define square areas on which a computation can be realized.

The Computation Layer. This last layer is the one where the real computation takes place. It will use the squares of the Robinson layer to define areas on which separate computations will enumerate ρ-simple patterns and look for them on the bitmap layer. When a ρ-simple pattern is found, a spreading state $\square$ is produced and the whole configuration is thus removed from the asymptotic set.

3.2 The Robinson Hierarchy

Properties of the Robinson Tiling. Fully describing the construction of a Robinson tiling and proving its properties is way beyond the scope of this article. Extensive literature has been written on the subject and any reader unfamiliar with the techniques but eager to learn more about it should refer to the books and articles presented in the bibliography [10,6,1]. We will only focus on the general properties of valid tilings without discussing how such properties are enforced locally by a set of Wang tiles.

The construction we will be using here is the "strong Robinson tiling" presented in [9]. Using this construction we can add decorations to the tiles (we "draw" lines on each tile) such that any valid tiling of the plane exhibits a self-similar structure of squares of different sizes. More precisely, in such strong Robinson tilings, all squares of a given size are aligned, while in the original Robinson tiling, it is not always the case because of the degenerate case where a fracture line appears.

Robinson Squares. The size of a square is the length of its sides. Any valid tiling has squares of size 4^n for all n. Squares of the same size are regularly arranged on a grid (rows and columns of aligned squares). The space between two adjacent squares of size 4^n is exactly 4^n. Moreover, grids of squares of different sizes are placed the ones relatively to the others in such a way that each square of size 4^{n+1} contains four squares of size 4^n. Figure 1 illustrates a portion of a valid Robinson tiling (filled and hatched areas should be ignored for now).

Infinite Squares. An important property of the Robinson squares that we will be using later is that each row and each column of the plane contains sides of squares of exactly one size: two squares of different sizes cannot have aligned sides and there are square sides on every row and column. A special case is when all squares are placed in such a way that a line or column has no finite-sized square side on it. In this case, an "infinite square" fills the empty row or column: if there was only an empty column (or an empty row), a bi-infinite line goes through it: it is the side of an infinite square. If both a line and a column are left empty, two perpendicular semi-infinite lines form the angle of an infinite square.

Computation. Robinson squares will be required to perform some computational tasks. The computation will take place on the lower side of the square.

For practical reasons, we will ignore squares that are too small to start a correct computation. We will therefore only consider squares of size greater than $s_0 = 4^{k_0}$ for some s_0 that should be chosen after explaining the whole construction to see how much space a segment needs to correctly start its computation. From now on, Robinson squares of size 4^{k_0+n} will be called squares of level n (or simply n-squares). Squares of level 0 are the smallest ones that we will consider.

The Life Cycle of a Robinson Square. Each Robinson square (large enough since we are ignoring the smaller ones) will perform a computation on its lower segment. Because this segment is finite, the duration of a computation before entering a loop is also limited. To make sure that each square does its verification correctly, we will reset the whole computation periodically. To do so, we implement a binary counter on the lower side of each square that starts on the bottom left corner and grows towards the bottom right one. When the counter reaches the bottom right corner of the square, a signal resets the counter and the computation that takes place on the square.

Incrementation signals are sent by all bottom left angles of all Robinson squares. Any Robinson square of size s will be reset by its counter after at most 2^{s+1} steps no matter what computation it was initially performing (if any). From there we know that the square behaves as intended from a freshly initialized configuration. The duration of a cycle is exponential in the size of the square, and we will show that it is sufficient to perform the tasks that we need, which require a polynomial time.

3.3 Reading the Bitmap Layer

Some cells of the plane are inside infinitely many Robinson squares. For this reason we cannot allow each square of any possible size to access directly the bitmap layer because this would lead to an unbounded number of competitive requests on some cells. Instead, only the smallest considered Robinson squares (level 0) will access directly the bitmap layer. All other squares will query recursively the squares of lower level to obtain the required information.

Visibility and Responsibility. Let us define the *visibility area* of a Robinson square of size s as the square surface of size $2s$ centered on it (see Figure 1). Because of the regular arrangement of Robinson squares, for each k, any bit of the bitmap layer is in the visibility area of exactly one k-square. Moreover, the visibility area of a k-square is exactly the union of that of the 16 closest $(k-1)$-squares.

Visibility areas are very convenient because, at a given level, they form an exact partition of the plane. They are, however, not sufficient for our purpose.

The problem comes from the special Robinson tilings described earlier that contain some "degenerate" squares of infinite size. This means that in some cases, some square patterns are not included in the visibility area of any finite Robinson square.

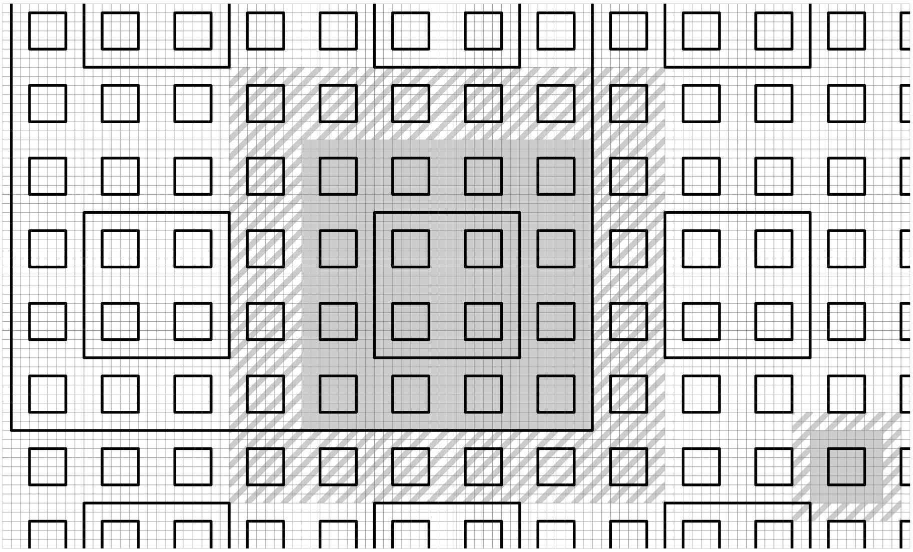


Fig. 1. The visibility area (grey) and the responsibility area (hatched) of two Robinson squares

To solve this problem, we will have to widen the area over which a square looks for simple patterns. We define the *responsibility area* of a Robinson square of size s as the square surface of size $3s$ centered on it (see Figure 1). Of particular interest is the fact that the responsibility area of a k-square is exactly the disjoint union of the visibility areas of the 36 closest $(k-1)$-squares. Note that responsibility areas of neighboring squares of the same level overlap, but it will not be a problem because a given cell can be in at most 4 responsibility areas of squares of the same level.

Communication Channels. All Robinson squares will need to read the bits on the bitmap layer when looking for simple patterns in their responsibility area. Because only 0-squares can access those bits directly, we have to set up a communication system between squares of different levels.

The idea is that each time a k-square needs to know a bit value, it will ask the corresponding $(k-1)$-square (the one whose visibility area contains the bit), which will in turn ask a $(k-2)$-square, and so on until the request is finally made to a 0-square that can directly access the bit and answer the query. Answers will then be transmitted recursively to the upper levels so that the original k-square can have the needed information.

A given Robinson square of any level can only request bits inside its responsibility area. However, each request is addressed to the square whose visibility area contains the bit, so after the first step of a recursive request, all subsequent requests are made inside visibility areas only. It is enough that any given k-square be able to request a bit to any of the 36 $(k-1)$-squares around itself. We

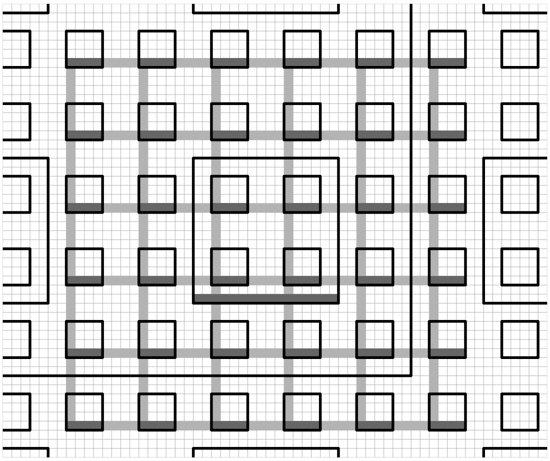


Fig. 2. The communication channels used by a level k Robinson square to communicate with the 36 closest level $(k-1)$ Robinson squares

will use communication channels (connected paths) from a square to another to transmit queries and answers. What we need to ensure is that there is a global upper bound on the number of communication channels that pass through a given cell.

Since requests from a k-square are aimed at level $(k-1)$ squares, we will make the corresponding channels along the lines and columns on which level $(k-1)$ square sides lie. Figure 2 illustrates the communication channels used by a k-square to communicate with the necessary $(k-1)$-squares (the dark grey areas are the computing segments of the squares).

Because Robinson squares of different levels cannot have aligned sides, a line or column used in a channel network between a level k square and its surrounding level $(k-1)$ squares will not be used for channels connecting other levels of Robinson squares. As a consequence, any given cell of the plane lies on at most two levels of communication channels. Moreover a cell is in the responsibility area of at most 4 k-squares. These two last observations mean that a cell is on the communication network of at most 8 different Robinson squares (of any level). If we ensure that each Robinson square does at most one request at time, no more than 8 requests can pass on a single cell at the same time so the communications can be realized with a finite number of states.

Recursive Requests. We now have the necessary framework to explain in details how a high level square of size s can obtain the value of a bit on its responsibility area.

Coordinates of the required bits are considered relatively to the lower-left corner of the responsibility area of a square if they were generated by the square itself (during its computation it requires a given bit) and relatively to the lower left corner of the visibility area otherwise (when the request comes from a higher level square). This might seem unnecessarily complicated but by doing so we will only deal with positive coordinates and all modulo computations will be very easy to perform. Simply consider that the coordinates are with respect to the lowest and leftmost possible location of a bit.

All coordinates will be represented in binary form and we will assume that coordinates are padded with 0 so that they all have the maximum length (coordinates relative to the responsibility area are one bit longer than the ones relative to the visibility area).

When a square knows the coordinates of the bit it requires, it has to determine in which of the neighboring lower level squares' area of visibility the bit is, convert the coordinates relatively to that lower level square and send them to it. All these operations are very easy to handle because lower level squares are of size $s/4$, regularly arranged around the square of size s and their coordinate system "coincides" with that of the larger square.

This means that the most significant bits directly determine which lower square should be contacted, and the remaining bits represent the coordinates of the bit relatively to this lower square's visibility area. If the initial coordinates are relative to the visibility area, the two most significant bits determine which lower square to ask. If relative to the visibility area, the 3 first bits should be considered.

From these bits, a *path* is created that indicates where the destination square is. This path indicates how to travel through the communication network described earlier, from square to square, to reach the destination. It is a word on the aphabet $\{\uparrow, \downarrow, \rightarrow, \leftarrow\}$. After the path has been determined (in constant time since there are only finitely many possibilities), it is appended to the reduced coordinates (coordinates relative to the lower level destination square), to form a message that is sent through the communication network. This message is written on several cells: one bit of the coordinates or one letter of the path per cell.

When the destination square receives the message, it keeps the path, reads the coordinates, converts them and sends the next request (to the lower level square) the same way. When the answer to this request arrives, it can use the path (interpreted backwards) to send back the result bit to the higher level square.

Note that a given square will send at most one request at a time in a normal behavior. If however it is reset while a request has been sent, a new one might be sent. In this case the most recent request is the only one that is considered.

Request Time. We will denote by $T(s)$ the maximum duration of a request from a square of size s, that is the maximum number of time steps from the time when the square has the coordinates of the bit it requires to the moment when it receives the bit value. Coordinates handled by a square of size s are of logarithmic size in s. According to what has been previously described, a request is handled as follows:

1. The square converts the coordinates, creates the path and message: $O(\log s)$
2. The message is sent to the lower level square: $O(s)$
3. The lower level square requests the bit: $T(s/4)$
4. The bit is send back from the lower level square using the path: $O(s)$.

There is however a slight complication: a given square can receive multiple requests at the same time. If it is in the area of responsibility of multiple squares of upper level, each of those can send a direct request. Each square also "produces" its own requests (from its own computation) that it has to send. A Robinson square can therefore handle up to 5 simultaneous requests.

In order not to exponentially increase this number, only one request is transmitted to the lower level at a time, the others are delayed until the answer is obtained. By using a "first in first out" ordering, we get $T(s) \leq 5(T(s/4)+O(s))$.

Since requests from 0-squares are completed in constant time, we get $T(s) = O(s^2)$. All requests are handled in quadratic time.

3.4 A Day in the Life of a Robinson Square

Now that we know how a Robinson square can get the values on the bitmap layer, which is by far the most complicated task performed on the automaton, let us have a look at the main computation.

The square's task is to look for "simple" square patterns in its responsibility area. The lower side of the Robinson square acts like a Turing machine with a tape of finite size s.

The computation of a Robinson square consists merely in enumerating ρ-simple patterns and, each time one is found, check that it does not appear in its responsibility area. Checking if a pattern of size $n \times n$ appears in the responsibility area of a square of size s can be done in time $O(n^2.s^4)$ without any optimization (roughly s^2 possible positions for the pattern, each bit requires s^2 steps).

Correctness of the Construction. Because looking for a pattern takes a polynomial time in the size of both the Robinson square and the pattern, and because each Robinson square has an exponential time to do the computation before it is reset, for any ρ-simple square pattern all sufficiently large Robinson squares will have enough time to produce it and search for it in their responsibility area.

The Robinson tiling has the property that any finite set of cells is in the responsibility area of an arbitrarily large Robinson square (hence the necessary overlapping of the responsibility areas). This means that any ρ-simple square pattern present on the bitmap layer will eventually be found by a large enough Robinson square (one whose responsibility area contains the pattern). If this pattern is of size larger than n_0, a $\square$ state will appear and the configuration will not appear in the asymptotic set. The construction therefore ensures that no configuration containing a ρ-simple pattern of size larger than n_0 on its bitmap layer is in the asymptotic set of the automaton (except for the all $\square$ configuration of course).

3.5 Infiniteness of the Asymptotic Set

Up to this point we have proved that the asymptotic set of the described automaton contained no ρ-simple configuration other than the uniform $\square$ configuration. It remains to be shown that the asymptotic set contains infinitely many ρ-complex configurations. We will use the following lemma proved in [11]:

Lemma 3. *For any* $0 < \rho < 1$*, there exists a* ρ*-complex configuration over the set of states* $\{0, 1\}$.

Consider now the evolution of the automaton from an initial configuration containing one of the complex configurations described in Lemma 3 on its bitmap layer, a valid Robinson tiling on its Robinson layer and empty initial computations on all Robinson squares. From such a configuration, all Robinson squares will properly perform their verification but none will find any simple pattern of size more than n_0. The automaton will evolve infinitely without producing any $\square$ state, and its bitmap layer will never be changed. This means that for each ρ-complex bitmap layer, there is a configuration in the asymptotic set of our automaton with this exact bitmap layer. There are hence infinitely many configurations in the asymptotic set. This completes the proof of Theorem 1.

4 Extensions

4.1 Optimality

For any $0 < \rho < 1$, the described automaton can ensure that no square pattern of size $n \geq n_0$ has complexity lower than $\rho.n^2$. It is clear that quadratic complexity is optimal but if the CA has N states the theoretical maximal complexity of a square pattern is $\log_2(N).n^2$. It is however possible to modify the construction to approach this theoretical bound by increasing the alphabet on the bitmap layer. If we use 2^α letters on this layer, the Robinson layer does not change and the computation layer only needs $O(\alpha)$ more states.

Producing the simple patterns is almost identical, requests on the bitmap layer are done bit by bit (α requests for a single letter) so the overall test time remains polynomial. Lemma 3 can be extended to larger alphabets and we can therefore ensure a complexity of at least $\rho.\alpha.n^2$ for all sufficiently large square patterns.

The number of states of this new automaton is $N = N_K.(N_C + \alpha).2^\alpha$. Since $\lim_{\alpha\to\infty} \log N = \alpha$ we can approach the optimal complexity as much as needed.

4.2 Other Dimensions

It might be possible to obtain a similar result in one dimension (with a linear lower bound on the complexity of the patterns). A similarly layered construction could work but the Robinson structure and the communications are more problematic. By using the “North-West deterministic” version of the Robinson tiling by Kari and Papasoglu [7] we can have a layer of the automaton on which a Robinson structure appears on the space-time diagram. In this case the finite areas are constantly evolving (segments of increasing and decreasing size) which makes the computation much harder to define and perform. We are currently working on a one-dimensional solution but the details have not yet been completely solved and written.

As for higher dimensions, there are different ways to extend the Robinson structure to finite cubes in such a way that any finite volume is in the visibility area of a cube (using three-dimensional substitution systems for instance). Computations are performed as in the previous construction on a segment and verifications can still be done in polynomial time.

References

1. Allauzen, C., Durand, B.: Tiling Problems. In: The classical decision problem. Perspectives in Mathematical Logic. Springer, Heidelberg (2001)
2. Cattaneo, G., Dennunzio, A., Margara, L.: Chaotic Subshifts and Related Languages Applications to one-dimensional Cellular Automata. Fundamenta Informaticae 52(1-3), 39–80 (2002)
3. Cervelle, J.: Complexité dynamique et algorithmique des automates cellulaires. Habilitation à diriger des recherches, Université Paris Est, Marne-la-Vallée (December 2007)
4. Culik II, K., Pachl, J.K., Yu, S.: On the Limit Sets of Cellular Automata. SIAM J. Comput. 18(4), 831–842 (1989)
5. Durand, B., Levin, L.A., Shen, A.: Complex tilings. Journal of Symbolic Logic 73(2), 593–613 (2008)
6. Johnson, A., Madden, K.: Putting the Pieces Together: Understanding Robinson's Nonperiodic Tilings. The College Mathematics Journal 28(3), 172–181 (1997)
7. Kari, J., Papasoglu, P.: Deterministic Aperiodic Tile Sets. Geometric And Functional Analysis 9, 353–369 (1999)
8. Kurka, P.: Languages, equicontinuity and attractors in cellular automata. Ergodic Theory & Dynamical Systems 17, 417–433 (1997)
9. Levin, L.A.: Aperiodic Tilings: Breaking Translational Symmetry. Comput. J. 48(6), 642–645 (2005)
10. Robinson, R.M.: Undecidability and Nonperiodicity for Tilings of the Plane. Inventiones Math 12 (1971)
11. Rumyantsev, A.Y., Ushakov, M.A.: Forbidden Substrings, Kolmogorov Complexity and Almost Periodic Sequences. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 396–407. Springer, Heidelberg (2006)
12. Wolfram, S.: Universality and complexity in cellular automata. Physica D 10, 1–35 (1984)

Strongly Regular Grammars and Regular Approximation of Context-Free Languages

Ömer Eğecioğlu

Department of Computer Science
University of California, Santa Barbara, CA 93106
omer@cs.ucsb.edu

Abstract. We consider algorithms for approximating context–free grammars by regular grammars, making use of Chomsky's characterization of non–self–embedding grammars as generating regular languages and a transformation by Mohri and Nederhof on sets of mutually recursive nonterminals. We give an exposition of strongly regular grammars and this transformation, and use it as a subprocedure to obtain tighter regular approximations to a given context-free grammar. In another direction, the generalization by a 1–lookahead extends Mohri and Nederhof's transformation by incorporating more context into the regular approximation at the expense of a larger grammar.

1 Introduction

The approximation of context-free languages with regular languages is a problem which has been extensively studied because of its importance in a number of applications [6,5,4]. A general framework for the approximation of formal languages by regular languages was studied by Shallit [7]. We consider the case in which a given context-free grammar is approximated from above by a regular grammar.

The algorithms discussed here make use of a transformation introduced by Mohri and Nederhof [4] as a subprocedure to provide tighter regular approximations. As in [4], the approximating grammar obtained is non–self–embedding. Such grammars generate regular languages by a result of Chomsky [2].

We assume that the grammar is in appropriate normal form, although for real-life problems discussed in [4] normal forms would already incur a quadratic increase in the size of the grammar, and may not be desirable. The starting point of normal forms is not a necessary assumption but simplifies the exposition: the resulting regular grammars are easier to keep track of because of the simplicity of their transition diagrams, for example.

We start with an exposition of the transformation of Mohri and Nederhof [4] and then discuss its variants that which provide tighter regular approximations. Regular approximation by two *cycle-breaking* based methods is presented in section 6 and approximation by *1–lookahead* is discussed in section 7.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 207–220, 2009.

2 Notation and Definitions

A *context–free grammar* (CFG) G is a 4–tuple $G = (N, T, P, S)$, where N and T are disjoint finite sets of *nonterminals* and *terminals*, respectively. P is a finite set of productions (rules); each production is of the form $A \rightarrow \alpha$, where A is a nonterminal and α is a string of symbols (sentential forms) from V^* where $V = N \cup T$. S is the *start symbol.* The relation $\rightarrow$ on $N \times V^*$ is extended to a relation on $V^* \times V^*$ as usual. The transitive and reflexive closure of $\rightarrow$ is denoted by $\xrightarrow{*}$. The language generated by an $A \in N$ is $\{w \in T^* \mid A \xrightarrow{*} w\}$. The language generated by G is $L(G) = \{w \in T^* \mid S \xrightarrow{*} w\}$. A context–free language (CFL) is a language generated by a CFG. The number of rules in the grammar G is denoted by $|G|$. We use the commonly used convention of denoting the set of nonterminals in N by capital letters $A, B, C, \ldots$, the set of terminals T with $a, b, c, \ldots$, strings of terminals in T^* with $u, v, w, \ldots$, strings of nonterminals and terminals in V^* by $\alpha, \beta, \gamma, \ldots$. The empty string is denoted by ε. Productions with left-hand side $A \in N$ are referred to as the *rules of* A or A*-rules.* The union of rules of $A \in M$ for $M \subseteq N$ are the *rules of* M.

If all productions of G are of the form $A \rightarrow wB$ or $A \rightarrow w$ then G is called a *right–linear* grammar. If all productions are of the form $A \rightarrow Bw$ or $A \rightarrow w$ then G is a *left–linear* grammar. G is a *regular* grammar if it is either right–linear or left–linear. Regular grammars characterize regular languages. In addition to regular grammars, regular languages can be represented in many forms such as finite automata (1NFA, 1DFA, 2NFA, 2DFA), and regular expressions, each giving a different insight into the structure of the language. In the Chomsky hierarchy of languages, context–free languages properly contain regular languages. Thus context–free grammars can generate languages which are non–regular, and in fact many languages of interest are context–free but non–regular.

A context–free grammar G is *self–embedding* (SE), if there exists a derivation $A \xrightarrow{*} \alpha A \beta$, with both α, β non–empty. G is *non–self–embedding* (NSE) if it is not self–embedding. By a result of Chomsky [2], any NSE grammar generates a regular language. For more details on notation and basic properties of CFGs and CFLs, the reader is referred to Hopcroft and Ullman [3].

3 Mohri and Nederhof's Transformation

In this section we describe the transformation of Mohri and Nederhof [4]. First, consider *strongly regular* CFGs which are defined as follows. Let $\Re$ be the relation defined on the set of nonterminals N of G by:

$$A \Re B \Leftrightarrow (\exists \alpha, \beta \in V^* \text{ s.t. } A \xrightarrow{*} \alpha B \beta) \wedge (\exists \alpha, \beta \in V^* \text{ s.t. } B \xrightarrow{*} \alpha A \beta) .$$

Note that α and β are not required to be nonempty. $\Re$ defines an equivalence relation on N, and partitions N into equivalence classes of nonterminals called *mutually recursive nonterminals.* Strongly regular grammars are grammars in which the rules of each set M of mutually recursive nonterminals are either all left–linear or all right–linear. In determining whether a rule of M is right–linear or left–linear, the nonterminals that do not belong to M are treated as if they are terminals. The class of languages generated by strongly regular grammars coincide with the class of languages generated by NSE grammars and therefore these languages are regular.

There are efficient algorithms to construct finite automata from strongly regular grammars. An offline construction was given by Nederhof in [6]. One may also construct an alternative, compact representation of the regular language generated, from which a finite automaton for it may be constructed, as shown by Mohri and Pereira in [5]. Briefly, the algorithm is as follows:

1. Determine sets of mutually recursive nonterminals by computing the strongly connected components of the graph of the grammar[1].
2. Construct a the automaton $\mathcal{K}(M)$ for each equivalence class M of mutually recursive nonterminals with unspecified initial state (in case M is right–linear) or unspecified final states (in case M is left–linear). For any $A \in M$, the automaton $\mathcal{N}(A)$ accepting terminals generated from A can be obtained from $\mathcal{K}(M)$.
3. For each input string w, obtain $\mathcal{N}(S)$ from the $\mathcal{K}(M)$ that satisfies $S \in M$. This automaton is then expanded in a lazy way by substituting other automata $\mathcal{N}(A)$ for occurrences of A in $\mathcal{N}(S)$ that are encountered while processing w.

In [4], Mohri and Nederhof describe a transformation that yields a strongly regular grammar from a given context-free grammar: for each class of mutually recursive nonterminals M such that the corresponding rules are not all right–linear or not all left–linear with respect to the nonterminals of M, the following transformation is applied:

1. For each nonterminal $A \in M$, introduce $A' \notin N$ and add the production $A' \to \varepsilon$ to the grammar.
2. For each production of the form: $A \to \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \dots B_m \alpha_m$ with $m \geq 0$, $B_1, \dots, B_m \in M, \alpha_0, \dots, \alpha_m \in (T \cup (N - M))^*$, replace it with

$$\begin{aligned} A &\to \alpha_0 B_1 \\ B'_1 &\to \alpha_1 B_2 \\ B'_2 &\to \alpha_2 B_3 \\ &\vdots \\ B'_{m-1} &\to \alpha_{m-1} B_m \\ B'_m &\to \alpha_m A' \end{aligned}$$

   If $m = 0$, this set of productions only contains $A \to \alpha_0 A'$.

[1] The graph of the grammar has a node for each nonterminal, and an edge from node A to node B iff B appears on the right hand side of a production having A on the left hand side.

All of the rules for M in the transformed grammar are right–linear. Therefore the resulting grammar is strongly regular. We will refer to this transformation as the *MN-transformation*, and the resulting regular approximation as the *MN-approximation*. The MN-approximation $\mathcal{L}(G)$ is a superset of $\mathcal{L}(G)$.

Since we are interested in how well the resulting regular language approximates the given one, we will consider the effect of the transformation on an individual equivalence class of mutually recursive set of nonterminals.

Example 1. As an example of the MN-transformation, consider the grammar G with productions

$$\begin{aligned} A &\rightarrow aBa \\ B &\rightarrow bA \mid b \end{aligned}$$

in which A is the start state. This grammar generates the nonregular language $\{(ab)^n a^n \mid n > 0\}$. We can show that the MN-transformation approximates this language by the regular language $(ab)^+a^*$. In G, A and B form a mutually recursive set of nonterminals. The transformed grammar G' consists of the productions

$$\begin{aligned} A &\rightarrow aB \\ A' &\rightarrow B' \mid \varepsilon \\ B &\rightarrow bA \mid bB' \\ B' &\rightarrow aA' \mid \varepsilon\ . \end{aligned}$$

The following derivation in G' simulates the derivation of $ababaa$: $A \rightarrow aB \rightarrow abA \rightarrow abaB \rightarrow ababB' \rightarrow ababaA' \rightarrow ababaB' \rightarrow ababaaA' \rightarrow ababaa$. For the nonterminal B, the newly introduced nonterminal B' serves two purposes:

1. It allows the termination of a derivation from B by replacing B with the terminals that B derives. In our example, $B \rightarrow b$ in G is simulated using the productions $B \rightarrow bB', B' \rightarrow \varepsilon$ from G'.
2. Since the productions are all right–linear, it provides a mechanism to return back to the branching point from the original production and continue the derivation.

However, this last point also introduces ambiguities in the grammar. Nonterminal pairs B and B' mark the beginning and end of strings generated by B in the original grammar. This can be used to compile the transformed grammar into a finite-state transducer that outputs bracketed strings equivalent to parse trees [4]. At the same time by making use of B', it is possible to continue the derivation from the right of B in a current sentential form by any production that has B on its right hand side, not necessarily the next nonterminal in the sentential form (see Example 5).

4 The Automaton for the MN-Approximation

Assume that N itself is a mutually recursive set of nonterminals. The structure of the transition diagram of the automaton constructed from the right-linear grammar G' in the standard way [3] allows us to quickly determine a regular expression For the MN-approximation, especially when the given grammar is in Chomsky Normal Form (CNF).

The transition diagram is organized as two rows of states where each state is labeled with a nonterminal in G', grouped as follows. (see Figure 1 (a) for the automaton corresponding to G' of Example 1.)

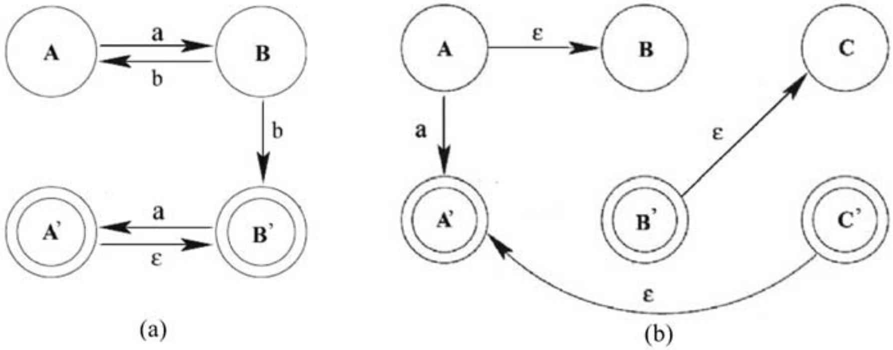


Fig. 1. (a) Automaton corresponding to the transformation of the grammar in Example 1. (b) Transformed CNF rules $A \to BC \mid a$.

- The nonterminals of the original grammar are represented in the upper part.
- The newly introduced nonterminals are represented in the lower part.
- The final states are the states in the lower part of the automaton.
- Every production of the type $A \to w$ induces a transition from the upper part to the lower part. The transition from B to B' in Figure 1 (a) that comes from the rule $B \to b$ demonstrates this.
- For every production of the type $A \to \alpha_0 B_1 \alpha_1 B_2 \alpha_2 \ldots B_m \alpha_m$ in G, the first rule $A \to \alpha_0 B_1$ in G' induces a transition strictly within the upper part of the automaton. The transition from A to B in Figure 1 (a) that comes from the rule $A \to aBa$ in this way demonstrates this.
- The last production in $B'_m \to \alpha_m A'$ in G' induces a transition strictly within the lower part of the automaton. The transition from B' to A' in Figure 1 (a) that comes from $B' \to aA'$ demonstrates this.
- All other intermediate productions in G' induce transitions from the lower part of the automaton to the upper part.

In CNF, the productions of G are of the form $A \to BC$ or $A \to a$. Assuming that A, B, C are all in the same set of mutually recursive nonterminals, the transformation for the above mentioned rules yields:

$$A \to B$$
$$B' \to C$$
$$C' \to A'$$
$$A \to aA' \ .$$

The first production leads to an ε-transition in the upper part of the automaton. The second production leads to an ε-transition from the lower part to the upper part. The third production leads to an ε-transition within the lower part. Its only productions of the fourth kind that actually derive all the terminals, and they result in transitions from the upper part of the automaton to the lower part. This is illustrated in Figure 1 (b).

5 NSE Grammars

We will assume for the rest of the discussion, that G is in CNF and that N is a mutually recursive set.

Recall that G is SE if for some nonterminal A, there is a derivation $A \xrightarrow{*} \alpha A \beta$, with both α, β non–empty. G is NSE if for any nonterminal A and a derivation $A \xrightarrow{*} \alpha A \beta$, either $\alpha = \varepsilon$ or $\beta = \varepsilon$. In general, it is undecidable if a context–free grammar generates a regular language [8], or even if $\mathcal{L}(G) = T^*$. However whether a context–free grammar is NSE is decidable [1]. By Chomsky's result, if G is NSE then $\mathcal{L}(G)$ is regular. Of course this leaves open the possibility that G is SE, but $\mathcal{L}(G)$ is nevertheless regular. The property $A \xrightarrow{*} \alpha A \beta$, with $\alpha, \beta \neq \varepsilon$ enables the grammar to generate terminal strings of the form $u^i x v^i$. If u and v are sufficiently complex, then the language has a counting property and cannot be regular. Therefore the nature of the terminal strings derivable by the self-embedding in G is the thin line that separates the decidable question of "Is G NSE?" and the undecidable question of "Is $\mathcal{L}(G)$ regular?".

We make use of some of the ideas from [1]. Define the edge-colored production graph $CP(G)$ for a grammar G by starting with the nonterminals as vertices. Since G is in CNF, all productions are of the form: $A \to BC$ or $A \to a$. In $CP(G)$, we are only concerned with productions of the form $A \to BC$. For every production $A \to BC$, $CP(G)$ has an edge from node A to node B colored l, and an edge from A to C colored r. We note that in $CP(G)$ self-loops are possible, and if we ignore the colors on the edges, then the graph is strongly connected. Also, an l-colored edge can arise from more than one rule, e.g. $A \to BC \mid BD$. Similarly for r-colored edges. Therefore the number of l-colored edges is not necessarily equal to the number of r-colored edges.

Theorem 1. *G is NSE iff all cycles in $CP(G)$ are monochromatic.*

Proof. Any derivation $A_1 \xrightarrow{*} \alpha A_1 \beta$ in G corresponds to a cycle in $CP(G)$. If the cycle containing A_1 is monochromatic with color l, then this a derivation is of the form $A_1 \to A_2 B_2 \to A_3 B_3 B_2 \to \cdots \to A_k B_k B_{k-1} \cdots B_2 \to A_1 B_1 B_k \cdots B_2$ with $\alpha = \varepsilon$. Similarly, if the cycle containing A_1 is monochromatic with color r, then $\beta = \varepsilon$. Conversely, any cycle with an edge $A_1 \to A_2$ colored l followed by an edge $A_2 \to A_3$ colored r gives a derivation of the form $A_1 \to A_2 B_2 \to B_3 A_3 B_2 \to \cdots \to \alpha A_1 \beta$ where α starts with B_3 and β ends with B_2. Therefore $\alpha, \beta \neq \varepsilon$, and G is SE.

6 Regular Approximation by Cycle-Breaking

Rather than replacing the rules of the grammar with the appropriate approximations, an alternative approach is to only use the approximation for non–monochromatic cycles in $CP(G)$, and leave the rest of the graph intact. We present an example to demonstrate this approach.

Example 2. Let $T = \{a, b\}$ and consider the CFG G:

$$A_1 \to A_2A_3 \mid b,\ A_2 \to A_3A_4,\ A_3 \to A_4A_5,\ A_4 \to A_5A_1,\ A_5 \to A_1A_2 \mid a\ . \quad (1)$$

Applying the MN-transformation, the resulting regular grammar G' is:

$$\begin{array}{l} A'_1 \to \varepsilon, \quad A'_2 \to \varepsilon, \quad A'_3 \to \varepsilon, \quad A'_4 \to \varepsilon, A'_5 \to \varepsilon \\ A_1 \to bA'_1 \\ A_5 \to aA'_5 \\ A_1 \to A_2,\ A'_1 \to A_2, A'_1 \to A'_4 \\ A_2 \to A_3,\ A'_2 \to A_3, A'_2 \to A'_5 \\ A_3 \to A_4,\ A'_3 \to A_4, A'_3 \to A'_1 \\ A_4 \to A_5,\ A'_4 \to A_5, A'_4 \to A'_2 \\ A_5 \to A_1,\ A'_5 \to A_1, A'_5 \to A'_3 \end{array} \quad (2)$$

To get a sense of the approximation, note that G is equivalent to the grammar

$$\begin{aligned} A_1 &\to A_5A_1A_5A_5A_1A_5A_1A_5 \mid b \\ A_5 &\to A_1A_5A_1A_5A_5A_1 \mid a\ , \end{aligned}$$

and in particular $\mathcal{L}(G)$ contains no word of length $2, 3, \ldots, 7$. The automaton corresponding to G' is shown in Figure 2. The language accepted is T^+. Therefore

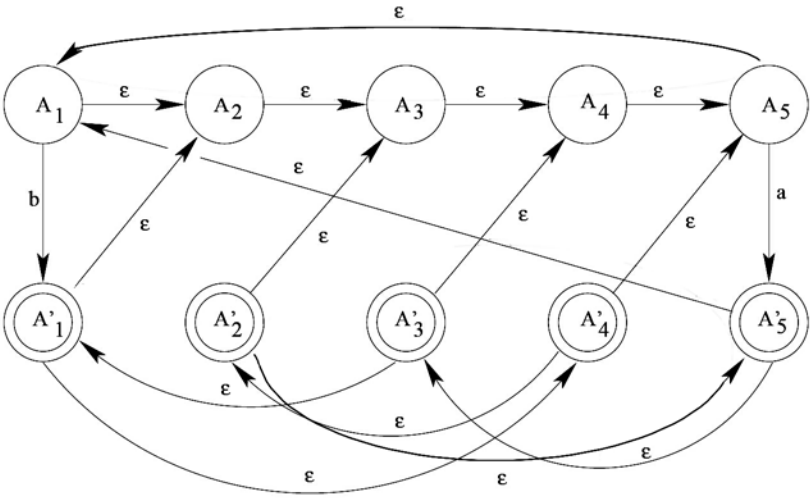


Fig. 2. The automaton for the grammar in Example 2

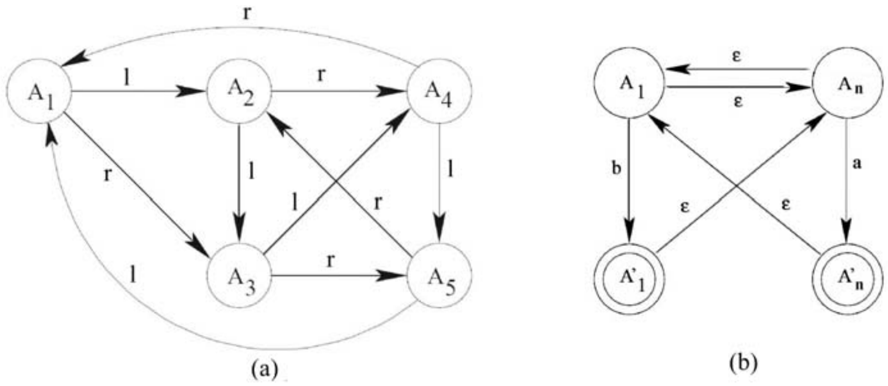


Fig. 3. (a) Edge colored production graph $CP(G)$ for the grammar in Example 2. (b) Transformation of the grammar G_n of Example 4 generates T^+.

the MN-approximation to $\mathcal{L}(G)$ is T^+. We also note from Figure 2 that any nonterminal A_i in this example generates T^+.

For the approximation using cycle-breaking, we first construct $CP(G)$. This is shown in Figure 3 (a) for the grammar in Example 2. Using cycle-breaking, it is possible to devise different regular approximations to $\mathcal{L}(G)$. We can use the MN-approximation itself as a subroutine, for example. Alternatively, we break non–monochromatic cycles by introducing a new nonterminal for each edge eliminated. Depending on what we allow these new nonterminal to derive in the new grammar, we obtain regular approximations that are supersets or subsets of the given language. It is also possible to mix these two ideas.

We consider cycle-breaking by using the MN-transformation first, and then describe cycle-breaking based on introduction of new nonterminals.

6.1 Cycle-Breaking Using the MN-Transformation

To eliminate an l-colored edge $A_i \to A_j$ in $CP(G)$, we proceed as follows. Suppose the A_i-productions of G the form $A_i \to A_j A_k$ are

$$A_i \to A_j A_{k_1} \mid A_j A_{k_2} \mid \cdots \mid A_j A_{k_t} \tag{3}$$

and G' is the grammar obtained from G by the MN-transformation. Make a fresh copy of G' by relabeling each A_k by B_k where a distinct symbol B is is used for every edge eliminated. G'_j be this grammar with start symbol B_j. We replace the rules (3) with

$$A_i \to B_j A_{k_1} \mid B_j A_{k_2} \mid \cdots \mid B_j A_{k_t} \ . \tag{4}$$

Similarly, to eliminate an r-colored edge $A_i \to A_j$, assume that the A_i-rules of the form $A_i \to A_k A_j$ are

$$A_i \to A_{k_1} A_j \mid A_{k_2} A_j \mid \cdots \mid A_{k_t} A_j \ . \tag{5}$$

Let G' be the grammar obtained from G by the MN-transformation. Make a fresh copy of G' by relabeling each A_k by C_k where a distinct symbol C is is used for every edge eliminated. Let G'_j be this grammar with start symbol C_j. Then we replace the rules (5) with

$$A_i \to A_{k_1} C_j \mid A_{k_2} C_j \mid \cdots \mid A_{k_t} C_j \ . \tag{6}$$

Example 3. Let G be the grammar in Example 2. From Figure 3 (b), we see that eliminating the r-colored edges $A_4 \to A_1, A_5 \to A_2$ and the l-colored edge $A_5 \to A_1$ are sufficient to make all cycles monochromatic in $CP(G)$. Using the above idea, we obtain the grammar

$$\begin{aligned}
A_1 &\to A_2 A_3 \mid b\\
A_2 &\to A_3 A_4\\
A_3 &\to A_4 A_5\\
A_4 &\to A_5 B_1\\
A_5 &\to D_1 C_2 \mid a
\end{aligned}$$

together with three copies of the productions in (2), one each for nonterminal B, C and D. From the automaton in Figure 2, we see that every nonterminal in G' derives T^+. For this example, the grammar G'' obtained by cycle-breaking using the MN-transformation generates the language

$$(T^* T^3 + aT^* T)(T^* T^2 + a)(T^* T^3 + aT^* T)^2 (T^* T^2 + a) + b \ , \tag{7}$$

which generates no word of length $2, 3, \ldots, 7$, and is strictly contained in the MN-approximation.

The derivations in the NSE grammar obtained by breaking cycles by using the MN-transformation can be simulated by the NM-transformation of the original grammar. Thus

Theorem 2. *The regular approximation G'' produced by breaking all non–monochromatic cycles of $CP(G)$ using the MN-transformation is finer than the MN-approximation G' of G. In other words $\mathcal{L}(G) \subseteq \mathcal{L}(G'') \subseteq \mathcal{L}(G')$.*

How close is $\mathcal{L}(G'')$ to $\mathcal{L}(G')$? The following example gives an idea.

Example 4. For $n \geq 3$ and $T = \{a, b\}$, consider the grammar G_n with rules $A_1 \to A_2 A_3 \mid b$, $A_2 \to A_3 A_4, \ldots, A_{n-2} \to A_{n-1} A_n$, $A_{n-1} \to A_n A_1$, $A_n \to A_1 A_2 \mid a$. Suppose we obtain G''_n by breaking all l-colored cycles by the MN-transformation, and let G'_n be grammar of the MN-transformation directly applied to G_n. Then $\mathcal{L}(G'_n) = T^+$ regardless of n, whereas

$$\mathcal{L}(G''_n) = \begin{cases} (\varepsilon + T^* T^m) b & \text{if } n = 2m,\\ (\varepsilon + T^* T^n)(b + T^* T^m a) & \text{if } n = 2m+1 \ . \end{cases}$$

The grammar G_n and the approximations given above are considered in detail next.

6.2 Cycle-Breaking Using New Nonterminals

We can simplify the resulting grammar by bypassing the MN-transformation for cycle-breaking. This done at some expense. We still have $\mathcal{L}(G) \subseteq \mathcal{L}(G'')$ but the inclusion $\mathcal{L}(G'') \subseteq \mathcal{L}(G')$ of Theorem 2 is lost.

To eliminate an l-colored edge $A_i \rightarrow A_j$ in $CP(G)$ with new nonterminals, we proceed as follows: Suppose the A_i-rules of G the form $A_i \rightarrow A_j A_k$ are $A_i \rightarrow A_j A_{k_1} \mid A_j A_{k_2} \mid \cdots \mid A_j A_{k_t}$.

1. Replace these by $A_i \rightarrow B_j A_{k_1} \mid B_j A_{k_2} \mid \cdots \mid B_j A_{k_t}$.
2. Add the productions

$$\begin{aligned} &B_j \rightarrow B_a B_j,\ \forall a \in T, \\ &B_a \rightarrow a,\ \forall a \in T, \\ &B_j \rightarrow a,\ \forall a \in T, \end{aligned}$$

   where B_j and B_a, $(a \in T)$ are new nonterminals.

Elimination of an r-colored edge is done similarly. Call the resulting grammar G''. This creates no new non–monochromatic cycles, and the edge $A_i \rightarrow A_j$ in $CP(G)$ has been eliminated in $CP(G'')$. In effect, we are replacing the terminals derivable from A_j for the A_i-rules that involve A_j, by terminals derivable from B_j. We generously made B_j derive all of T^+, so that the language generated by G'' is a superset of the language generated by G. We note that if it is possible to make each B_j derive a regular language that is contained in what A_j derives in G, then cycle-breaking gives a regular approximation to $\mathcal{L}(G)$ from below.

One obvious way to eliminate non–monochromatic cycles is to break all l-colored edges in $CP(G)$. For the example grammar $G = G_5$, we can write the resulting grammar (using T^+ for any nonterminal that now derives only T^+ to simplify notation) by

$$A_1 \rightarrow T^+ A_3 \mid b,\ A_2 \rightarrow T^+ A_4,\ A_3 \rightarrow T^+ A_5,\ A_4 \rightarrow T^+ A_1,\ A_5 \rightarrow T^+ A_2 \mid a$$

so that the approximating language is generated by $A_1 \rightarrow (T^+)^5 A_1 \mid (T^+)^2 a \mid b$. A regular expression for this language is

$$(\varepsilon + T^* T^5)(b + T^* T^2 a)\ . \tag{8}$$

For the same G, eliminating all r-colored edges from $CP(G)$, we obtain the grammar

$$A_1 \rightarrow A_2 T^+ \mid b,\ A_2 \rightarrow A_3 T^+,\ A_3 \rightarrow A_4 T^+,\ A_4 \rightarrow A_5 T^+,\ A_5 \rightarrow A_1 T^+ \mid a$$

which is equivalent to $A_1 \rightarrow A_1 (T^+)^5 \mid a (T^+)^4 \mid b$. Therefore the approximation is given by the regular expression

$$(\varepsilon + T^* T^5)(b + a T^* T^4)\ . \tag{9}$$

In either case, the resulting approximating language is properly contained in the language $(a + b)^+$ of the MN-approximation.

To eliminate non–monochromatic cycles in $CP(G)$, removing all l-colored edges or all r-colored edges may be an overkill. It suffices to eliminate any set of edges with the property that all the cycles in the resulting graph are monochromatic. It would appear that the fewer edges we remove, the closer the approximation is to the original language, because fewer nonterminals are made to derive T^+ instead of what they originally derive in G. However it is possible that the we make more of an error when breaking a short cycle because T^+ may be far from what each of the eliminated nonterminals for this cycle derives, whereas eliminated edges on a long cycle may be each coming from nonterminals that derive languages much closer to T^+.

Continuing with Example 2, eliminating the r-colored edges $A_4 \to A_1, A_5 \to A_2$ and the l-colored edge $A_5 \to A_1$ are sufficient to make all cycles monochromatic in $CP(G)$. The resulting grammar is

$$A_1 \to A_2A_3 \mid b,\ A_2 \to A_3A_4,\ A_3 \to A_4A_5,\ A_4 \to A_5T^+,\ A_5 \to T^+T^+ \mid a\ .$$

This generates the language denoted by the regular expression in (7). The language in (7) is obtained by eliminating 3 edges of $CP(G)$ whereas the regular expressions in (8) and (9) were both obtained by eliminating 5 edges. Now consider the grammar G_n of Example 4.

Lemma 1. *Let $G = G_n$ be the grammar of Example 4. The regular language $\mathcal{L}(G'')$ obtained from G by eliminating l-colored edges in $CP(G)$ is given by*

$$\begin{array}{rl} (\varepsilon + T^*T^m)b & \text{if } n = 2m, \\ (\varepsilon + T^*T^n)(b + T^*T^m a) & \text{if } n = 2m+1\ . \end{array}$$

Proof. By repeated substitutions, we compute that the G'' is equivalent to the grammar

$$\begin{array}{ll} A_1 \to (T^+)^m A_1 \mid b & \text{if } n = 2m, \\ A_1 \to (T^+)^n A_1 \mid (T^+)^m a \mid b & \text{if } n = 2m+1\ , \end{array}$$

from which we obtain

$$\begin{array}{rl} ((T^+)^m)^* b & \text{if } n = 2m, \\ ((T^+)^n)^*(b + (T^+)^m a) & \text{if } n = 2m+1\ . \end{array}$$

From the identities $(T^+)^m = T^*T^m$ and $(T^*T^m)^* = \varepsilon + T^*T^m$, the regular expressions in (1) follow.

A similar result can be obtained for the left-linear grammars constructed by eliminating r-colored edges.

Remark: In the automaton M of the MN-transformation for the grammar G_n of Example 4 contains ε-transitions from A_n to A_1, and A'_n to A_1; ε-paths from A_1 to A_n, and from A'_1 to A_n. Therefore the automaton in Figure 3 (b) sits inside M, and the language accepted is $\mathcal{L}(G') = T^+$. We have $\mathcal{L}(G) \subseteq \mathcal{L}(G'') \subseteq \mathcal{L}(G')$

where $\mathcal{L}(G)$ is the context-free language generated by the grammar $G = G_n$ of Example 4, $\mathcal{L}(G')$ is the regular approximation from the MN-transformation and $\mathcal{L}(G'')$ is the regular approximation obtained by eliminating l-colored edges from $CP(G)$. Since $\mathcal{L}(G') = T^+$ independently of n whereas $\mathcal{L}(G'')$ is given as in Lemma 1, the difference between the former approximation and the latter can be made as large as we please.

7 Using a 1–Lookahead

In the MN-approximation, there is a certain memory in the rules carried by symbols such as A'_i which allow us to continue parsing from where we left off. We can remember more of the context of the branching by using a type of lookahead. This removes some of the ambiguity and therefore result in a smaller regular approximation, but it is at the cost of increasing the size of the new grammar. Mohri and Nederhof's grammar has size $O(|G|)$. The approximating grammar we obtain by eliminating all l-colored or all r-colored edges in $CP(G)$ in cycle-breaking is also $O(|G|)$. The lookahead considered here has $O(|G|^2)$ productions, as the approximating grammar construction in [6] (see also [4]). A k–lookahead approximation will cost $O(|G|^k)$ nonterminals, probably an unrealistically large bound of theoretical interest only.

For simplicity, in this section we consider the grammar to be in Greibach Normal Form (GNF). In GNF, all productions are of the form $A \to aA_1A_2 \ldots A_n$ or $A \to b$. In *1–lookahead*, we introduce new nonterminals for *pairs* of consecutive nonterminals that appear on the right hand side of a production with nonterminals. For a generic GNF production with a nonterminal right hand side, these would be $A_{12}, A_{34}, \ldots$. The idea is to preserve the memory of the production from where a branch occurred so that the derivation can continue if the next nonterminal is also present. This memory is at the odd indices only since we do not remember A_2A_3, for example. We will demonstrate the 1–lookahead idea with the help of an example.

Example 5. Let $T = \{a, b\}$ and start with the following grammar G:

$$\begin{aligned} A_1 &\to aA_2A_2 \mid a \\ A_2 &\to bA_2A_1 \mid b \end{aligned}$$

Straightforward MN-transformation results in the right-linear grammar

$$\begin{array}{ll} A'_1 \to \varepsilon & A'_2 \to \varepsilon \\ A_1 \to aA'_1 & A_2 \to bA'_2 \\ A_1 \to aA_2 & A_2 \to bA_2 \\ A'_2 \to A_2 & A'_2 \to A_1 \\ A'_2 \to A'_1 & A'_1 \to A'_2 \end{array} \tag{10}$$

In a leftmost derivation from A_1, after the first A_2 is processed, the end marker A'_2 allows for the derivation to continue with A_2, but also with A_1. In the approximation with 1–lookahead we remember that the current A'_2 should be followed by processing A_2 and not A_1. This can be achieved by using the MN-transformation in the following way. Start with the first production above and introduce new nonterminals A_{22} and B_2 to indicate that the continuation is by A_2. We change the original production $A_1 \rightarrow aA_2A_2$ by using A_{22} for the first A_2 and using B_2 for the second A_2 as $A_1 \rightarrow aA_{22}B_2$, and then apply the MN-transformation. The first column in (10) is replaced by the rules

$$A_1' \rightarrow \varepsilon,\ A_1 \rightarrow aA_1',\ A_1 \rightarrow aA_{22},\ A_{22}' \rightarrow B_2,\ B_2' \rightarrow A_1'\ . \tag{11}$$

The second column of rules in (10) becomes

$$A_{22}' \rightarrow \varepsilon,\ A_{22} \rightarrow bA_{22}',\ A_{22} \rightarrow bA_{22},\ A_{22}' \rightarrow A_1,\ A_1' \rightarrow A_{22}'\ . \tag{12}$$

The new set of rules in (11) and (12) are the 1–lookahead transformation of the production $A_1 \rightarrow aA_2A_2$ of the original grammar.

For the transformation of the the second production $A_2 \rightarrow bA_2A_1$ we introduce two new nonterminals A_{21} and B_1 to indicate that the continuation is by A_1. We change $A_2 \rightarrow bA_2A_1$ by using A_{21} instead of A_2 and using B_1 instead of A_1, and write $A_2 \rightarrow bA_{21}B_1$. Applying the MN-transformation to this has the effect of replacing the the second column of (10) by the rules

$$A_2' \rightarrow \varepsilon,\ A_2 \rightarrow bA_2',\ A_2 \rightarrow bA_{21},\ A_{21}' \rightarrow B_1,\ B_1' \rightarrow A_2' \tag{13}$$

and replacing the first column of (10) by

$$B_1' \rightarrow \varepsilon,\ B_1 \rightarrow aB_1',\ B_1 \rightarrow A_{21},\ A_{21}' \rightarrow A_{21},\ A_{21}' \rightarrow B_1'\ . \tag{14}$$

The rules in (13) and (14) are the 1–lookahead transformation of the production $A_2 \rightarrow bA_2A_1$ of the original grammar.

Finally, we allow B_1 and B_2 derive the same sentential forms as A_1 and A_2 in the MN-approximation (10) by making copies of these rules using Bs for the corresponding As:

$$\begin{array}{ll} B_1' \rightarrow \varepsilon & B_2' \rightarrow \varepsilon \\ B_1 \rightarrow aB_1' & B_2 \rightarrow bB_2' \\ B_1 \rightarrow aB_2 & B_2 \rightarrow bB_2 \\ B_2' \rightarrow B_2 & B_2' \rightarrow B_1 \\ B_2' \rightarrow B_1' & B_1' \rightarrow B_2'\ . \end{array} \tag{15}$$

Let G' denote the MN-transformation of the given CFG G and denote by G'' the grammar obtained from G by the 1–lookahead transformation. Since any derivation in the 1–lookahead grammar can be simulated by a derivation in the original MN-transformation of G, we have

Theorem 3. *Let G' denote the MN-transformation of the CFG G and G'' the grammar obtained from G by the 1–lookahead transformation. Then $\mathcal{L}(G) \subseteq \mathcal{L}(G'') \subseteq \mathcal{L}(G')$.*

For the grammar G in Example 5, the MN-transformation G' is given by (10). In G', A_2 derives $b(a+b)^*$ and the MN-approximation itself is given by $\mathcal{L}(G') = a(a+b)^*$. The 1–lookahead transformation gives the grammar G'' with $\mathcal{L}(G'') = a + ab(a+b)^*$, which is properly contained in $\mathcal{L}(G')$.

8 Summary and Remarks

We considered the problem of approximation of a given context-free grammar by a regular grammar while trying to preserve the structure of the original grammar as much as possible. The algorithms considered are improvements on Mohri and Nederhof's original transformation and make use of the characterization of non–self-embedding grammars as generating regular languages.

In the approximations based on cycle-breaking, we start with a grammar in Chomsky normal form as input, and provide a regular grammar as output. The language generated is a superset of the given language, and a subset of the original Mohri and Nederhof approximation. We also consider a lookahead transformation which starts with the Greibach normal form and produces a regular grammar as its output. This approximation is also a superset of the given language, and a subset of the Mohri and Nederhof approximation.

References

1. Anselmo, M., Giammarresi, D., Varricchio, S.: Non–self–embedding grammars as representation for regular languages. In: CIAA Conference Proceedings (2002)
2. Chomsky, N.: A note on phrase structure grammars. Information and Control 4(2), 386–392 (1959)
3. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Publishing Co., Reading (1979)
4. Mohri, M., Nederhof, M.-J.: Regular approximation of context–free grammars through transformation. In: Robustness in Language and Speech Technology, vol. 9, pp. 251–261. Kluwer Academic Publishers, Dordrecht (2000)
5. Mohri, M., Pereira, F.N.: Dynamic compilation of weighted context–free grammars. In: 36th Annual Meeting of the ACL and 17th International Conference on Computational Linguistics, vol. 2, pp. 891–897 (1998)
6. Nederhof, M.-J.: Regular approximation of cfls: A grammatical view. In: International Workshop on Parsing Technologies, pp. 159–170. MIT Press, Cambridge (1997)
7. Shallit, J.: Automaticity and rationality. J. of Automata, Languages and Combinatorics 5(3), 255–268 (2000)
8. Ullian, J.S.: Partial algorithm problems for context free languages. Information and Control 11, 80–101 (1967)

Powers of Regular Languages

Szilárd Zsolt Fazekas

Research Group in Mathematical Linguistics
Universidad Rovira i Virgili
szilard.zsolt@estudiants.urv.cat

Abstract. In this paper we prove that it is decidable whether the set pow(L), which we get by taking all the powers of all the words in some regular language L, is regular or not. The problem was originally posed by Calbrix and Nivat in 1995. Partial solutions have been given by Cachat for unary languages and by Horváth et al. for various kinds of exponent sets for the powers and regular languages which have primitive roots satisfying certain properties. We show that the regular languages which have a regular power are the ones which are 'almost' equal to their Kleene-closure.

1 Introduction

Calbrix and Nivat defined the power pow(L) of a language L in a paper about prefix and period languages of rational ω-languages [3]:

$$\mathrm{pow}(L) = \{w^i \mid w \in L, i \geq 1\}.$$

It is easy to see that there are examples of regular languages L for which pow(L) is regular, and examples for which pow(L) is not regular. Take, for instance, the regular language ab^* whose power is not even context-free. Calbrix and Nivat posed the problem of characterizing those regular languages whose powers are also regular, and to decide whether a given regular language has this property. They conjectured that "rational languages such that their power is also rational are 'almost' a union of rational subsemigroups of Σ^* and the point is to give the right sense to this almost". Cachat [2] gave a partial solution to this problem showing that for a regular language L over a one-letter alphabet, it is decidable whether pow(L) is regular. Unfortunately Cachat's result cannot be extended to arbitrary alphabets since he translated unary languages into sets of natural numbers to reach his solution. Horváth et al. [5] provided more partial results. They looked at arbitrary alphabets and did a case analysis based on the primitive roots of the regular languages in question. However, the case when $L \cap \sqrt{L}$ is not regular and $\sqrt{\mathrm{pow}(L) \setminus L}$ is finite was left open, and thus a complete solution to the original problem was not given. They also considered other exponent sets instead of the whole set of natural numbers and an algorithm based on [2] to decide whether the power of a regular language with finite primitive root is regular or not. We will use the results of [5] (and [2] resp.) as part of the

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 221–227, 2009.

decision procedure. There exist several other papers that study regular languages containing powers of their words or consisting solely of non-primitive words, see [4,6]. Recently Anderson et al. [1] presented a characterization of regular languages consisting only of powers.

In this paper we characterize the regular languages whose power is also regular. First we present a short overview of the notions and results needed to proceed with the paper, and then we go on to solve the decidability problem posed by Calbrix and Nivat by reducing it to decidable subproblems.

2 Preliminaries

In this section we briefly recall some definitions and known results needed throughout the rest of the paper. Let Σ be a fixed finite nonempty alphabet. By Σ^* we mean the free monoid generated by Σ, that is the set of all words over Σ. The empty word we denote by λ, and $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. A language over Σ is a subset L of Σ^*. For a word $p \in \Sigma^*$, $|p|$ denotes the length of p, and for a set M, $|M|$ denotes the cardinality of M. For a natural number k, p^k denotes the concatenation of k copies of the word p, and $p^0 = \lambda$. As usual, p^* denotes the set $\{p^k : k \geq 0\}$, and p^*q the set $\{p^kq : k \geq 0\}$. For two words $u, v \in \Sigma^*$ and a language $L \subseteq \Sigma^*$, by saying that $u \equiv v(P_L)$ we mean the following

$$xuy \in L \text{ if and only if } xvy \in L \text{ for all } x, y \in \Sigma^*.$$

For a word $w \in \Sigma^*$ the congruence class $[w]_{P_L}$ consists of all words congruent with w according to P_L, that is $[w]_{P_L} = \{v \in \Sigma^* \mid w \equiv v(P_L)\}$. Since P_L is a congruence relation, $\Sigma^*/P_L = \{[w] \mid w \in \Sigma^*\}$ is a monoid, which is called the *syntactic monoid* of L and denoted by $\text{Synt}(L)$.

A *non-deterministic finite automaton* (NFA) is a quintuple $\mathcal{A} = \{\Sigma, Q, I, F, \sigma\}$ with the usual conventions, i.e. Σ is the input alphabet, Q is the set of states, I is the set of initial states, F is the set of final states and $\sigma : Q \times \Sigma \to 2^Q$ is the transition function. We will use σ as an extended transition function taking words as second arguments instead of only letters, interpreted as follows $\sigma(p, ab) = \{q \in \sigma(q_1, b) \mid q_1 \in \sigma(p, a)\}$. A finite automaton is *deterministic* (DFA) if I is a singleton and $p, q \in \sigma(r, w)$ implies $p = q$.

When talking about regular languages we will often mean languages accepted by some finite automaton or languages which are unions of some of the equivalence classes of a congruence relation of finite index. As it is well known from Rabin and Scott [8] these language classes are in fact the same. We note here that the size of the syntactic monoid for a regular language L is at most $2^{|Q|^2}$, if Q is the set of states of some NFA accepting L.

Definition 1. *A word p is primitive if there is no word $q \neq p$ and no positive integer n such that $p = q^n$. We denote the language of all primitive words over a given alphabet Σ by Q.*

Definition 2. *The (primitive) root of a word $p \in \Sigma^+$ is the unique primitive word q such that $p = q^n$ for some $n \geq 1$. $\sqrt{p}$ denotes the root of p. For a language L, $\sqrt{L} = \{\sqrt{p} : p \in L \wedge p \neq \lambda\}$ is the root of L.*

The next result is known as the theorem of Fine and Wilf. Intuitively it tells us how far two periodic events (strings) have to match in order to guarantee a common period, that is to guarantee that the two sequences are ultimately the same.

Theorem 1. *Let $x, y \in \Sigma^*$, $n = |x|$, $m = |y|$, $d = \gcd(n, m)$. If two powers x^p and y^q of x and y have a common left factor of length at least equal to $n+m-d$, then x and y are powers of the same word.*

The following is a well known theorem by Lyndon and Schützenberger.

Theorem 2. *[7] If $u^m v^n = w^k$ for non-empty words u, v, w and natural numbers $m, n, k \geq 2$ then $\sqrt{u} = \sqrt{v} = \sqrt{w}$.*

This is a basic property of words that will prove useful to us later on through a theorem by Shyr and Yu, which we present here with a short proof to better illustrate our argument.

Corollary 1. *[9] Let u, v be primitive words such that $u \neq v$. Then there is at most one non-primitive word in u^+v^+.*

Proof. Let $w = u^m v^n$ be non-primitive. Then either $m = 1$ or $n = 1$ by Theorem 2. So, by symmetry let $uv^n = w^i$ for some primitive word w and $i \geq 2$. We may choose n to be minimal with that property. It is enough to show that all uv^{n+k} are primitive. By contradiction, suppose that uv^{n+k} is not primitive for some $k \geq 1$, that is there exists some $z \in Q$ and $j \geq 2$ such that $z^j = uv^{n+k}$. It follows that $w^i v^k = z^j$.

First consider the case $k \geq 2$. Since $i, j, k \geq 2$, we can apply the Lyndon-Schützenberger theorem and get that $\sqrt{w} = \sqrt{v} = \sqrt{z}$, but then $u = v$, a contradiction.

Now let us see the case $k = 1$. Non-primitivity is invariant to cyclic shifts, so w^i and z^j being non-primitive gives us that $v^n u$ and $v^n uv$ are non-primitive too. Hence there are words $w_1, z_1 \in Q$ such that $w_1^i = v^n u$ and $z_1^j = v^n uv$, moreover $|w_1| = |w|$ and $|z_1| = |z|$. From here $z_1^j = w_1^i v$. As v is a prefix of w_1^i, we have that z_1^j has both periods $|z_1|$ and $|w_1|$. Now we can apply the theorem of Fine and Wilf and get that $z_1 = w_1 = v$. This means $w = z = v$ and then $u = v$, a contradiction again. □

Corollary 2. *For all words $x, y, z \in \Sigma^*$ with $y \neq \lambda$ with $|\sqrt{xyz}| \neq |\sqrt{y}|$, there is at most one non-primitive word in the language xy^+z.*

Proof. Suppose there exist numbers $i, j \geq 1$ with $i < j$ such that both xy^iz and xy^jz are non-primitive. Non-primitivity is invariant to cyclic shifts, so zxy^i and zxy^j are non-primitive too.

If zx is non-primitive then we can apply the Lyndon-Schützenberger theorem on zxy^j and get that $\sqrt{zx} = \sqrt{y}$. This would mean $\sqrt{zxy} = \sqrt{y}$ and from here $|\sqrt{xyz}| = |\sqrt{y}|$, a contradiction.

If zx is primitive then from Theorem 1 we have that only one of the words zxy^i and zxy^j is non-primitive, contradicting our original assumption. □

3 The Power of a Regular Language

There are easy examples for non-trivial regular languages that do not have a regular power. Besides the one mentioned in the introduction one could take $aaa(aa)^*$ whose power $\{a^k : k \text{ is not a power of } 2\}$ is not even context-free (in particular, powers of regular languages are not semi-linear, in general).

In fact, as it turns out, it is quite difficult to come up with examples of regular languages L with regular power other than the ones for which $L = L^*$ or $L = L^* \setminus K$, where either K is finite or $K = \bigcup_{w \in S} w^*$ for some finite set of words S. This seems to justify the conjecture formulated by Calbrix and Nivat cited before. Hence, rather than trying to solve the case left open in [5] one might try a new approach.

Indeed, as we will shortly see, we can give an equivalent criterion for a regular language to have a regular power, i.e., we can now give sense to that 'almost'.

Theorem 3. *Let L be a regular language. Then $pow(L)$ is regular if and only if* $\mathrm{pow}(L) \setminus L$ *is a regular language such that its primitive root is a finite language.*

Proof. The class of regular languages is closed under union and taking the difference of two sets, therefore if $\mathrm{pow}(L) \setminus L$ is a regular language then so is $(\mathrm{pow}(L) \setminus L) \cup L = pow(L)$.

Now let us look at the "only if" part. If $\mathrm{pow}(L)$ is regular then so is $L_{\mathrm{diff}} = \mathrm{pow}(L) \setminus L$. Note that L_{diff} consists solely of non-primitive words. Let n be the number of states of the minimal deterministic automaton accepting L_{diff}. Now suppose that $\sqrt{L_{\mathrm{diff}}}$ is infinite. In this case there must be some $w \in L_{\mathrm{diff}}$ such that $|\sqrt{w}| > n$. On the other hand the pumping property of regular languages tells us that $w = xyz$ for some $xz, y \notin \{\lambda\}$ with $|y| \le n$ such that $xy^iz \in L_{\mathrm{diff}}$ for all $i \ge 0$, so xy^iz is non-primitive for all i. Corollary 2 says that in this case $|\sqrt{xyz}| = |\sqrt{y}| \le |y| \le n$, contradicting the assumption $|\sqrt{w}| > n$. □

Lemma 1. *Let L be a regular language given by an NFA having n states. If $pow(L)$ is regular, then we have*

$$pow(L) \subseteq L \cup \{\sqrt{u}^i \mid u \in L \wedge |u| \le \max(n^2, m) \wedge i \ge 1\},$$

where m is the size of $\mathrm{Synt}(L)$*.*

Proof. We have seen in Theorem 3 that $pow(L)$ being regular means that it has to be a subset of $L \cup \bigcup_{u \in U} u^+$ for some finite set U of words. We need to prove that for every $w \in L_{\mathrm{diff}}$ there is a $u \in L$ such that $w \in \sqrt{u}^+$ and $|u| \le \max(n^2, m)$.

Take the shortest $u \in L$ such that w is a power of u. If $|u| > \max(n^2, m)$ then according to the pumping property of regular languages u can be written as xyz for some $y \neq \lambda \neq xz$ such that $xy^jz \in L$ for all $j \ge 0$. Here we can distinguish two cases.

1. If $|y|$ is a multiple of $|\sqrt{u}|$ then $|\sqrt{u}| \leq n$. As $u \in \sqrt{u}^+ \cap L$ we can apply the pumping argument on powers of $\sqrt{u}$ as if it was a unary language. If k is the smallest number for which $\sqrt{u}^k \in L$ then $k \leq n$, or else there would be some numbers p, q with $p < q < k$ such that from the initial state we reach the same state by reading $\sqrt{u}^p$ or $\sqrt{u}^q$, and we could cut out $\sqrt{u}^{q-p}$ from the word. From here we get that there is a word $\sqrt{u}^k \leq n^2$ having the same root as w.
2. We are left with the case when in any decomposition $u = xyz$, $|y|$ is not a multiple of $|\sqrt{xyz}|$ and $|u| > m$. Then we find a decomposition $u = xyz$ with $0 < |y| \leq m$ and $[xy] = [x]$ in $\mathrm{Synt}(L)$. This way we know that $xy^jz \in L$, for all $j \geq 1$. As a consequence of Corollary 2 we also know that at most one of these xy^jz can be a non-primitive word. At the same time L_{diff} has finite root, hence for all but finitely many values of j, $(xy^jz)^+ \subseteq L$, so we find some j such that $xy^jz \in L$ and xy^jz is primitive and at the same time $(xy^jz)^+ \subseteq L$. Due to $[xy] = [x]$ in $\mathrm{Synt}(L)$ we can conclude $(xyz)^+ \subseteq L$. However, we supposed that $w \in L_{\mathrm{diff}}$ is some power of xyz, a contradiction.

So for every $w \in L_{\mathrm{diff}}$ there is some $u \in L$, with $|u| \leq \max(n^2, m)$ such that $\sqrt{w} = \sqrt{u}$ and this concludes the proof. □

To make it easy to see why the latter half of the previous theorem can be checked effectively, we should replace $\max(n^2, m)$ with a bound depending only on the number of states n of the automaton accepting L.

Remark 1. Let L be a regular language given by an NFA having n states. If $pow(L)$ is regular, then we have

$$pow(L) \subseteq L \cup \{u^i \mid u \in L \wedge |u| \leq 2^{n^2} \wedge i \geq 1\},$$

where m is the size of $\mathrm{Synt}(L)$.

Proof. This is clear because $n^2 < 2^{n^2}$ and the syntactic monoid is a divisor of the monoid of Boolean $n \times n$ matrices, so $\mathrm{Synt}(L)$ has size at most 2^{n^2}. □

Let us recall the following result from the paper by Calbrix and Nivat about languages which are equal to their power.

Lemma 2. *[3] Let L be a regular language of Σ^*. Then* $\mathrm{pow}(L) = L$ *if and only if there are regular languages $(L_i)_{1 \leq i \leq n}$ such that $L = \bigcup_{i=1}^n L_i^+$.*

The statement above is useful for testing if a language is equal to its power or not, we only need to specify the languages L_i for an effective construction. Using the syntactic monoid of L gives us the tool we need. We can translate $pow(L) = L$ into the following statement involving the congruence classes of P_L:

$$\bigcup_{u \in L} [u]^+ \subseteq L = \bigcup_{u \in L} [u] \subseteq \bigcup [u]^+.$$

Given an automaton accepting L we can effectively construct its syntactic monoid and from here we can effectively define the set of words in the class

$[u]$ for all $u \in L$. In the case of a regular language P_L induces a finite number of classes, so we can decide whether the equality holds or not. Hence, we can state the following.

Proposition 1. *For a regular language L it is decidable whether $pow(L) = L$ holds or not.*

Now we are ready to proceed with the algorithm. Theorem 3 reduces the original problem to an equivalent one of deciding whether the language, in some sense, lacks only a "few" words to be equal to its power. Lemma 1 provides the means to find those "few" missing words and after adding them to our starting language Proposition 1 will tell us whether the result is a power or not, that is whether the power of the original language is regular or not.

Theorem 4. *For a regular language L it is decidable whether* $\mathrm{pow}(L)$ *is regular.*

Proof. We propose the following algorithm:

```
 1. Input: an NFA $\mathcal{A} = \{\Sigma, Q, I, F, \sigma\}$.
 2. Output: "YES", if $\mathrm{pow}(L(\mathcal{A}))$ is regular, and "NO" otherwise.
 3. $U = \emptyset$
 4. FOR all words $w \in L(\mathcal{A})$ shorter than $2^{|Q|^2}$:
 5. —IF $w^* \setminus L(\mathcal{A}) \neq \emptyset$ THEN:
 6. ——IF $\mathrm{pow}((\sqrt{w})^* \cap L(\mathcal{A}))$ is regular THEN add $w$ to $U$
 7. ——ELSE output "NO"
 8. compute the syntactic monoid for $L' = L(\mathcal{A}) \setminus \bigcup_{u \in U} (\sqrt{u})^*$
 9. IF $L' = \mathrm{pow}(L')$ then output "YES"
10. ELSE output "NO"
```

The enumeration of words in $L(\mathcal{A})$ shorter than $2^{|Q|^2}$ can be done in finite time due to the length limit. The condition in line 5 can be checked effectively too. First we have to perform the difference of two regular languages, then check whether the result is empty or not. As it is stated in [5], the condition in line 6 can be verified using Cachat's algorithm ([2]), because $(\sqrt{w})^* \cap L(\mathcal{A})$ is isomorphic to a unary language, which can be computed effectively. In step 8 we have to compute the syntactic monoid for a regular language, which is the difference of a regular language and the finite union of some regular languages, all effectively presented. If a regular language L is equal to $M \cup N$ for some regular languages M and N, such that $\sqrt{M} \cap \sqrt{N} = \emptyset$, then from the closure properties of the regular class we get that $\mathrm{pow}(L)$ is regular if and only if both $\mathrm{pow}(M)$ and $\mathrm{pow}(N)$ are regular. Moreover, L and M being powers implies N being a power as well. Therefore, in step 9 we only need to check whether a regular language is equal to its power or not; by Proposition 1 this is decidable too. Hence, the algorithm terminates after finitely many steps; however, the complexity is at least exponential due to both Cachat's algorithm and the exponential length bound on the words we need to check in step 4. □

4 Conclusion

We managed to characterize regular languages that have regular power following the conjecture of Calbrix and Nivat formulated in [3] and we gave an effective albeit inefficient procedure to decide this property. Although the decision procedure is not a direct extension of previous results ([2,5]), Cachat's algorithm is needed in an essential step, which identifies those "few words" in pow(L) missing from L.

Acknowledgements

The author would like to thank the referees and, in particular, Volker Diekert for their suggestions on correcting the proofs and improving the overall presentation of the paper.

References

1. Anderson, T., Rampersad, N., Santean, N., Shallit, J.: Finite Automata, Palindromes, Powers, and Patterns. In: Martín-Vide, C., Otto, F., Fernau, H. (eds.) LATA 2008. LNCS, vol. 5196, pp. 52–63. Springer, Heidelberg (2008)
2. Cachat, T.: The Power of One-Letter Rational Languages. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 145–154. Springer, Heidelberg (2002)
3. Calbrix, H., Nivat, M.: Prefix and Period Languages of Rational omega-Languages. In: Dassow, J., Rozenberg, G., Salomaa, A. (eds.) Developments in Language Theory 1995, pp. 341–349. World Scientific, Singapore (1996)
4. Dömösi, P., Horváth, G., Ito, M.: A small hierarchy of languages consisting of non-primitive words. Publ. Math. Debrecen 64(3-4), 261–267 (2004)
5. Horváth, S., Leupold, P., Lischke, G.: Roots and Powers of Regular Languages. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 220–230. Springer, Heidelberg (2003)
6. Lischke, G.: The root of a language and its complexity. In: Kuich, W., Rozenberg, G., Salomaa, A. (eds.) DLT 2001. LNCS, vol. 2295, pp. 272–280. Springer, Heidelberg (2002)
7. Lyndon, R.C., Schützenberger, M.P.: On the equation $a^M = b^N c^P$ in a free group. Michigan Math. Journ. 9, 289–298 (1962)
8. Rabin, M.O., Scott, D.: Finite automata and their decision problems. IBM J. Res. Develop. 3, 114–125 (1959)
9. Shyr, H.J., Yu, S.S.: Non-primitive words in the language p^+q^+. Soochow J. Math. 20, 535–546 (1994)

Existence and Nonexistence of Descriptive Patterns

Dominik D. Freydenberger[1] and Daniel Reidenbach[2,⋆]

[1] Institut für Informatik, Goethe-Universität, Postfach 111932,
D-60054 Frankfurt am Main, Germany
freydenberger@em.uni-frankfurt.de

[2] Department of Computer Science, Loughborough University,
Loughborough, Leicestershire, LE11 3TU, United Kingdom
D.Reidenbach@lboro.ac.uk

Abstract. In the present paper, we study the existence of descriptive patterns, i. e. patterns that cover all words in a given set through morphisms and that are optimal in terms of revealing commonalities of these words. Our main result shows that if patterns may be mapped onto words by arbitrary morphisms, then there exist infinite sets of words that do not have a descriptive pattern. This answers a question posed by Jiang, Kinber, Salomaa, Salomaa and Yu (*International Journal of Computer Mathematics* 50, 1994). Since the problem of whether a pattern is descriptive depends on the inclusion relation of so-called pattern languages, our technical considerations lead to a number of deep insights into the inclusion problem for and the topology of the class of terminal-free E-pattern languages.

1 On Patterns Descriptive of a Set of Strings

A *pattern* is a finite string that consists of variables taken from an alphabet X and terminal symbols taken from an alphabet Σ. For any pattern α and any word w over Σ, α is said to cover w if w can be obtained from α by substituting the variables with appropriate strings of terminal symbols. Whenever α contains several occurrences of the same variable, the substitution of variables needs to be "uniform", i. e. each of the occurrences must be replaced with the same word over Σ. Therefore, and more formally, such a substitution is simply a terminal-preserving morphism $\sigma : (\Sigma \cup X)^* \rightarrow \Sigma^*$, i. e. a morphism that satisfies $\sigma(a) = a$ for every terminal symbol a in the pattern. For instance, the pattern $\alpha := xy\mathtt{b}x\mathtt{a}$ (where x, y are variables and $\mathtt{a}, \mathtt{b}$ are terminal symbols) covers the word $w_1 :=$ $\mathtt{abababa}$ since there is a substitution σ, given by $\sigma(x) := \mathtt{ab}$ and $\sigma(y) := \mathtt{a}$, satisfying $\sigma(\alpha) = w$. In contrast to this, α does not cover, e. g., $w_2 := \mathtt{bbbbaa}$.

Due to the simplicity of the concepts involved, the above described notion of a pattern is studied in a variety of fields of research. The present paper mainly deals with two quite closely related approaches: Firstly, a pattern α over $\Sigma \cup X$ can be

[⋆] Corresponding author.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 228–239, 2009.

regarded as a generator of a formal language $L(\alpha)$, the so-called *pattern language*, which simply comprises all words in Σ^* that can be obtained from the pattern by arbitrary substitutions. Secondly, for any given finite or infinite language S, patterns can be used to approximate S; i. e., a pattern α is sought that is *consistent* with S (which means that α covers all words in S or alternatively, in terms of pattern languages, $L(\alpha) \supseteq S$). The latter concept is motivated by the fact that if a pattern is consistent with a language S, then this pattern reveals a common structure of the strings in S. Hence, and since they are compact devices that can be easily read and interpreted by humans, patterns can be very helpful when commonalities of data represented by strings are analysed.

The characteristics of pattern languages have been intensively studied in the past decades. Therefore, quite a number of basic properties of pattern languages, e. g. regarding the usual decision problems for classes of formal languages, are known (cf. the surveys by Mateescu and Salomaa [7] and Salomaa [11] and our recent paper [4]). Furthermore, pattern languages have been a focus of interest of inductive inference from the very beginning, investigating whether it is possible to infer a pattern from the words in its pattern language (see Ng and Shinohara [8]). It is quite remarkable that many of the corresponding results in language theory and inductive inference differ for the two main types of pattern languages that are normally considered, namely the *NE*-pattern language of a pattern (introduced by Angluin [1]), which merely consists of those words in Σ^* that can be obtained from the pattern by *nonerasing* substitutions (i. e. substitutions that do not replace any variables with the empty word), and the *E*-pattern language (established by Shinohara [12]), which additionally comprises those words that can be derived from the pattern by substituting the empty word for arbitrary variables.

The problem of finding a consistent pattern for an arbitrary set S of strings is often referred to as *(string) pattern discovery*, and many of its applications are derived from tasks in bioinformatics (cf. Brazma et al. [2]). In contrast to the inductive inference approach to pattern languages, where a pattern shall be inferred that exactly describes the given language, string pattern discovery faces the problem that S can typically have many consistent patterns showing very different characteristics. For instance, both

$$\alpha_1 := xyxyx \text{ and } \alpha_2 := x\texttt{ab}y$$

are consistent with the language

$$S_0 := \{\texttt{ababa}, \texttt{ababbababbab}, \texttt{babab}\},$$

and the pattern $\alpha_0 := x$ is consistent with every set of strings, anyway. Hence, the algorithms of string pattern discovery require an underlying notion of the quality of a pattern in order to determine what patterns to strive for. With regard to the above example set and patterns, it seems quite likely that one might not be interested in a procedure outputting α_0 when reading S_0. Concerning α_1 and α_2, however, it is, a priori, by no means evident which of them to prefer. Thus, the definition of the quality of a pattern might often depend on the field of application where string pattern discovery is conducted. In addition to this, it is a worthwhile goal to develop *generic* notions of quality of consistent patterns that can inform the design of pattern discovery algorithms.

In this regard, the *descriptiveness* of patterns is a well-known and plausible concept, that is also used within the scope of inductive inference (cf. Ng and Shinohara [8]). A pattern δ is said to be descriptive of a given set S of strings if there is no pattern α satisfying $S \subseteq L(\alpha) \subset L(\delta)$. Intuitively, this means that if δ is descriptive of S, then no consistent pattern for S provides a strictly closer match than δ. Thus, although δ does not need to be unique (as to be further discussed below), it is guaranteed that it is one of the most accurate approximations of S that can be provided by patterns. While descriptiveness is unquestionably an appropriate notion of quality of consistent patterns, it leads to major technical challenges, as its application requires insights into the inclusion problem for pattern languages, which is known to be undecidable in the general case and still combinatorially involved for some major natural subclasses where it is decidable. This aspect is crucial to the subsequent formal parts of our paper.

Since the definition of a descriptive pattern is based on the concept of pattern languages, the question of whether NE- or E-pattern languages are chosen can have a significant impact on the descriptiveness of a pattern. This is reflected by the terminology we use: we call a pattern δ an NE-descriptive pattern if it is descriptive in terms of its NE-pattern language and the NE-pattern languages of all patterns in $(\Sigma \cup X)^+$; accordingly, we call δ E-descriptive if its descriptiveness is based on interpreting all patterns as generators of E-pattern languages. In order to illustrate these terms, we now briefly discuss the descriptiveness of the example patterns introduced above (though the full verification of our corresponding claims is not always straightforward and might require certain tools to be introduced later). If we deal with S_0 and the patterns in the context of NE-pattern languages, then it can be stated that both α_1 and α_2 are NE-descriptive of S_0, since no NE-pattern languages can comprise S_0 and, at the same time, be a proper sublanguage of the NE-pattern languages of α_1 or α_2. If we study S_0 in terms of E-pattern languages, it turns out that α_1 is also E-descriptive of S_0, i. e. there is no pattern generating an E-pattern language that is consistent with S_0 and strictly included in the E-pattern language of α_1. However, the second NE-descriptive example pattern α_2 is not E-descriptive of S_0, since the E-pattern language generated by

$$\alpha_3 := x\mathtt{abab}y$$

is a proper sublanguage of the E-pattern language of α_2 and comprises S_0. The pattern α_3, in turn, is even E-descriptive of S_0, but not NE-descriptive, since it is not consistent with S_0 if we disallow empty substitutions. Exactly the same holds for $\alpha_4 := x\mathtt{baba}y$, which also is consistent with S_0 if we allow the empty substitution of variables, generates an E-pattern language that is strictly included in the E-pattern language of α_2 and is E-descriptive, but not NE-descriptive of S_0.

The present paper examines the basic underlying problem of descriptive pattern discovery, namely the *existence* of such patterns; this means that we study the question of whether or not, for a given language S, there is a pattern that is descriptive of S. To this end, four different cases can be considered: NE-descriptive patterns of finite languages, NE-descriptive patterns of infinite languages, E-descriptive patterns of finite languages and E-descriptive patterns of infinite languages. The problem of the existence of the former three types of descriptive patterns is either trivial or has already been solved in previous publications. We therefore largely study the latter case, and our corresponding main result answers a question posed by Jiang, Kinber, Salomaa, Salomaa and Yu [5]. Our technical considerations do not only provide insights into the actual topic of our paper, but – due to the definition of descriptive patterns – also reveal vital phenomena related to the inclusion of E-pattern languages and, hence, the topology of class of terminal-free E-pattern languages. Due to the way the inclusion of terminal-free E-pattern languages is characterised, this implies that we have to deal with combinatorial properties of morphisms in free monoids.

2 Basic Definitions and Preparatory Technical Considerations

This paper is largely self-contained. For notations not explicitly defined, Rozenberg and Salomaa [10] can be consulted.

Let $\mathbb{N} := \{0, 1, 2, 3, \ldots\}$ and, for every $k \geq 0$, $\mathbb{N}_k := \{n \in \mathbb{N} \mid n \geq k\}$. The symbols $\subseteq$, $\subset$, $\supseteq$ and $\supset$ refer to subset, proper subset, superset and proper superset relation, respectively. The symbol ∞ stands for infinity. For an arbitrary alphabet A, a *string* (over A) is a finite sequence of symbols from A, and λ stands for the *empty string*. The symbol A^+ denotes the set of all nonempty strings over A, and $A^* := A^+ \cup \{\lambda\}$. For the *concatenation* of two strings w_1, w_2 we write $w_1 \cdot w_2$ or simply $w_1 w_2$. We say that a string $v \in A^*$ is a *factor* of a string $w \in A^*$ if there are $u_1, u_2 \in A^*$ such that $w = u_1 v u_2$. If $u_1 = \lambda$ (or $u_2 = \lambda$), then v is a *prefix* of w (or a *suffix*, respectively). The notation $|K|$ stands for the size of a set K or the length of a string K; the term $|w|_a$ refers to the number of occurrences of the symbol a in the string w. For any $w \in \Sigma^*$ and any $n \in \mathbb{N}$, w^n denotes the *n-fold concatenation of* w, with $w^0 := \lambda$.

For any alphabets A, B, a *morphism* is a function $h : A^* \to B^*$ that satisfies $h(vw) = h(v)h(w)$ for all $v, w \in A^*$. Given morphisms $g : A^* \to B^*$ and $h : B^* \to C^*$ (for alphabets A, B, C), their *composition* $(h \circ g)$ is defined by $(h \circ g)(w) := h(g(w))$ for all $w \in A^*$. For every morphism $h : A^* \to A^*$ and every $n \geq 0$, h^n denotes the *n-fold iteration of* h, i. e., $h^{n+1} := (h \circ h^n)$, where h^0 is the identity on A^*.

A morphism $h : A^* \to B^*$ is said to be *nonerasing* if $h(a) \neq \lambda$ for all $a \in A$. For any string $w \in C^*$, where $C \subseteq A$ and $|w|_a \geq 1$ for every $a \in C$, the morphism $h : A^* \to B^*$ is called a *renaming (of w)* if $h : C^* \to B^*$ is injective and $|h(a)| = 1$ for every $a \in C$.

Let Σ be a (finite or infinite) alphabet of so-called *terminal symbols* (or: *letters*) and X an infinite set of *variables* with $\Sigma \cap X = \emptyset$. We normally assume $\{\mathtt{a}, \mathtt{b}, \ldots\} \subseteq \Sigma$ and $\{y, z, x_0, x_1, x_2 \ldots\} \subseteq X$. A *pattern* is a string over $\Sigma \cup X$, a *terminal-free pattern* is a string over X and a *word* is a string over Σ. The set of all patterns over $\Sigma \cup X$ is denoted by Pat_Σ. For any pattern α, we refer to the set of variables in α as $\mathrm{var}(\alpha)$.

A morphism $\sigma : (\Sigma \cup X)^* \to (\Sigma \cup X)^*$ is called *terminal-preserving* if $\sigma(a) = a$ for every $a \in \Sigma$. A terminal-preserving morphism $\sigma : (\Sigma \cup X)^* \to \Sigma^*$ is called a *substitution.* Let $S \subseteq \Sigma^*$; then we say that a pattern α is *consistent with* S if, for every $w \in S$, there exists a substitution σ satisfying $\sigma(\alpha) = w$.

Intuitively, the pattern language of a pattern α is the maximum set of words α is consistent with. Formally, we consider two types of pattern languages, depending on whether we restrict ourselves to nonerasing substitutions: the *NE-pattern language* $L_{\mathrm{NE},\Sigma}(\alpha)$ of a pattern $\alpha \in \mathrm{Pat}_\Sigma$ is given by

$$L_{\mathrm{NE},\Sigma}(\alpha) := \{\sigma(\alpha) \mid \sigma : (\Sigma \cup X)^* \to \Sigma^* \text{ is a nonerasing substitution}\},$$

and the *E-pattern language* $L_{\mathrm{E},\Sigma}(\alpha)$ of α is given by

$$L_{\mathrm{E},\Sigma}(\alpha) := \{\sigma(\alpha) \mid \sigma : (\Sigma \cup X)^* \to \Sigma^* \text{ is a substitution}\}.$$

The term *pattern language* refers to any of the definitions introduced above. We call a pattern language *terminal-free* if it is generated by a terminal-free pattern.

We now can introduce our terminology on the main topic of this paper, namely the descriptiveness of a pattern. For any alphabet Σ and any language $S \subseteq \Sigma^*$, a pattern $\delta \in \mathrm{Pat}_\Sigma$ is said to be *NE-descriptive (of S)* provided that $L_{\mathrm{NE},\Sigma}(\delta) \supseteq S$ and, for every $\alpha \in \mathrm{Pat}_\Sigma$ with $L_{\mathrm{NE},\Sigma}(\alpha) \supseteq S$, $L_{\mathrm{NE},\Sigma}(\alpha) \not\subset L_{\mathrm{NE},\Sigma}(\delta)$. Analogously, δ is called *E-descriptive (of S)* if $L_{\mathrm{E},\Sigma}(\delta) \supseteq S$ and, for every $\alpha \in \mathrm{Pat}_\Sigma$ with $L_{\mathrm{E},\Sigma}(\alpha) \supseteq S$, $L_{\mathrm{E},\Sigma}(\alpha) \not\subset L_{\mathrm{E},\Sigma}(\delta)$.

Obviously, the definition of a descriptive pattern is based on the inclusion of pattern languages, which is an undecidable problem for both the full class of NE-pattern languages and the full class of E-pattern languages (cf. Jiang et al. [6], Freydenberger and Reidenbach [4]). A significant part of our subsequent technical considerations, however, can be restricted to terminal-free E-pattern languages, and here the inclusion problem is known to be decidable. This directly results from the following characterisation:

Theorem 1 (Jiang et al. [6]). *Let Σ be an alphabet, $|\Sigma| \geq 2$, and let $\alpha, \beta \in X^+$ be terminal-free patterns. Then $L_{\mathrm{E},\Sigma}(\alpha) \subseteq L_{\mathrm{E},\Sigma}(\beta)$ if and only if there exists a morphism $h : X^* \to X^*$ satisfying $h(\beta) = \alpha$.*

While Theorem 1 is a powerful tool when dealing with the inclusion of terminal-free E-pattern languages, the examination of the descriptiveness of a pattern requires insights into *proper* inclusion relations, and therefore we use some further combinatorial results on morphisms in free monoids to give a more convenient criterion that can replace the use of Theorem 1.

In accordance with Reidenbach and Schneider [9], we designate a terminal-free pattern $\alpha \in X^+$ as *morphically imprimitive* if there is a pattern $\beta \in X^*$ satisfying the following conditions: $|\beta| < |\alpha|$ and there are morphisms $g, h : X^* \to X^*$ such that $g(\alpha) = \beta$ and $h(\beta) = \alpha$. Otherwise, α is *morphically primitive.* Let $\alpha \in X^+$ be morphically primitive. A morphism $h : X^* \to X^*$ is said to be an *imprimitivity morphism (for α)* provided that $|h(\alpha)| > |\alpha|$ and there is a morphism $g : X^* \to X^*$ satisfying $g(h(\alpha)) = \alpha$. Referring to these concepts, we now can give a characterisation of certain proper inclusion relations between terminal-free E-pattern languages:

Lemma 1. *Let Σ be an alphabet, $|\Sigma| \geq 2$, $\alpha \in X^+$ a morphically primitive pattern and $h : X^* \to X^*$ a morphism. Then $L_{\mathrm{E},\Sigma}(h(\alpha)) \subset L_{\mathrm{E},\Sigma}(\alpha)$ if and only if h is neither an imprimitivity morphism for, nor a renaming of α.*

The proof for Lemma 1 is omitted due to space constraints.

The question of whether a given morphism is an imprimitivity morphism for a pattern can be easily answered using the following insight:

Theorem 2 (Reidenbach, Schneider [9]). *Let $\alpha \in X^+$ be a morphically primitive pattern. Then a morphism $h : X^* \to X^*$ is an imprimitivity morphism for α if and only if*

1. *for every $x \in \mathrm{var}(\alpha)$, there exists an $x_h \in \mathrm{var}(h(x))$ such that $|h(x)|_{x_h} = 1$ and $|h(y)|_{x_h} = 0$ for every $y \in \mathrm{var}(\alpha) \setminus \{x\}$, and*
2. *there exists an $x \in \mathrm{var}(\alpha)$ with $|h(x)| \geq 2$.*

Evidently, Lemma 1 can only be applied if there is a tool for checking whether a terminal-free pattern is morphically primitive. This is provided by the following characterisation:

Theorem 3 (Reidenbach, Schneider [9]). *A pattern $\alpha \in X^+$ is morphically primitive if and only if there is no factorisation $\alpha = \beta_0 \gamma_1 \beta_1 \gamma_2 \beta_2 \ldots \beta_{n-1} \gamma_n \beta_n$ with $n \geq 1$, $\beta_k \in X^*$ and $\gamma_k \in X^+$, $k \leq n$, such that*

1. *$|\gamma_k| \geq 2$ for every k, $1 \leq k \leq n$,*
2. *$\mathrm{var}(\beta_0 \ldots \beta_n) \cap \mathrm{var}(\gamma_1 \ldots \gamma_n) = \emptyset$ and*
3. *for every k, $1 \leq k \leq n$, there exists an $x_k \in \mathrm{var}(\gamma_k)$ such that $|\gamma_k|_{x_k} = 1$ and, for every k', $1 \leq k' \leq n$, if $x_k \in \mathrm{var}(\gamma_{k'})$ then $\gamma_k = \gamma_{k'}$.*

Thus, with Lemma 1, Theorem 2 and Theorem 3 we now have an appropriate tool for deciding on particular proper inclusion relations between terminal-free E-pattern languages.

3 Descriptive Patterns and Infinite Strictly Decreasing Chains of Pattern Languages

Before we state and prove the main results of our paper, we discuss some simple yet enlightening observations that establish a connection between descriptiveness of patterns and infinite strictly decreasing chains of pattern languages over some fixed alphabet, i. e. sequences $(L_i)_{i \in \mathbb{N}}$ of pattern languages satisfying, for every $j \in \mathbb{N}$, $L_j \supset L_{j+1}$. This aspect is already briefly mentioned by Jiang et al. [5].

Since, by definition, a descriptive pattern generates a smallest pattern language comprising a language S, S does not have a descriptive pattern if and only if no pattern language L satisfying $L \supseteq S$ is smallest. Hence, the existence of a descriptive pattern essentially depends on the existence of a pattern language that is not contained in an infinite strictly decreasing chain:

Observation 1. *Let Σ be an alphabet and $S \subseteq \Sigma^*$ a language. Then there is no pattern that is NE-descriptive (or E-descriptive) of S if and only if, for every pattern α with $L_{\mathrm{NE},\Sigma}(\alpha) \supseteq S$ (or $L_{\mathrm{E},\Sigma}(\alpha) \supseteq S$, respectively) there is*

- *a sequence of patterns $\alpha_i \in \mathrm{Pat}_\Sigma$, $i \in \mathbb{N}$, satisfying, for every $j \in \mathbb{N}$, $L_{\mathrm{NE},\Sigma}(\alpha_j) \supset L_{\mathrm{NE},\Sigma}(\alpha_{j+1})$ (or $L_{\mathrm{E},\Sigma}(\alpha_j) \supset L_{\mathrm{E},\Sigma}(\alpha_{j+1})$, respectively) and $L_{\mathrm{NE},\Sigma}(\alpha_j) \supseteq S$ (or $L_{\mathrm{E},\Sigma}(\alpha_j) \supseteq S$, respectively), and*
- *an $n \in \mathbb{N}$ with $L_{\mathrm{NE},\Sigma}(\alpha_n) = L_{\mathrm{NE},\Sigma}(\alpha)$ (or $L_{\mathrm{E},\Sigma}(\alpha_n) = L_{\mathrm{E},\Sigma}(\alpha)$, respectively).*

Proof. Directly from the definition of an NE-descriptive (or E-descriptive) pattern. □

Thus, the question of whether there is a descriptive pattern for a language S requires insights into the inclusion problem for pattern languages. As partly stated in Section 2, this problem is undecidable in the general case, but it is decidable for the class of terminal-free E-pattern languages (though combinatorially complex and, according to Ehrenfeucht and Rozenberg [3], NP-complete).

In order to illustrate and substantiate Observation 1 and as a reference for further considerations in Section 4, we now give some examples of strictly decreasing chains of pattern languages. We begin with a sequence of patterns that has identical properties for both NE- and E-pattern languages:

Example 1. Let Σ be any alphabet. For every $i \in \mathbb{N}$, we define $\alpha_i := x_1^{2^i}$, i. e. $\alpha_0 = x_1$, $\alpha_1 = x_1^2$, $\alpha_2 = x_1^4$, $\alpha_3 = x_1^8$ and so on. Evidently, for every $j \in \mathbb{N}$, the morphism $h : \{x_1\}^+ \to \{x_1\}^+$, defined by $h(x_1) := x_1^2$, satisfies $h(\alpha_j) = \alpha_{j+1}$. Since, for both NE- and E-pattern languages, the existence of such a morphism is a sufficient condition for an inclusion relation (cf. Theorems 2.2 and 2.3 by Jiang et al. [5]), $L_{\mathrm{NE},\Sigma}(\alpha_j) \supseteq L_{\mathrm{NE},\Sigma}(\alpha_{j+1})$ and $L_{\mathrm{E},\Sigma}(\alpha_j) \supseteq L_{\mathrm{E},\Sigma}(\alpha_{j+1})$ are satisfied. In the given example, it is evident that all inclusions of NE-pattern languages are strict. The same holds for the inclusion of E-pattern languages; alternatively, for all but unary alphabets Σ, it is directly proved by Lemma 1 (using Theorem 2 and Theorem 3) given in Section 2. Hence, the sequence of α_i leads to an infinite strictly decreasing chain for NE-pattern languages as well as for E-pattern languages. Nevertheless, the sequence of patterns is irrelevant in the context of Observation 1, as the sets $S_{\mathrm{NE}} := \bigcap_{i=0}^{\infty} L_{\mathrm{NE},\Sigma}(\alpha_i)$ and $S_{\mathrm{E}} := \bigcap_{i=0}^{\infty} L_{\mathrm{E},\Sigma}(\alpha_i)$, i. e. those languages all patterns are consistent with, are empty.

Our next example looks quite similar to Example 1, but here a difference between NE- and E-pattern languages can be noted:

Example 2. Let Σ be an alphabet with $|\Sigma| \geq 2$. For every $i \in \mathbb{N}$, we define $\alpha_i := x_1^{2^i} y^2$, i. e. $\alpha_0 = x_1 y^2$, $\alpha_1 = x_1^2 y^2$, $\alpha_2 = x_1^4 y^2$, $\alpha_3 = x_1^8 y^2$ and so on. Referring to the same facts as mentioned in Example 1, it can be shown that the patterns again define one infinite strictly decreasing chain of NE-pattern languages and another one of E-pattern languages. However, while the set $S_{\mathrm{NE}} := \bigcap_{i=0}^{\infty} L_{\mathrm{NE},\Sigma}(\alpha_i)$ again is empty, $S_{\mathrm{E}} := \bigcap_{i=0}^{\infty} L_{\mathrm{E},\Sigma}(\alpha_i)$ now equals $L_{\mathrm{E},\Sigma}(y^2)$. Hence, we have a chain of E-pattern languages that are all consistent with a nonempty language. Nevertheless, $L_{\mathrm{E},\Sigma}(y^2)$ obviously has a descriptive pattern, namely $\delta := y^2$, and this of course holds for all infinite sequences of patterns where S_{E} equals an E-pattern language. Consequently, the existence of a single infinite strictly decreasing chain of E-pattern languages L_i satisfying, for every $i \in \mathbb{N}$, $L_i \supseteq S$, does not mean that there is no E-descriptive pattern for S. Furthermore, it is worth mentioning that we can replace S_{E} with a finite language and still preserve the above described properties of the α_i and δ. For $\Sigma \supseteq \{\mathtt{a}, \mathtt{b}\}$, this is demonstrated, e. g., by the language $S := \{\mathtt{aa}, \mathtt{bb}\}$, which satisfies, for every $i \in \mathbb{N}$, $S \subseteq L_{\mathrm{E},\Sigma}(\alpha_i)$ and has the E-descriptive pattern δ.

Our final example presents a special phenomenon of E-pattern languages, namely the existence bi-infinite strictly decreasing/increasing chains of such languages:

Example 3. Let Σ be an alphabet with $|\Sigma| \geq 2$. For every $i \in \mathbb{Z}$, we define

$$\alpha_i := \begin{cases} x_1^{2^{-i}} & \text{if } i \text{ is negative,} \\ x_1^2 x_2^2 \ \ldots \ x_{i+2}^2 & \text{else.} \end{cases}$$

Hence, for example, from $i = -3$ to $i = 2$ the patterns read $\alpha_{-3} = x_1^8$, $\alpha_{-2} = x_1^4$, $\alpha_{-1} = x_1^2$, $\alpha_0 = x_1^2 x_2^2$, $\alpha_1 = x_1^2 x_2^2 x_3^2$, and $\alpha_2 = x_1^2 x_2^2 x_3^2 x_4^2$. Using Theorem 3, it is easy to show that all patterns are morphically primitive. Theorem 2 demonstrates that all morphisms mapping an α_k onto an α_j, $j < k$, are not imprimitivity morphisms. Therefore we can conclude from Lemma 1 that $L_{\mathrm{E},\Sigma}(\alpha_j) \subset L_{\mathrm{E},\Sigma}(\alpha_k)$ if and only if $j < k$. For the given patterns, $S_{\mathrm{E}} := \bigcap_{i=-\infty}^{\infty} L_{\mathrm{E},\Sigma}(\alpha_i)$ is empty, but if we define, for every $i \in \mathbb{Z}$, $\alpha_i' := y^2 \alpha_i$, then these α_i' generate a bi-infinite strictly decreasing/increasing chain of E-pattern languages where $S_{\mathrm{E}} := \bigcap_{i=-\infty}^{\infty} L_{\mathrm{E},\Sigma}(\alpha_i') = L_{\mathrm{E},\Sigma}(y^2)$ is an E-pattern language.

Note that the example patterns given above are terminal-free merely for the sake of convenience. They can be effortlessly turned into certain patterns containing terminal symbols and still showing equivalent properties.

4 The Existence of Descriptive Patterns

In the present chapter we study the existence of patterns that are descriptive of sets S of strings. According to our remarks in Section 1, four main cases can be considered, depending on whether S is finite or infinite and whether NE- or E-descriptiveness is examined. We focus on the existence of E-descriptive patterns for infinite languages since, for the other three cases, answers are absolutely straightforward or directly or indirectly provided by Angluin [1] and Jiang et al. [5]. In order to give a comprehensive description and further explain some of our formal concepts and statements we nevertheless also briefly describe the known or trivial cases.

Using Observation 1, the question of the existence of *NE-descriptive* patterns can be easily answered for all types of languages S. We begin with the case of a *finite* S. Here, it is primarily necessary to observe that a word w can only be covered by a pattern α through nonerasing substitutions if α is not longer than w. Hence, for any finite alphabet Σ and any word over Σ, there are only finitely many NE-pattern languages over Σ covering this word; this property of a class of languages is commonly referred to as *finite thickness* (cf. Wright [13]). Quite obviously, the same holds for infinite alphabets Σ, since the number of different terminal symbols that can occur in patterns covering w is limited by the number of different terminal symbols in w. With regard to infinite sequences of patterns (generating languages that all differ from each other) over a fixed alphabet, this means that none of them can contain infinitely many patterns that cover, e. g., the shortest word in a given finite set of strings. This immediately shows that, for every finite S, there exists an NE-descriptive pattern:

Proposition 1 (Angluin [1]). *Let Σ be an alphabet and $S \subseteq \Sigma^+$ a finite language. Then there is a pattern $\delta \in \mathrm{Pat}_\Sigma$ that is NE-descriptive of S.*

Note that Angluin [1] does not explicitly state Proposition 1, but directly studies more challenging questions by introducing a procedure computing an NE-descriptive pattern for any finite language S and examining the computational complexity of the problem of finding such patterns for finite languages.

With regard to NE-descriptive patterns for *infinite* languages S, the same reasoning as for finite languages S leads to the analogous result:

Proposition 2. *Let Σ be an alphabet and $S \subseteq \Sigma^+$ an infinite language. Then there is a pattern $\delta \in \mathrm{Pat}_\Sigma$ that is NE-descriptive of S.*

Proof. Directly from Observation 1 and the finite thickness of the class of NE-pattern languages. □

A closer look at the underlying reasoning proving Propositions 1 and 2 reveals that it does not need to consider whether any infinite sequence of patterns leads to an infinite strictly decreasing chain of NE-pattern languages (as featured by Observation 1), but can be completely based on the concept of finite thickness. If we nevertheless wish to examine the properties of such chains, then we can easily observe that, for every sequence of patterns α_i, $i \in \mathbb{N}$, with $L_{\mathrm{NE},\Sigma}(\alpha_i) \supset L_{\mathrm{NE},\Sigma}(\alpha_{i+1})$, the set $S_{\mathrm{NE}} := \bigcap_{i=0}^{\infty} L_{\mathrm{NE},\Sigma}(\alpha_i)$ necessarily is empty. Hence, Examples 1 and 2 illustrate the only option possible.

With regard to *E-descriptiveness*, the situation is more complex. As shown by Examples 2 and 3, the class of E-pattern languages does not have finite thickness and there are even finite and infinite languages that are contained in all E-pattern languages of an infinite strictly decreasing chain. Nevertheless, it is known that every *finite* language has an E-descriptive pattern:

Theorem 4 (Jiang et al. [5]). *Let Σ be an alphabet and $S \subseteq \Sigma^*$ a finite language. Then there is a pattern $\delta \in \mathrm{Pat}_\Sigma$ that is E-descriptive of S.*

The proof for Theorem 4 given by Jiang et al. [5] demonstrates that for every finite language S an upper bound n can be given such that, for every pattern α consistent with S, there exists a pattern β satisfying $|\beta| \leq n$ and $L_{\mathrm{E},\Sigma}(\beta) \subseteq L_{\mathrm{E},\Sigma}(\alpha)$. So if, for any finite S, there is a sequence of patterns α_i, $i \in \mathbb{N}$, leading to an infinite strictly decreasing chain of E-pattern languages comprising S – which implies that there is no upper bound for the length of the α_i – then all but finitely many of these patterns need to have variables that are not required for generating the words in S. This phenomenon is illustrated by Example 2, where only the subpattern y^2 of all patterns is necessary in order to map the patterns onto the words in S_{E}.

In the proof for Theorem 4, the upper bound n equals the sum of the lengths of the words in S. Thus, this method cannot be adopted when investigating the existence of E-descriptive patterns for *infinite* sets of words. In fact, as to be demonstrated below, we here need to consider two subcases depending on the number of different letters occurring in the words of S. If the underlying alphabet is unary, then the descriptiveness of a pattern is related to the inclusion relation of E-pattern languages over this unary alphabet. The structure of such E-pattern languages, however, is significantly simpler than that of E-pattern languages over larger alphabets; in particular, the full class of these languages is a specific subclass of the regular languages (namely the linear unary languages). Therefore, and just as in the previous cases, it can be shown that, for every sequence of patterns $(\alpha_i)_{i\in\mathbb{N}}$ leading to a infinite strictly decreasing chain of E-pattern languages over a unary alphabet, the language $S_{\mathrm{E}} := \bigcap_{i=0}^{\infty} L_{\mathrm{E},\Sigma}(\alpha_i)$ is empty. Referring to Observation 1, this directly leads to the following result:

Theorem 5. *Let Σ be an alphabet, $|\Sigma| = 1$, and $S \subseteq \Sigma^*$ an infinite language. Then there is a pattern $\delta \in \mathrm{Pat}_\Sigma$ that is E-descriptive of S.*

The proof for Theorem 5 is omitted due to space constraints.

In contrast to this, Example 2 demonstrates that, for alphabets with at least two letters, there is an infinite strictly decreasing chain of E-pattern languages such that the intersection of all these languages is nonempty. Since this intersection is an E-pattern language, Example 2 can nevertheless not be used to establish a result that differs from those given for the other cases. In order to answer the question of whether this holds true for all such chains, we now consider a more sophisticated infinite sequence of patterns, that is defined as follows:

Definition 1. *We define the pattern $\alpha_0 := y^2z^2$ and the morphism $\phi : X^* \to X^*$ (note that we assume $X \supseteq \{y, z, x_0, x_1, x_2 \ldots\}$) through, for every $i \in \mathbb{N}$,*

$$\phi(x_i) := x_{i+1},\ \phi(y) := y^2x_1,\ \phi(z) := x_1z^2.$$

Then, for every $i \in \mathbb{N}$, the pattern α_{i+1} is given by $\alpha_{i+1} := \phi(\alpha_i) = \phi^i(\alpha_0)$.

This means that, for example,

$$\begin{aligned}
\alpha_1 &= y^2x_1\, y^2x_1\, x_1z^2\, x_1z^2,\\
\alpha_2 &= (y^2x_1y^2x_1x_2)\, (y^2x_1y^2x_1x_2)\, (x_2x_1\, z^2x_1z^2)\, (x_2x_1z^2x_1z^2),\\
\alpha_3 &= (y^2x_1y^2x_1x_2\, y^2x_1y^2x_1x_2\, x_3)\, (y^2x_1y^2x_1x_2\, y^2x_1y^2x_1x_2\, x_3)\\
&\quad (x_3\, x_2x_1z^2x_1z^2\, x_2x_1z^2x_1z^2)\, (x_3\, x_2x_1z^2x_1z^2\, x_2x_1z^2x_1z^2).
\end{aligned}$$

It can be shown that this sequence $(\alpha_i)_{i\in\mathbb{N}}$ defines an infinite strictly decreasing chain of E-pattern languages. Furthermore, if we define the morphism $\psi : X^* \to$

X^* through $\psi(x_i) := x_i$ and $\psi(y) := \psi(z) := x_0$, then, for every alphabet Σ with $|\Sigma| \geq 2$, $L_\Sigma := \bigcup_{i=0}^{\infty} L_{\mathrm{E},\Sigma}(\psi(\alpha_i))$ satisfies $L_\Sigma \subseteq \bigcap_{i=0}^{\infty} L_{\mathrm{E},\Sigma}(\alpha_i)$. Finally, it can be demonstrated that the sequence $(\alpha_i)_{i\in\mathbb{N}}$ has a very particular property, since for every pattern γ with $L_{\mathrm{E},\Sigma}(\gamma) \supseteq L_\Sigma$ there exists an α_i satisfying $L_{\mathrm{E},\Sigma}(\gamma) \supseteq L_{\mathrm{E},\Sigma}(\alpha_i)$. Referring to Observation 1, this implies the main result of our paper:

Theorem 6. *For every alphabet Σ with $|\Sigma| \geq 2$ there is an infinite language $L_\Sigma \subset \Sigma^*$ that has no E-descriptive pattern.*

The proof for Theorem 6 is omitted due to space constraints.

Consequently, when searching for descriptive patterns, the case of E-descriptive patterns of infinite languages over alphabets of at least two letters is the only one where the existence of such patterns is not always guaranteed. This directly answers a question posed by Jiang et al. [5].

Finally, it can be shown that, while the proof of Theorem 6 is based on the particular shape of the infinite union L_Σ of E-pattern languages described above, L_Σ can be replaced by a language L^t_Σ which, for every pattern $\psi(\alpha_i)$, $i \geq 0$, contains just a single word. In order to describe this insight more precisely, we have to introduce the following concept:

Definition 2. *A language L is called* properly thin *if, for every $n \geq 0$, L contains at most one word of length n.*

Referring to this definition, we can strengthen Theorem 6 as follows:

Corollary 1. *For every alphabet Σ with $|\Sigma| \geq 2$, there is an infinite properly thin language $L^t_\Sigma \subset \Sigma^*$ that has no E-descriptive pattern.*

The proof for Corollary 1 is omitted due to space constraints.

5 Conclusions and Further Directions of Research

In the present paper, we have studied the existence and nonexistence of patterns that are descriptive of a set of strings. We have explained that this question is related to the existence of infinite strictly decreasing chains of pattern languages. Our main result has demonstrated that there exist infinite languages over alphabets of at least two letters that do not have an E-descriptive pattern.

This insight leads to the question of characteristic criteria describing infinite languages without an E-descriptive pattern. We have referred to one example of such languages, namely a particular infinite union of E-pattern languages. Although we have mentioned that an infinite properly thin language can be substituted for this union, we anticipate that only very special languages (and very special infinite strictly decreasing chains of E-pattern languages) can be used for the proof of our main result. Thus, we expect the nonexistence of E-descriptive patterns to be a rare phenomenon. In addition to the said criteria, we consider it worthwhile to further investigate the existence of efficient procedures finding descriptive patterns of given languages (for those cases where descriptive patterns exist). So far, this question has only been answered for NE-descriptive patterns of finite languages (see Angluin [1]), demonstrating that no such procedure can have polynomial runtime (provided that P≠NP). We feel that a more pleasant result might be possible for E-descriptive patterns.

References

1. Angluin, D.: Finding patterns common to a set of strings. Journal of Computer and System Sciences 21, 46–62 (1980)
2. Brazma, A., Jonassen, I., Eidhammer, I., Gilbert, D.: Approaches to the automatic discovery of patterns in biosequences. Journal of Computational Biology 5, 279–305 (1998)
3. Ehrenfeucht, A., Rozenberg, G.: Finding a homomorphism between two words is NP-complete. Information Processing Letters 9, 86–88 (1979)
4. Freydenberger, D.D., Reidenbach, D.: Bad news on decision problems for patterns. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257. Springer, Heidelberg (2008)
5. Jiang, T., Kinber, E., Salomaa, A., Salomaa, K., Yu, S.: Pattern languages with and without erasing. International Journal of Computer Mathematics 50, 147–163 (1994)
6. Jiang, T., Salomaa, A., Salomaa, K., Yu, S.: Decision problems for patterns. Journal of Computer and System Sciences 50, 53–63 (1995)
7. Mateescu, A., Salomaa, A.: Patterns. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, 4.6, vol. 1, pp. 230–242. Springer, Heidelberg (1997)
8. Ng, Y.K., Shinohara, T.: Developments from enquiries into the learnability of the pattern languages from positive data. Theoretical Computer Science 397, 150–165 (2008)
9. Reidenbach, D., Schneider, J.C.: Morphically primitive words. Theoretical Computer Science 410, 2148–2161 (2009)
10. Rozenberg, G., Salomaa, A.: Handbook of Formal Languages, vol. 1. Springer, Berlin (1997)
11. Salomaa, K.: Patterns. In: Martin-Vide, C., Mitrana, V., Păun, G. (eds.) Formal Languages and Applications. Studies in Fuzziness and Soft Computing, vol. 148, pp. 367–379. Springer, Heidelberg (2004)
12. Shinohara, T.: Polynomial time inference of extended regular pattern languages. In: Goto, E., Nakajima, R., Yonezawa, A., Nakata, I., Furukawa, K. (eds.) RIMS 1982. LNCS, vol. 147, pp. 115–127. Springer, Heidelberg (1983)
13. Wright, K.: Identification of unions of languages drawn from an identifiable class. In: Proc. 2nd Annual Workshop on Computational Learning Theory, COLT 1989, pp. 328–333 (1989)

On Stateless Multihead Finite Automata and Multihead Pushdown Automata

Pierluigi Frisco[1] and Oscar H. Ibarra[2,*]

[1] School of Mathematical and Computer Sciences, Heriot-Watt University,
EH14 4AS Edinburgh, UK
pier@macs.hw.ac.uk

[2] Department of Computer Science, University of California, Santa Barbara,
CA 93106, USA
ibarra@cs.ucsb.edu

Abstract. A stateless k-head two-way deterministic finite automaton (k-head 2DFA), has only one state, hence the designation stateless. Its transitions depends solely on the symbols currently scanned by its k heads, and in every such transition each head can move one cell left, right, or remain stationary. An input, which is delimited by end markers, is accepted if the machine, when started with all heads on the left end marker, reaches the configuration where all the heads are on the right end marker. The nondeterministic version is denoted by k-head 2NFA.

We prove that stateless $(k+1)$-head 2DFAs (resp., 2NFAs) are computationally more powerful than k-head 2DFAs (resp., 2NFAs), improving a recent result where it was shown that $(k+4)$ heads are better than k heads.

We also study stateless multihead pushdown automata in their two-way and one-way, deterministic and nondeterministic variations and show that for all these varieties, $k+1$ heads allow more computational power than k heads. Finally, we give some characterizations of stateless multihead finite and multihead pushdown automata.

1 Introduction

Stateless multihead finite automata are formal systems with no states (hence can be regarded as just having one state) equipped with a tape and with several reading heads. The operation of these systems only depends on the symbols scanned by the heads. These automata were first investigated in [10] because of their connection to certain aspects of membrane computing and P systems, a subarea of molecular computing that was introduced in [6] (see also [7]). A membrane in a P system consists of a multiset of objects drawn from a given finite set of objects. The system has no global state (i.e., stateless) and works on the evolution of objects.

[*] The work of Oscar H. Ibarra was supported in part by NSF Grant CCF-0524136 and a Distinguished Visiting Fellowship at the University of Liverpool from the UK Royal Academy of Engineering.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 240–251, 2009.

The stateless multihead automata in [10] were mainly studied with respect to decision problems (e.g., emptiness). Later, hierarchies with respect to the number of heads were established in [4], where it was shown that stateless $(k+1)$-head 1DFAs (resp., 1NFAs) are computationally more powerful than stateless k-head 1DFAs (resp., 1NFAs), and stateless $(k+4)$-head 2DFAs (resp., 2NFAs) are computationally more powerful than stateless k-head 2DFAs (resp., 2NFAs). It was left open in [4], whether the "k+4" in the latter result can be improved. In this paper, we show that, indeed, stateless $(k+1)$-head 2DFAs (resp., 2NFAs) are computationally more powerful than stateless k-head 2DFAs (resp., 2NFAs).

We also look at stateless multihead pushdown automata in their two-way deterministic, two-way nondeterministic, one-way deterministic, and one-way nondeterministic versions (2DPDA, 2NPDA, 1DPDA, 1NPDA, respectively). The operation of these machines are similar to multihead automata, except that now, the transitions also depend on the stack.

Finally, we give some characterizations of stateless multihead finite and multihead pushdown automata.

Our proofs employ translational techniques like the ones in [3,5,8] but the translations we use are much more complicated because of the absence of states in the simulating systems.

2 Basic Definitions

Denote a two-way nondeterministic (deterministic) finite automaton by 2NFA (2DFA). Similarly denote the one-way variant by 1NFA (1DFA). We consider *stateless k-head 2NFAs* and define them as pairs (Σ, δ) where:

Σ is an *alphabet* with $\triangleright, \triangleleft \notin \Sigma$, $\triangleright$ is the left end marker and $\triangleleft$ is the right end marker;

δ is a finite *set of transitions* of the form $(a_1, \ldots, a_k) \rightarrow (d_1, \ldots, d_k)$, where $a_i \in \Sigma \cup \{\triangleright, \triangleleft\}$, $1 \leq i \leq k$, is the symbol read by the i-th head, while $d_i \in \{\mathsf{l}, \mathsf{s}, \mathsf{r}\}$ tells where the i-th head is to be moved (l, s and r, stand for *left*, *stay* and *right*, respectively).

If there is at most one transition for every combination of symbols $a_1, \ldots,$ $a_k \in \Sigma \cup \{\triangleright, \triangleleft\}$, we refer to such an automaton as a *stateless k-head 2DFA*. If none of the transitions move any heads to the left, such an automaton is called a *stateless k-head 1NFA (1DFA)*.

For an input string $w \in \Sigma^*$, the machine works on a tape containing $\triangleright w \triangleleft$ and start with all heads reading the left end marker $\triangleright$. At every step of the computation, the symbols $a_1, \ldots, a_k$ currently scanned by all k heads are considered, any corresponding transition $a_1 \ldots a_k \rightarrow d_1 \ldots d_k \in \delta$ is chosen, and each i-th heads is moved according to d_i. The string is accepted if there exists a computation resulting in all k heads reading the end marker $\triangleleft$. As an example, the stateless 2-head 1DFA with transitions $(\triangleright, \triangleright) \rightarrow (\mathsf{s}, \mathsf{r})$, $(\triangleright, a) \rightarrow (\mathsf{s}, \mathsf{r})$, $(\triangleright, b) \rightarrow (\mathsf{r}, \mathsf{r})$, $(a, b) \rightarrow (\mathsf{r}, \mathsf{r})$, and $(b, \triangleleft) \rightarrow (\mathsf{r}, \mathsf{s})$ recognizes the language $L = \{a^n b^{n+1} \mid n \geqslant 0\}$.

Stateless multihead pushdown automata can be defined as stateless k-head 2NFA but with the set δ having elements of the form $(a_1, \ldots, a_k, Z) \rightarrow (d_1, \ldots d_k,$

γ), where Z is the symbol at the top of the pushdown stack, and $\gamma \in \Gamma^*$, with Γ the pushdown alphabet. (In this transition, Z is replaced by the string γ, with the rightmost symbol of γ becoming the new top of the stack. If γ is the null string, Z is popped from the stack.) We show that for all varieties of stateless multihead pushdown automata, $k+1$ heads are computationally more powerful than k heads.

In this paper, when we say that a machine M can be simulated by another machine M', we mean that there is an encoding of inputs for M to inputs for M' such that an input w is accepted by M if and only if the encoding w' of w is accepted by M'.

3 Stateless Multihead Two-Way Finite Automata

Our proofs involve "translations" (or reductions) to multihead automata with states. A *k-head 2NFA (2DFA) with states*, is a tuple $(\Sigma, \delta, S, s_0, s_f)$ where:

Σ is an *alphabet* with $\triangleright, \triangleleft \notin \Sigma$;
δ a finite *set of transitions* of the form $(s, a_1, \ldots, a_k) \rightarrow (s', d_1, \ldots, d_k)$ with $s, s' \in S$, being the *current* and *next* state, respectively, $a_1, \ldots, a_k \in \Sigma \cup \{\triangleright, \triangleleft\}$ being the symbols currently read by the k heads, and $d_1, \ldots, d_k \in \{\mathsf{l}, \mathsf{s}, \mathsf{r}\}^k$ being the movements of the k heads;
S is a finite *set of states*;
$s_0 \in S$ is the *initial state*;
$s_f \in S$ is the *final state*.

Such an automaton starts on a tape containing $\triangleright w \triangleleft$ in initial state s_0 with all heads reading $\triangleright$. If the automaton is in state s, then a transition of the form $(s, a_1, \ldots, a_k) \rightarrow (s', d_1, \ldots, d_k)$ can be applied only if head i reads a_i, $1 \leq i \leq k$. As a consequence of the application of such a transition, the state of the automaton changes into s' and head i moves according to d_i. The automaton accepts when it is in state s_f with all heads reading $\triangleleft$.

Definition 1. *Let $M_1 = (\Sigma, \delta, S, s_0, s_f)$ be a k-head $2DFA$ with states such that its transitions are of the form $(x, a_1, \ldots, a_k) \rightarrow (y, d_1, \ldots, d_k)$ where $d_1, d_2 \in \{\mathsf{l}, \mathsf{r}\}$ and $d_3, \ldots, d_k \in \{\mathsf{l}, \mathsf{r}, \mathsf{s}\}$.*

Given a string $w = w_1 \ldots w_{|w|}$ over Σ we define the string $w' = p_1 p_2 w$ over

$$\Sigma' = \Sigma \cup \{q^{d_1,d_2}_{x,y}, q'^{d_1,d_2}_{x,y}, \bar{q}^{d_1,d_2}_{x,y}, \bar{q}'^{d_1,d_2}_{x,y} \mid (x, a_1, \ldots, a_k) \rightarrow (y, d_1, \ldots, d_k) \in \delta, \; d_1, d_2 \in \{\mathsf{l}, \mathsf{r}\}\}$$

with:

$p_1 = w_1 t w_2 t \ldots w_{|w|} t$ where t is the lexicographic ordering of $tr_{x,y}$ for $x, y \in S$, $(x, a_1, \ldots, a_k) \rightarrow (y, d_1, \ldots, d_k) \in \delta$, and
$tr_{x,y} = \bar{q}^{\mathsf{l},\mathsf{l}}_{x,y}\; q^{\mathsf{l},\mathsf{l}}_{x,y}\; \bar{q}^{\mathsf{r},\mathsf{l}}_{x,y}\; q^{\mathsf{r},\mathsf{l}}_{x,y}\; q_{x,y}\; q^{\mathsf{l},\mathsf{r}}_{x,y}\; \bar{q}^{\mathsf{l},\mathsf{r}}_{x,y}\; q^{\mathsf{r},\mathsf{r}}_{x,y}\; \bar{q}^{\mathsf{r},\mathsf{r}}_{x,y}$;

$p_2 = w_1 t' w_2 t' \ldots w_{|w|} t'$ where t' is the lexicographic ordering of $tr'_{x,y}$ for $x, y \in S$, $(x, a_1, \ldots, a_k) \rightarrow (y, d_1, \ldots, d_k) \in \delta$, and
$tr'_{x,y} = \bar{q}'^{\mathsf{r},\mathsf{l}}_{x,y}\; q'^{\mathsf{r},\mathsf{l}}_{x,y}\; \bar{q}'^{\mathsf{l},\mathsf{l}}_{x,y}\; q'^{\mathsf{l},\mathsf{l}}_{x,y}\; q'_{x,y}\; q'^{\mathsf{l},\mathsf{r}}_{x,y}\; \bar{q}'^{\mathsf{l},\mathsf{r}}_{x,y}\; q'^{\mathsf{r},\mathsf{r}}_{x,y}\; \bar{q}'^{\mathsf{r},\mathsf{r}}_{x,y}$.

With lexicographic order *we mean that, for instance, $tr_{x,y}$ comes before $tr_{x',y'}$ if and only if $x < x'$ or $x = x'$ and $y \leq y$.*

For a better understanding of the previous definition we add a very simple example. Let $M = (\Sigma, \delta, S, s_0, s_2)$ be a k-head $2DFA$ with $\Sigma = \{a, b\}$, $\delta = \{(s_0, \triangleright) \rightarrow (s_0, \mathsf{r}), (s_1, a) \rightarrow (s_1, \mathsf{r}), (s_1, b) \rightarrow (s_2, \mathsf{r})\}$ and $S = \{s_0, s_1, s_2\}$.

Assuming that $s_0 < s_1 < s_2$, then by Definition 1, we have $t = tr_{s_0,s_1} tr_{s_1,s_1} tr_{s_1,s_2}$ and $t' = tr'_{s_0,s_1} tr'_{s_1,s_1} tr'_{s_1,s_2}$. If we then consider $w = aab$, then $p_1 = atatbt$ and $p_2 = at'at'bt'$.

Lemma 1. *For $k \geq 2$, any k-head* 2DFA *with states having the string w as input can be simulated by a stateless k-head* 2DFA *having the string w' as input, with w' as in Definition 1.*

Proof. (Sketch) Let $M_1 = (\Sigma, \delta, S, s_0, s_f)$ be a k-head $2DFA$ with states such that $S = \{1, \ldots, |S|\}$, $s_0 = 1$ and $s_f = |S|$. In order to have a simpler proof we assume that the transitions of M_1 are of the form $(x, a_1, \ldots, a_k) \rightarrow (y, d_1, \ldots, d_k)$ where $d_1, d_2 \in \{\mathsf{l}, \mathsf{r}\}$ and $d_3, \ldots, d_k \in \{\mathsf{l}, \mathsf{r}, \mathsf{s}\}$. This assumption is not restrictive as given M_1, a $2DFA$ such that $d_1, d_2 \in \{\mathsf{l}, \mathsf{r}, \mathsf{s}\}$, then it is always possible to construct another $2DFA$ M_1' accepting the same language of M_1 and such that $d_1, d_2 \in \{\mathsf{l}, \mathsf{r}\}$. For each transition of M_1 for which head 1 or head 2 does not move, M_1' has two transitions: the corresponding head moves right and then back to the left if the head was not on $\triangleleft$; otherwise, it moves left and then right.

We define $M_2 = (\Sigma', \delta')$, a stateless k-head *2DFA* with

$$\Sigma' = \Sigma \cup \{q^{d_1,d_2}_{x,y}, {q'}^{d_1,d_2}_{x,y}, \bar{q}^{d_1,d_2}_{x,y}, \bar{q}'^{d_1,d_2}_{x,y} \mid (x, a_1, \ldots, a_k) \rightarrow (y, d_1, \ldots, d_k) \in \delta, \\ d_1, d_2 \in \{\mathsf{l}, \mathsf{r}\}\}.$$

Given a string $w = w_1 \ldots w_{|w|}$ over Σ we define the string $w' = p_1 p_2 w$ as in Definition 1.

During a computation of M_2 head 1 mainly reads p_1, head 2 mainly reads p_2, while the remaining heads mainly read w. The computations of M_2 can be logically divided in three parts: *initialization*, *simulation* and *termination*.

Initialization. In the initial configuration all the heads of M_2 read $\triangleright$ in w'. During initialization the heads move such that at the end of it:

head 1 reads $q_{s_0,y}$ in the $tr_{s_0,y}$ coming after w_1 in p_1;
head 2 reads $q'_{s_0,y}$ in the $tr'_{s_0,y}$ coming after w_1 in p_2;
the remaining heads read a_1 in w.

Simulation. The simulation of transitions of M_1 starts and ends with head 1 and head 2 reading symbols of the kind $q_{x,y}$ and $q'_{x,y}$, respectively, for $x, y \in S$. This process is rather complex but it simply consists of the following. Let us assume that $p_1 = w_1 t_1 w_2 t_2 w_3 t_3 w_4$ (we know that $t_1 = t_2 = t_3 = t$ but here we need to refer to a specific t so we use subscripts to differentiate them) and that head 1 reads $q_{x,y}$ in t_2 for $(x, a_1, \ldots, a_k) \rightarrow (y, d_1, \ldots, d_k) \in \delta$. If $d_1 = \mathsf{r}$, then head 1 moves to the right until it reads $q_{y,z}$ in t_3, if instead $d_1 = \mathsf{l}$, then head 1 moves to the left until it reads $q_{y,z}$ in t_1. Head 2 moves in p_2 in a similar fashion depending on d_2.

It should be clear that this process requires more than one transition in M_2. The strings $tr_{x,y}$ and $tr'_{x,y}$ have substrings of pairs of symbols: one with a bar and one without. As we will see, the barred symbols in p_1 (p_2) are used to indicate that head 2 (head 1) passed to another t while scanning p_2 (p_1). The superscripts (pairs or elements in $\{\mathsf{l}, \mathsf{r}\}$) indicate in which direction head 1 and head 2 have to move.

Phase 1. The transition is simulated on the heads from 3 until k, while head 1 moves to read $q^{d_1,d_2}_{x,y}$. Head 2 co-operates with head 1.

Phase 2. Let us assume that head 1 reads $q^{d_1,d_2}_{x,y}$. When head 2 reads a symbol in w, head 1 moves to read $\bar{q}^{d_1,d_2}_{x,y}$, then head 2 moves in direction d_2 until it reads ${q'}^{d_1,g}_{y,z}$ (where $g = \mathsf{l}$ if $d_2 = \mathsf{r}$ and $g = \mathsf{r}$ if $d_2 = \mathsf{l}$).

Phase 3. When head 2 reads ${q'}^{d_1,g}_{y,z}$ the transitions in the present phase let head 1 to move in direction d_1 until it reads a symbol in Σ. Because of the lexicographic order during phase 3 the transitions in phase 2 cannot be applied.

Phase 4. Similarly to what performed by the transitions in phase 2, when head 1 reads a symbol in Σ and head 2 reads ${q'}^{d_1,g}_{y,z}$, the transitions in the present phase let head 2 to move to read $\bar{q'}^{d_1,g}_{y,z}$. Head 1 goes on moving in direction d_1 until it reads $q_{y,z}$.

Phase 5. When head 1 reads $q_{y,z}$, then head 2 moves to read $q'_{y,z}$.

Termination. When head 1 reads q_{x,s_f} and head 2 reads q'_{x,s_f}, then all the heads move to the right until they read $\triangleleft$. □

We were not able to prove a result similar to Lemma 1 for $2NFA$ due to the following difficulty. In the proof of Lemma 1 the transitions in phase 2 let head 2 to 'search' for (the unique) ${q'}^{d_1,g}_{y,z}$ while head 1 reads either $q^{d_1,d_2}_{x,y}$ or $\bar{q}^{d_1,d_2}_{x,y}$. When this happens the transitions in phase 3 let head 1 to move while head 2 does not. If a $2NFA$ tries to do a similar thing, then, as there can be several $q^{d_1,d_2}_{x,y}$ and $\bar{q}^{d_1,d_2}_{x,y}$ for different y in p_1, then when head 2 reads ${q'}^{d_1,g}_{y,z}$ (for a certain y) and transitions in phase 3 are applied, then it can be that head 1 reads $q^{d_1,d_2}_{x,y'}$. In this case the transitions in phase 2 can be applied again leading not to a simulation of any transition in M_1.

We did not succeed either to avoid such behaviour or to let the computation of M_1 to halt in a non accepting configuration when such behaviour occurs.

If the number of heads of M_1 is at least 3, then a stateless $2NFA$ can simulate a $2NFA$ with states.

Definition 2. *Let $M_1 = (\Sigma, \delta, S, s_0, s_f)$ be a k-head $2DFA$ with states such that its transitions are of the form $(x, a_1, \ldots, a_k) \rightarrow (y, d_1, \ldots, d_k)$ where $d_1, d_2 \in \{\mathsf{l}, \mathsf{r}\}$ and $d_3, \ldots, d_k \in \{\mathsf{l}, \mathsf{r}, \mathsf{s}\}$.*

Given a string $w = w_1 \ldots w_{|w|}$ over Σ we define the string $w' = p_1p_2p_3w$ over

$$\Sigma' = \{q^{d_1,d_2}_{x,y}, q'^{d_1,d_2}_{x,y}, \bar{q}^{d_1,d_2}_{x,y}, \bar{q}'^{d_1,d_2}_{x,y} \mid (x, a_1, \ldots, a_k) \to (y, d_1, \ldots, d_k) \in \delta, \\ d_1, d_2 \in \{\mathsf{l}, \mathsf{s}, \mathsf{r}\}\} \cup \Sigma \cup \{\bar{\sigma} \mid \sigma \in \Sigma\}$$

with:

p_1 *and* p_2 *as in Definition 1;*
$p_3 = w_1\bar{w}_1 \ldots w_{|w|}\bar{w}_{|w|}$.

Lemma 2. *For $k \geq 3$, any k-head* 2NFA *with states having the string w as input can be simulated by a stateless k-head* 2NFA *having the string w' as input, with w' as in Definition 2.*

We can now state the main results of this section:

Theorem 1. *For $k \geq 2$, stateless k-head* 2DFAs *are computationally more powerful than stateless $(k-1)$-head* 2DFAs*.*

Proof. The case $k = 2$ is obvious, since the language $\{a^nb^{n+1} \mid n \geq 0\}$ can be accepted by a stateless 2-head DFA but not by a stateless 1-head DFA, since 1-head DFAs (even with states) accept only regular sets.

Now let $k \geq 3$. Let L_1 be a language accepted a k-head 2DFA with states M_1 but not by any $(k-1)$-head 2DFA with states. Such a language exists [5]. Let M_2 be the stateless k-head 2DFA simulating M_1 as described in Lemma 1. Let $L_2 = L(M_2)$. We claim that L_2 cannot be accepted by a stateless $(k-1)$-head 2DFA. Suppose not, i.e., L_2 is accepted by a stateless $(k-1)$-head 2DFA M_3. We can then construct from M_3 a $(k-1)$-head 2DFA M_4 with states to accept the original language $L_1 = L(M_1)$ as follows.

When given $\triangleright w \triangleleft$, M_4 simulates the computation of M_3 on $\triangleright p_1p_2w \triangleleft$. But since M_4 is only given $\triangleright w \triangleleft$ it will use the finite-state control to keep track of the movements of each head when each head "moves" into the segment p_1p_2. Clearly, since p_1 and p_2 are fixed patterns, this can be done.

We get a contradiction, since L_1 cannot be accepted by a $(k-1)$-head 2DFA with states. □

Similarly, since k-head 2NFAs with states are computationally more powerful than $(k-1)$-head 2NFAS (for $k \geq 2$) [5], we have the following result using Lemma 2. However, the case $k = 3$ is still open.

Theorem 2. *Stateless k-head* 2NFAs *are computationally more powerful than stateless $(k-1)$-head* 2NFAs *for $k = 2$ and $k \geq 4$.*

We note that in the proof of Lemma 1 $|\Sigma'| = |\Sigma| + 16|\delta|$, where $|\delta|$ is the number of transitions in δ, while $|w'| = 18|\delta||w| + |w|$ (as $|p_1| = |p_2| = 9|\delta||w|$). If extra heads are added, then the size of the alphabet Σ' and the length of string w' used by a stateless 2DFA to simulate a k-head 2DFA with states can substantially decrease.

Lemma 3. *Let $M_1 = (\Sigma, \delta, S, s_0, s_f)$ be a k-head* 2DFA *with states where $k \geq 1$. There is a stateless $(k+1)$-head 2DFA $M_2 = (\Sigma', \delta')$ simulating M_1 with $|\Sigma'| = |\Sigma| + 4|\delta|$. The machine M_2 can simulate M_1 with input $|w|$ using an input string of length $|w'| = 3|\delta| + |w||\delta| + |w|$.*

Lemma 4. *Let* $M_1 = (\Sigma, \delta, S, s_0, s_f)$ *be a* k*-head* 2DFA *with states where* $k \geq 1$. *There is a stateless* $(k+2)$*-head 2DFA* $M_2 = (\Sigma', \delta')$ *simulating* M_1 *with* $|\Sigma'| = |\Sigma| + 4|\delta|$. *The system* M_2 *can simulate* M_1 *with input* $|w|$ *using an input string of length* $|w'| = 4|\delta| + |w|$.

Finally, when the number of heads of the simulating stateless 2DFA is $k+3$, we have [4, Lemma 1]:

Lemma 5. *Let* $M_1 = (\Sigma, \delta, S, s_0, s_f)$ *be a* k*-head* 2DFA *with states where* $k \geq 1$. *There is a stateless* $(k+3)$*-head 2DFA* $M_2 = (\Sigma', \delta')$ *simulating* M_1 *with* $|\Sigma'| = |\Sigma| + |\delta|$. *The system* M_2 *can simulate* M_1 *with input* $|w|$ *using an input string of length* $|w'| = |\delta| + |w|$.

We do not know if the cardinality of Σ' and the length of w' in Lemmas 1, 3, 4 and 5 are optimal. Here we simply wanted to point out how the cardinality and length decrease when we increase the number of heads.

4 Stateless Multihead Pushdown Automata

Let M be a k-head 2DPDA (two-way deterministic pushdown automaton) over input alphabet Σ. An input to M is a tape of the form $\triangleright w \triangleleft$, where $w \in \Sigma^*$ with $\triangleright$ and $\triangleleft$ the left and right end markers. Let $\triangleright w \triangleleft = a_1 \cdots a_n$. (Thus, $n \geq 2$, and $a_1 = \triangleright$ and $a_n = \triangleleft$.) Let $\{1, \cdots, m\}$ be the sets of states. Initially, the stack contains B (the bottom of the stack) and all k heads are on a_1 (i.e., on $\triangleright$). The transitions of M are of the form $(i, x_1, \cdots, x_k, z) \to (j, d_1, \cdots, d_k, \gamma)$. The transition means that when M is in state i, the heads $H_1, \cdots, H_k$ read $x_1, \cdots, x_k \in \Sigma \cup \{\triangleright, \triangleleft\}$, and the *top of the stack* (TOS) is z, then M changes state to j, moves head H_s in direction $d_s \in \{\mathsf{l}, \mathsf{s}, \mathsf{r}\}$, and replaces z with the string γ. The tape $w = a_1 \cdots a_n$ (actually, $a_2 \cdots a_{n-1}$, without the end markers) is accepted if and only if M eventually reaches the configuration where all the heads are on a_n (i. e., on $\triangleleft$) and the stack is empty (there is no move on empty stack). We assume that no head falls off the input tape.

The construction (translation) for multihead 2DFAs in the previous section does not carry over to multihead 2DPDAs, since the contents of the stack cannot be handled in that construction. Hence, we provide a different construction here to show that stateless k-head 2DPDAs are more powerful than stateless $(k-1)$-head 2DPDAs for any $k \geq 2$, and this result also holds for the nondeterministic machines (2NPDAs). Note that for 2NFAs, we were not able to show that 3 heads are better than 2 heads.

We describe a stateless k-head 2DPDA M' simulating M. The input alphabet of M' is $\Sigma' = \Sigma \cup \{\triangleright, \triangleleft, \#\} \cup \{i^{\mathsf{l}}, i^{\mathsf{r}} \mid 1 \leq i \leq m\}$. The left and right end markers of inputs to M' are now $\triangleright'$ and $\triangleleft'$, respectively. Note that $\triangleright, \triangleleft$ are now considered input symbols in Σ'.

Let $p = 1^{\mathsf{l}} 2^{\mathsf{l}} \cdots m^{\mathsf{l}} 1^{\mathsf{r}} 2^{\mathsf{r}} \cdots m^{\mathsf{r}}$. For an input $w = a_1 \cdots a_n$ to M, let $w' = \triangleright' a_1 p a_2 p \cdots a_n p \# a_1 \cdots a_n \triangleleft'$. We refer to w' as the encoding of w. We construct M' such that M accepts w if and only if M' accepts w'. For inputs to M' that are not valid encodings of inputs to M, we do not care. We now briefly describe the operation of M' when given input $w' = \rhd' a_1 p a_2 p \cdots a_n p \# a_1 \cdots a_n \lhd'$. For convenience, we also call the heads of M', $H_1, \cdots, H_k$.

M' simulates the computation of M on $w = a_1 \cdots a_n$ on input $w' = \rhd' a_1 p a_2 p \cdots a_n p \# a_1 \cdots a_n \lhd'$ as follows.

1. While H_1 remains on $\rhd'$, heads $H_2, \cdots, H_k$ are moved to $\#$.
2. Then $H_1, \cdots, H_k$ are moved right one cell. So now, H_1 is on the first a_1 and $H_2, \cdots, H_k$ are on the second a_1 (directly to the right of $\#$).

   In what follows, $H_2, \cdots, H_k$ will simulate the corresponding heads of M on $a_1 \cdots a_n$. H_1 will simulate H_1 of M on the segment $a_1 p a_2 p \cdots a_n p$. For convenience, let $w_1 = a_1 p a_2 p \cdots a_n p$ and $w_2 = a_1 \cdots a_n$.
3. The current state i of M is stored on the TOS of M' along with the stack symbol, as a pair (z, i).
4. At the beginning of each simulation step, H_1 will be on some symbol a_t of w_1 and $H_2, \cdots, H_k$ are on w_2, and the TOS is some pair (z, i). (Initially, the stack contains only $(B, 1)$, where 1 is the initial state of M.)
5. If in the move, M' is supposed to replace (z, i) by $(z_1 \cdots z_r, j)$, $r > 0$, M' moves $H_2, \cdots, H_k$ as in M and, depending on whether H_1 is supposed to move right, left, or remain stationary of a_t, M' does the following:
   (a) moves H_1 right and replaces the TOS (z, i) by $(z_1 \cdots (z_r, j^{\mathsf{r}})$;
   (b) moves H_1 left and replaces the TOS (z, i) by $(z_1 \cdots (z_r, j^{\mathsf{l}})$;
   (c) does not move H_1 but replaces the TOS (z, i) by $(z_1 \cdots (z_r, j)$.
   Then, we consider three cases.
   Case 1: TOS is (z_r, j^{r}). Then H_1 moves right until it reads a_{t+1}. It then replaces the TOS (z_r, j^{r}) by (z_r, j), and continues the simulation of the next step of M;
   Case 2: TOS is (z_r, j^{l}). Then H_1 moves left until it reads a_{t-1}. It then replaces the TOS (z_r, j^{l}) by (z_r, j), and it continues the simulation of the next step of M;
   Case 3: TOS is (z_r, j). M' continues the simulation of the next step of M.
6. If in the move, M' is supposed to replace (z, i) by (τ, j) (i.e., the stack is popped), then M' moves $H_2, \cdots, H_k$ as in M and, depending on whether H_1 is supposed to move right, left, or remain stationary of a_t does the following:
   (a) Moves H_1 right and replaces the TOS (z, i) by (τ, j^{r});
   (b) Moves H_1 left and replaces the TOS (z, i) by (τ, j^{l});
   (c) Moves H_1 right and replaces the TOS (z, i) by (τ, j^{s}).
   Again, we consider three cases:
   Case 4: TOS is (τ, j^{r}). Then H_1 moves right until it reads j^{r}. Then it pops the stack. Let the new top be y. Then M' replaces y by (y, j^{r}). Then M' operates like Case 1;
   Case 5: TOS is (τ, j^{l}). Then H_1 moves left until it reads j^{l}. Then it pops the stack. Let the new top be y. Then M' replaces y by (y, j^{l}). Then M' operates like Case 2;

Case 6: TOS is (τ, j^{s}). Then H_1 moves right until it reads j^{r}. Then it pops the stack. Let the new top be y. Then M' replaces y by (y, j^{l}). Then M' operates like Case 2.

It follows that M' can simulate M provided the input to M' is well-formed as described above.

Clearly, the above construction also works for nondeterministic machines, i.e., 2NPDAs.

Theorem 3. *For $k \geq 2$, stateless k-head 2DPDAs are computationally more powerful than stateless $(k-1)$-head 2DPDAs.*

Proof. Let L be a language accepted a k-head 2DPDA with states M but not by any $(k-1)$-head 2DPDA with states. Such a language exists [3]. Let M' be the stateless k-head 2DPDA simulating M as described above. Let $L' = L(M')$. We claim that L cannot be accepted by a stateless $(k-1)$-head 2DPDA. Suppose not, i.e., L' is accepted by a stateless $(k-1)$-head 2DPDA M''. We can then construct from M'' a $(k-1)$-head 2DPDA M''' with states to accept the original language $L = L(M)$ as follows.

When given $a_1 \cdots a_n$, M''' simulates the computation of M'' on $\triangleright' a_1 p \cdots a_n p \#$ $a_1 \cdots a_n \triangleleft'$. But since M''' is only given $a_1 \cdots a_n$ it will use the finite-state control to keep track of the movements of each head when each head "moves" into the segment $\triangleright a_1 p a_2 p \cdots a_n p \#$. Clearly, since p is a fixed pattern, this can be done. We omit the details. □

Similarly, since k-head 2NPDAs with states are computationally more powerful than those with k-heads [3], we have:

Theorem 4. *For $k \geq 2$, stateless k-head 2NPDAs are computationally more powerful than stateless $(k-1)$-head 2NPDAs.*

Turning now to the one-way varieties, consider the following language over the alphabet $\{a, b, \#, @\}$:

$$L_k = \{u_{\frac{k(k-1)}{2}} \# u_{\frac{k(k-1)}{2}-1} \# \ldots \# u_2 \# u_1 @ v_1 \# v_2 \# \ldots \# v_{\frac{k(k-1)}{2}-1} \# v_{\frac{k(k-1)}{2}} \mid u_i, v_i \in \{a, b\}^*, u_i = v_i\}.$$

It was shown in [11] that L_k can be accepted by a k-head 1DFA with states, but cannot be accepted by any $(k-1)$-head 1NFA with states. Later in [1] it was shown that, in fact, L_k cannot be accepted by any $(k-1)$-head 1NPDA with states.

Recently, in [4], it was shown that a language L'_k, which is a "padded" version of the language L_k, has the following properties:

1. L'_k can be accepted by a stateless k-head 1DFA, and therefore also by a stateless k-head 1DPDA;
2. If L'_k can be accepted by a stateless $(k-1)$-head 1NFA, then the unpadded version L_k can be accepted by a $(k-1)$-head 1NFA with states, and therefore also by a $(k-1)$-head 1NPDA with states.

In fact, following the argument in item 2 described in [4]), L'_k cannot be accepted by $(k-1)$-head 1NPDA with states.

Hence, L'_k can be accepted by a stateless k-head 1DFA (hence also by a stateless k-head 1DPDA) but not by any stateless $(k-1)$-head 1NPDA.

Theorem 5. *There are languages accepted by stateless k-head 1DFAs that cannot be accepted by stateless $(k-1)$-head 1NPDAs.*

Corollary 1. *For $k \geq 2$, stateless k-head 1DPDAs (resp., 1NPDAs) are computationally more powerful than stateless $(k-1)$-head 1DPDAs (resp., 1NPDAs).*

5 Characterizations

Clearly, any multihead 2DFA (resp., 2NFA, 2DPDA, 2NPDA) with n states can be simulated by a corresponding stateless machine by adding $log_2 n$ extra heads to keep track of the states. Each head is positioned on the left end marker (to represent 0) and the cell to its right (to represent 1). Thus, $log_2 n$ heads can keep track of n states.

Theorem 6. *Let L be a language. Then the following statements are equivalent:*

1. *L is accepted by a stateless multihead 2NPDA.*
2. *L is accepted by a stateless multihead 2DPDA.*
3. *L is accepted by a multihead 2NPDA with states.*
4. *L is accepted by a multihead 2DPDA with states.*
5. *L is accepted by a deterministic TM in n^c time for some constant c.*

The above follows from [2], where the equivalence of items 3, 4, and 5 was shown.

One can also easily show the following:

Theorem 7. *A language L can be accepted by stateless multihead 1DPDA (resp., 1NPDA, 1DFA, 1NFA) if and only if it can be accepted by a multihead 1DPDA (resp., 1NPDA, 1DFA, 1NFA) with states M with the following property: when M enters a state, say q, it can remain in that state and move the heads (and change the stack) but once it leaves q and it enters another state, say q', then M cannot reenter state q.*

It is well known that multihead 2DFAs (resp., 2NFAs) are equivalent to *log n* space-bounded deterministic (nondeterministic) Turing machines. Hence, we have:

Theorem 8. *Stateless multihead 2DFAs (resp., 2NFAs) are equivalent to log n space-bounded deterministic (resp., nondeterministic) Turing machines.*

It is a long-standing open problem whether *log n* space-bounded deterministic Turing machines are equivalent to *log n* space-bounded nondeterministic Turing machines. Here we show:

Theorem 9. *If every language accepted by a stateless* 3*-head 2NFA can be accepted by a multihead 2DFA with states, then log* n *space-bounded deterministic Turing machines are equivalent to log* n *space-bounded nondeterministic Turing machines.*

Proof. We will use the fact that this theorem is true when the 3-head 2NFA has states [8]. (In fact, the result in [8] is already true for 2 heads.)

Let L_1 be a language accepted by a 3-head 2NFA. Then, the language L_2 (defined in the proof of Lemma 2) can be accepted by a stateless 3-head 2NFA. If L_2 can be accepted by a multihead 2DFA with states, then we can easily construct a multihead 2DFA with states accepting L_1.

The result follows. □

It would be interesting to know if the 3-head above can be reduced to 2-head.

6 Conclusion

We showed that stateless $(k+1)$-head 2DFA (resp., 2NFAs) are computationally more powerful than k-head 2DFAs (resp., 2NFAS) for $k \geq 1$ (resp., $k = 1$ and $k \geq 3$), improving recent results in [4]. We also proved similar results for stateless multihead pushdown automata. Some interesting problems remain. For example, we conjecture that stateless 3-head 2NFAs are computationally more powerful than stateless 2-head 2NFAs, but we have no proof at this time.

We do not know if the relations between the number of heads, the size of Σ' and the length of w' presented in Section 3 are optimal. We think that such study, relating some features of the system to the size of the alphabet used in the simulation and the length of the input string, is important and that it is worth pursuing for other computing devices as well. We believe that the precise relationships between these measures depend on the kind of data structure (strings, multiset, sets, etc.) on which a system operates and on the 'power' of the transitions (rules, operation, etc.) used by the system.

References

1. Chrobak, M., Li, M.: k+1 heads are better than k for PDA's. In: Proceedings of the 27th Annual Symp. on Foundations of Computer Science, pp. 361–367 (1986)
2. Cook, S.: Variations on pushdown machines. In: Proceedings of the Annual ACM Symp. on Theory of Computing, pp. 229–231 (1969)
3. Ibarra, O.H.: On two-way multihead automata. J. of Computer and System Sciences 7, 28–36 (1973)
4. Ibarra, O.H., Karhumaki, J., Okhotin, A.: On stateless multihead automata: hierarchies and the emptiness problem. In: Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds.) LATIN 2008. LNCS, vol. 4957, pp. 94–105. Springer, Heidelberg (2008)
5. Monien, B.: Two-way multihead automata over a one-letter alphabet. RAIRO Informatique theoretique 14(1), 67–82 (1980)

6. Păun, G.: Computing with membranes. Journal of Computer and System Sciences 108, 143 (2000)
7. Păun, G.: Membrane Computing, An Introduction. Springer, Heidelberg (2002)
8. Sudborough, I.H.: On tape-bounded complexity classes and multi-head finite automata. In: Proc. of 14th Annual Symp. on Foundations of Computer Science, pp. 138–144 (1973)
9. Rosenberg, A.L.: On multi-head finite automata. IBM Journal of Research and Development 10(5), 388–394 (1966)
10. Yang, L., Dang, Z., Ibarra, O.H.: On stateless automata and P systems. In: Pre-Proceedings of Workshop on Automata for Cellular and Molecular Computing (August 2007)
11. Yao, A.C., Rivest, R.L.: $k + 1$ heads are better than k. Journal of the ACM 25(2), 337–340 (1978)

On Negative Bases

Christiane Frougny[1] and Anna Chiara Lai[2]

[1] LIAFA, CNRS UMR 7089, case 7014, 75205 Paris Cedex 13, France,
and University Paris 8,
Christiane.Frougny@liafa.jussieu.fr
[2] LIAFA, CNRS UMR 7089, and Me.Mo.Mat, Università La Sapienza, Rome, Italy
annachiara.lai@liafa.jussieu.fr

Abstract. We study expansions in non-integer negative base $-\beta$ introduced by Ito and Sadahiro [7]. Using countable automata associated with $(-\beta)$-expansions, we characterize the case where the $(-\beta)$-shift is a system of finite type. We prove that, if β is a Pisot number, then the $(-\beta)$-shift is a sofic system. In that case, addition (and more generally normalization on any alphabet) is realizable by a finite transducer.

1 Introduction

Expansions in integer negative base $-b$, where $b \geqslant 2$, seem to have been introduced by Grünwald in [6], and rediscovered by several authors, see the historical comments given by Knuth [8]. The choice of a negative base $-b$ and of the alphabet $\{0, \dots, b-1\}$ is interesting, because it provides a signless representation for every number (positive or negative). In this case it is easy to distinguish the sequences representing a positive integer from the ones representing a negative one: denoting $(w\raisebox{0.2ex}{.})_{-b} := \sum_{i=0}^{k} w_k(-b)^k$ for any $w = w_k \cdots w_0$ in$\{0, \dots, b-1\}^*$ with no leading 0's, we have $\mathbb{N} = \{(w\raisebox{0.2ex}{.})_{-b} \mid |w| \text{ is odd}\}$. The classical monotonicity between the lexicographical ordering on words and the represented numerical values does not hold anymore in negative base, for instance $3 = (111\raisebox{0.2ex}{.})_{-2}$, $4 = (100\raisebox{0.2ex}{.})_{-2}$ and $111 >_{lex} 100$. Nevertheless it is possible to restore such a correspondence by introducing an appropriate ordering on words, in the sequel denoted by $\prec$, and called the *alternate order*.

Representations in negative base also appear in some complex base number systems, for instance base $\beta = 2i$ where $\beta^2 = -4$ (see [5] for a study of their properties from an automata theoretic point of view). Thus, beyond the interest in the problem in itself, the authors also wish the study of negative bases to be an useful preliminar step to better understanding the complex case.

Ito and Sadahiro recently introduced expansions in non-integer negative base $-\beta$ in [7]. They have given a characterization of admissible sequences, and shown that the $(-\beta)$-shift is sofic if and only if the $(-\beta)$-expansion of the number $-\frac{\beta}{\beta+1}$ is eventually periodic.

In this paper we pursue their work. The purpose of this contribution is to show that many properties of the positive base (integer or not) numeration systems extend to the negative base case, the main difference being the sets of numbers

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 252–263, 2009.

that are representable in the two different cases. The results could seem not surprising, but this study put into light the important role played by the order on words: the lexicographic order for the positive bases, the alternate order for the negative bases.

We start by a general result which is not related to numeration systems but to the alternate order, and which is of interest in itself. We define a symbolic dynamical system associated with a given infinite word s satisfying some properties with respect to the alternate order on infinite words. We design an infinite countable automaton recognizing it. We then are able to characterize the case when the symbolic dynamical system is sofic (resp. of finite type). Using this general construction we can prove that the $(-\beta)$-shift is a symbolic dynamical system of finite type if and only if the $(-\beta)$-expansion of $-\frac{\beta}{\beta+1}$ is purely periodic. We also show that the entropy of the $(-\beta)$-shift is equal to $\log \beta$.

We then focus on the case where β is a Pisot number, that is to say, an algebraic integer greater than 1 such that the modulus of its Galois conjugates is less than 1. The natural integers and the Golden Mean are Pisot numbers. We extend all the results known to hold true in the Pisot case for β-expansions to the $(-\beta)$-expansions. In particular we prove that, if β is a Pisot number, then every number from $\mathbb{Q}(\beta)$ has an eventually periodic $(-\beta)$-expansion, and thus that the $(-\beta)$-shift is a sofic system.

When β is a Pisot number, it is known that addition in base β — and more generally normalization in base β on an arbitrary alphabet — is realizable by a finite transducer [4]. We show that this is still the case in base $-\beta$.

2 Definitions and Preliminaries

2.1 Words and Automata

An *alphabet* is a totally ordered set. In this paper the alphabets are always finite. A finite sequence of elements of an alphabet A is called a *word*, and the set of words on A is the free monoid A^*. The empty word is denoted by ε. The set of infinite (resp. bi-infinite) words on A is denoted by $A^{\mathbb{N}}$ (resp. $A^{\mathbb{Z}}$). Let v be a word of A^*, denote by v^n the concatenation of v to itself n times, and by v^ω the infinite concatenation $vvv\cdots$. A word of the form uv^ω is said to be *eventually periodic*. A (purely) *periodic* word is an eventually periodic word of the form v^ω.

A finite word v is a *factor* of a (finite, infinite or bi-infinite) word x if there exists u and w such that $x = uvw$. When u is the empty word, v is a *prefix* of x. The prefix v is *strict* if $v \neq x$. When w is empty, v is said to be a *suffix* of x.

We recall some definitions on automata, see [2] and [13] for instance. An *automaton over* A, $\mathcal{A} = (Q, A, E, I, T)$, is a directed graph labelled by elements of A. The set of vertices, traditionally called *states*, is denoted by Q, $I \subset Q$ is the set of *initial* states, $T \subset Q$ is the set of *terminal* states and $E \subset Q \times A \times Q$ is the set of labelled *edges*. If $(p, a, q) \in E$, we write $p \xrightarrow{a} q$. The automaton is *finite* if Q is finite. The automaton $\mathcal{A}$ is *deterministic* if E is the graph of a (partial) function from $Q \times A$ into Q, and if there is a unique initial state. A subset H of

A^* is said to be *recognizable by a finite automaton*, or *regular*, if there exists a finite automaton $\mathcal{A}$ such that H is equal to the set of labels of paths starting in an initial state and ending in a terminal state.

Recall that two words u and v are said to be *right congruent modulo H* if, for every w, uw is in H if and only if vw is in H. It is well known that H is recognizable by a finite automaton if and only if the congruence modulo H has finite index.

Let A and A' be two alphabets. A *transducer* is an automaton $\mathcal{T} = (Q, A^* \times A'^*, E, I, T)$ where the edges of E are labelled by couples in $A^* \times A'^*$. It is said to be *finite* if the set Q of states and the set E of edges are finite. If $(p, (u, v), q) \in E$, we write $p \xrightarrow{u|v} q$. The *input automaton* (resp. *output automaton*) of such a transducer is obtained by taking the projection of edges on the first (resp. second) component. A transducer is said to be *sequential* if its input automaton is deterministic.

The same notions can be defined for automata and transducer processing words from right to left : they are called *right* automata or transducers.

2.2 Symbolic Dynamics

Let us recall some definitions on symbolic dynamical systems or subshifts (see [10, Chapter 1] or [9]). The set $A^{\mathbb{Z}}$ is endowed with the lexicographic order, denoted $<_{lex}$, the product topology, and the shift σ, defined by $\sigma((x_i)_{i\in\mathbb{Z}}) = (x_{i+1})_{i\in\mathbb{Z}}$. A set $S \subseteq A^{\mathbb{Z}}$ is a *symbolic dynamical system*, or *subshift*, if it is shift-invariant and closed for the product topology on $A^{\mathbb{Z}}$. A bi-infinite word z *avoids* a set of word $X \subset A^*$ if no factor of z is in X. The set of all words which avoid X is denoted S_X. A set $S \subseteq A^{\mathbb{Z}}$ is a subshift if and only if S is of the form S_X for some X.

The same notion can be defined for a one-sided subshift of $A^{\mathbb{N}}$.

Let $F(S)$ be the set of factors of elements of S, let $I(S) = A^+ \setminus F(S)$ be the set of words avoided by S, and let $X(S)$ be the set of elements of $I(S)$ which have no proper factor in $I(S)$. The subshift S is *sofic* if and only if $F(S)$ is recognizable by a finite automaton, or equivalently if $X(S)$ is recognizable by a finite automaton. The subshift S is of *finite type* if $S = S_X$ for some finite set X, or equivalently if $X(S)$ is finite.

The topological entropy of a subshift S is

$$h(S) = \lim_{n\to\infty} \frac{1}{n} \log(B_n(S))$$

where $B_n(S)$ is the number of elements of $F(S)$ of length n. When S is sofic, the entropy of S is equal to the logarithm of the spectral radius of the adjacency matrix of the finite automaton recognizing $F(S)$.

2.3 Numeration Systems

The reader is referred to [10, Chapter 7] for a detailed presentation on these topics. Representations of real numbers in a non-integer base β were introduced by Rényi [12] under the name of β-*expansions*. Let x be a real number in the interval $[0,1]$. A *representation in base* β (or a β-representation) of x is an infinite word $(x_i)_{i\geqslant 1}$ such that

$$x=\sum_{i\geqslant 1} x_i\beta^{-i}.$$

A particular β-representation — called the β-*expansion* — can be computed by the "greedy algorithm" : denote by $\lfloor y\rfloor$, $\lceil y\rceil$ and $\{y\}$ the lower integer part, the upper integer part and the fractional part of a number y. Set $r_0=x$ and let for $i\geqslant 1$, $x_i=\lfloor \beta r_{i-1}\rfloor$, $r_i=\{\beta r_{i-1}\}$. Then $x=\sum_{i\geqslant 1} x_i\beta^{-i}$. The digits x_i are elements of the canonical alphabet $A_\beta=\{0,\ldots,\lceil\beta\rceil-1\}$.

The β-expansion of $x\in[0,1]$ will be denoted by $\mathsf{d}_\beta(x)=(x_i)_{i\geqslant 1}$. If $x>1$, there exists some $k\geqslant 1$ such that x/β^k belongs to $[0,1)$. If $\mathsf{d}_\beta(x/\beta^k)=(y_i)_{i\geqslant 1}$ then by shifting $x=(y_1\cdots y_k{\bullet}y_{k+1}y_{k+2}\cdots)_\beta$.

An equivalent definition is obtained by using the β-*transformation* of the unit interval which is the mapping

$$T_\beta : x\mapsto \beta x-\lfloor\beta x\rfloor.$$

Then $\mathsf{d}_\beta(x)=(x_i)_{i\geqslant 1}$ if and only if $x_i=\lfloor \beta T_\beta^{i-1}(x)\rfloor$.

If a representation ends in infinitely many zeros, like $v0^\omega$, the ending zeros are omitted and the representation is said to be *finite*.

In the case where the β-expansion of 1 is finite, there is a special representation playing an important role. Let $\mathsf{d}_\beta(1)=(t_i)_{i\geqslant 1}$ and set $\mathsf{d}^*_\beta(1)=\mathsf{d}_\beta(1)$ if $\mathsf{d}_\beta(1)$ is infinite and $\mathsf{d}^*_\beta(1)=(t_1\cdots t_{m-1}(t_m-1))^\omega$ if $\mathsf{d}_\beta(1)=t_1\cdots t_{m-1}t_m$ is finite.

Denote by D_β the set of β-expansions of numbers of $[0,1)$. It is a shift-invariant subset of $A_\beta^{\mathbb{N}}$. The β-*shift* S_β is the closure of D_β and it is a subshift of $A_\beta^{\mathbb{Z}}$. When β is an integer, S_β is the full β-shift $A_\beta^{\mathbb{Z}}$.

Theorem 1 (Parry[11]). *Let $\beta>1$ be a real number. A word $(w_i)_{i\geqslant 1}$ belongs to D_β if and only if for all $n\geqslant 1$*

$$w_nw_{n+1}\cdots <_{lex} \mathsf{d}^*_\beta(1).$$

A word $(w_i)_{i\in\mathbb{Z}}$ belongs to S_β if and only if for all n

$$w_nw_{n+1}\cdots \leqslant_{lex} \mathsf{d}^*_\beta(1).$$

The following results are well-known (see [10, Chapt. 7]).

Theorem 2. *1. The β-shift is sofic if and only if $\mathsf{d}_\beta(1)$ is eventually periodic.*
2. The β-shift is of finite type if and only if $\mathsf{d}_\beta(1)$ is finite.

It is known that the entropy of the β-shift is equal to $\log\beta$.

If β is a Pisot number, then every element of $\mathbb{Q}(\beta)\cap[0,1]$ has an eventually periodic β-expansion, and the β-shift S_β is a sofic system [1,14].

Let C be an arbitrary finite alphabet of integer digits. The *normalization function* in base β on C

$$\nu_{\beta,C} : C^{\mathbb{N}}\to \mathcal{A}_\beta^{\mathbb{N}}$$

is the partial function which maps an infinite word $y = (y_i)_{i \geqslant 1}$ over C, such that $0 \leqslant y = \sum_{i \geqslant 1} y_i \beta^{-i} \leqslant 1$, onto the β-expansion of y. It is known [4] that, when β is a Pisot number, normalization is computable by a finite transducer on any alphabet C. Note that addition is a particular case of normalization, with $C = \{0, \ldots, 2(\lceil \beta \rceil - 1)\}$.

3 Symbolic Dynamical Systems and the Alternate Order

Define the *alternate order* $\prec$ on infinite words or finite words with same length on an alphabet A:

$$x_1 x_2 x_3 \cdots \prec y_1 y_2 y_3 \cdots$$

if and only if there exists $k \geqslant 1$ such that

$$x_i = y_i \text{ for } 1 \leqslant i < k \quad \text{and} \quad (-1)^k (x_k - y_k) < 0.$$

This order was implicitely defined in [6].

Let A be a finite alphabet, and let $s = s_1 s_2 \cdots$ be a word in $A^{\mathbb{N}}$ such that $s_1 = \max A$ and for each $n \geqslant 1$, $s \preceq s_n s_{n+1} \cdots$. Let

$$S = \{w = (w_i)_{i \in \mathbb{Z}} \in A^{\mathbb{Z}} \mid \forall n,\ s \preceq w_n w_{n+1} \cdots \}.$$

We construct a countable infinite automaton $\mathcal{A}_S$ as follows (see Fig.1, where $[a, b]$ denotes $\{a, a+1, \ldots, b\}$ if $a \leqslant b$, ε else. It is assumed in Fig. 1 that $s_1 > s_j$ for $j \geqslant 2$.) The set of states is $\mathbb{N}$. For each state $i \geqslant 0$, there is an edge $i \xrightarrow{s_{i+1}} i+1$. Thus the state i is the name corresponding to the path labelled $s_1 \cdots s_i$. If i is even, then for each a such that $0 \leqslant a \leqslant s_{i+1} - 1$, there is an edge $i \xrightarrow{a} j$, where j is such that $s_1 \cdots s_j$ is the suffix of maximal length of $s_1 \cdots s_i a$. If i is odd, then for each b such that $s_{i+1} + 1 \leqslant b \leqslant s_1 - 1$, there is an edge $i \xrightarrow{b} j$ where j is maximal such that $s_1 \cdots s_j$ is a suffix of $s_1 \cdots s_i b$; and if $s_{i+1} < s_1$ there is one edge $i \xrightarrow{s_1} 1$. By contruction, the deterministic automaton $\mathcal{A}_S$ recognizes exactly the words w such that every suffix of w is $\succeq s$ and the result below follows.

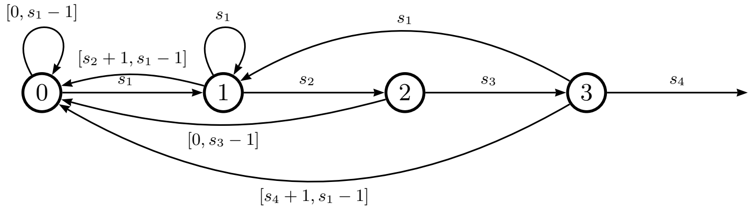


Fig. 1. The automaton $\mathcal{A}_S$

Proposition 1. *The subshift $S = \{w = (w_i)_{i \in \mathbb{Z}} \in A^{\mathbb{Z}} \mid \forall n,\ s \preceq w_n w_{n+1} \cdots \}$ is recognizable by the countable infinite automaton $\mathcal{A}_S$.*

Proposition 2. *The subshift $S = \{w = (w_i)_{i \in \mathbb{Z}} \in A^{\mathbb{Z}} \mid \forall n,\ s \preceq w_n w_{n+1} \cdots\}$ is sofic if and only if s is eventually periodic.*

Proof. The subshift S is sofic if and only if the set of its finite factors $F(S)$ is recognizable by a finite automaton. Given a word u of A^*, denote by $[u]$ the right class of u modulo $F(S)$. Then in the automaton $\mathcal{A}_S$, for each state $i \geqslant 1$, $i = [s_1 \cdots s_i]$, and $0 = [\varepsilon]$. Suppose that s is eventually periodic, $s = s_1 \cdots s_m (s_{m+1} \cdots s_{m+p})^{\omega}$, with m and p minimal. Thus, for each $k \geqslant 0$ and each $0 \leqslant i \leqslant p-1$, $s_{m+pk+i} = s_{m+i}$.
Case 1: p is even. Then $m+i = [s_1 \cdots s_{m+i}] = [s_1 \cdots s_{m+pk+i}]$ for every $k \geqslant 0$ and $0 \leqslant i \leqslant p-1$. Then the set of states of $\mathcal{A}_S$ is $\{0, 1, \ldots, m+p-1\}$.
Case 2: p is odd. Then $m+i = [s_1 \cdots s_{m+i}] = [s_1 \cdots s_{m+2pk+i}]$ for every $k \geqslant 0$ and $0 \leqslant i \leqslant 2p-1$. The set of states of $\mathcal{A}_S$ is $\{0, 1, \ldots, m+2p-1\}$.

Conversely, suppose that s is not eventually periodic. Then there exists an infinite sequence of indices $i_1 < i_2 < \cdots$ such that the sequences $s_{i_k} s_{i_k+1} \cdots$ are all different for all $k \geqslant 1$. Take any pair (i_j, i_ℓ), $j, \ell \geqslant 1$. If i_j and i_ℓ do not have the same parity, then $s_1 \cdots s_{i_j}$ and $s_1 \cdots s_{i_\ell}$ are not right congruent modulo $F(S)$. If i_j and i_ℓ have the same parity, there exists $q \geqslant 0$ such that $s_{i_j} \cdots s_{i_j+q-1} = s_{i_\ell} \cdots s_{i_\ell+q-1} = v$ and, for instance, $(-1)^{i_j+q}(s_{i_j+q} - s_{i_\ell+q}) > 0$ (with the convention that, if $q = 0$ then $v = \varepsilon$). Then $s_1 \cdots s_{i_j-1} v s_{i_j+q} \in F(S)$, $s_1 \cdots s_{i_\ell-1} v s_{i_\ell+q} \in F(S)$, but $s_1 \cdots s_{i_j-1} v s_{i_\ell+q}$ does not belong to $F(S)$. Hence $s_1 \cdots s_{i_j}$ and $s_1 \cdots s_{i_\ell}$ are not right congruent modulo $F(S)$, so the number of right congruence classes is infinite and $F(S)$ is thus not recognizable by a finite automaton. □

Proposition 3. *The subshift $S = \{w = (w_i)_{i \in \mathbb{Z}} \in A^{\mathbb{Z}} \mid \forall n,\ s \preceq w_n w_{n+1} \cdots\}$ is a subshift of finite type if and only if s is purely periodic.*

Proof. Suppose that $s = (s_1 \cdots s_p)^{\omega}$. Consider the finite set $X = \{s_1 \cdots s_{n-1} b \mid b \in A,\ (-1)^n (b - s_n) < 0,\ 1 \leqslant n \leqslant p\}$. We show that $S = S_X$. If w is in S, then w avoids X, and conversely. Now, suppose that S is of finite type. It is thus sofic, and by Proposition 2 s is eventually periodic. If it is not purely periodic, then $s = s_1 \cdots s_m (s_{m+1} \cdots s_{m+p})^{\omega}$, with m and p minimal, and $s_1 \cdots s_m \neq \varepsilon$. Let $I = \{s_1 \cdots s_{n-1} b \mid b \in A,\ (-1)^n (b - s_n) < 0,\ 1 \leqslant n \leqslant m\} \cup \{s_1 \cdots s_m (s_{m+1} \cdots s_{m+p})^{2k}\ s_{m+1} \cdots s_{m+n-1} b \mid b \in A,\ k \geqslant 0, (-1)^{m+2kp+n}(b - s_{m+n}) < 0,\ 1 \leqslant n \leqslant 2p\}$. Then $I \subset A^+ \setminus F(S)$. First, suppose there exists $1 \leqslant j \leqslant p$ such that $(-1)^j (s_j - s_{m+j}) < 0$ and $s_1 \cdots s_{j-1} = s_{m+1} \cdots s_{m+j-1}$. For $k \geqslant 0$ fixed, let $w^{(2k)} = s_1 \cdots s_m (s_{m+1} \cdots s_{m+p})^{2k} s_1 \cdots s_j \in I$. We have $s_1 \cdots s_m (s_{m+1} \cdots s_{m+p})^{2k} s_{m+1} \cdots s_{m+j-1} \in F(S)$. On the other hand, for $n \geqslant 2$, $s_n \cdots s_m (s_{m+1} \cdots s_{m+p})^{2k}$ is $\succ$ than the prefix of s of same length, thus $s_n \cdots s_m (s_{m+1} \cdots s_{m+p})^{2k} s_1 \cdots s_j \in F(S)$. Hence any strict factor of $w^{(2k)}$ is in $F(S)$. Therefore for any $k \geqslant 0$, $w^{(2k)} \in X(S)$, and $X(S)$ is thus infinite: S is not of finite type. Now, if such a j does not exist, then for every $1 \leqslant j \leqslant p$, $s_j = s_{m+j}$, and $s = (s_1 \cdots s_m)^{\omega}$ is purely periodic. □

Remark 1. Let $s' = s'_1 s'_2 \cdots$ be a word in $A^{\mathbb{N}}$ such that $s'_1 = \min A$ and, for each $n \geqslant 1$, $s'_n s'_{n+1} \cdots \preceq s'$. Let $S' = \{w = (w_i)_{i \in \mathbb{Z}} \in A^{\mathbb{Z}} \mid \forall n,\ w_n w_{n+1} \cdots \preceq s'\}$. The statements in Propositions 1, 2 and 3 are also valid for the subshift S' (with the automaton $\mathcal{A}_{S'}$ constructed accordingly).

4 Negative Real Base

4.1 The $(-\beta)$-Shift

Ito and Sadahiro [7] introduced a greedy algorithm to represent any real number in real base $-\beta$, $\beta > 1$, and with digits in $A_{-\beta} = \{0, 1, \ldots, \lfloor \beta \rfloor\}$. Remark that, when β is not an integer, $A_{-\beta} = A_\beta$.

A transformation on $I_{-\beta} = \left[-\frac{\beta}{\beta+1}, \frac{1}{\beta+1}\right)$ is defined as follows:

$$T_{-\beta}(x) = -\beta x - \lfloor -\beta x + \frac{\beta}{\beta+1} \rfloor.$$

For every real number $x \in I_{-\beta}$ denote $\mathsf{d}_{-\beta}(x)$ the $(-\beta)$-expansion of x. Then $\mathsf{d}_{-\beta}(x) = (x_i)_{i \geqslant 1}$ if and only if $x_i = \lfloor -\beta T_{-\beta}^{i-1}(x) + \frac{\beta}{\beta+1} \rfloor$, and $x = \sum_{i \geqslant 1} x_i (-\beta)^{-i}$. When this last equality holds, we may also write:

$$x = (\centerdot x_1 x_2 \cdots)_{-\beta}.$$

Since for every $x \in \mathbb{R} \backslash I_{-\beta}$ there exists an integer $k \geqslant 1$ such that $x/(-\beta)^k \in I_{-\beta}$, the sequence $\mathsf{d}_{-\beta}(x/(-\beta)^k) = (y_i)_{i \geqslant 1}$ satisfies $x = (y_1 \cdots y_k \centerdot y_{k+1} y_{k+2} \cdots)_{-\beta}$. Thus, allowing an opportune shift on the digits, every real number has a $(-\beta)$-expansion.

We show that the alternate order $\prec$ on $(-\beta)$-expansions gives the numerical order.

Proposition 4. *Let x and y be in $I_{-\beta}$. Then*

$$x < y \iff \mathsf{d}_{-\beta}(x) \prec \mathsf{d}_{-\beta}(y).$$

Proof. Suppose that $\mathsf{d}_{-\beta}(x) \prec \mathsf{d}_{-\beta}(y)$. Then there exists $k \geqslant 1$ such that $x_i = y_i$ for $1 \leqslant i < k$ and $(-1)^k (x_k - y_k) < 0$. Suppose that k is even, $k = 2q$. Then $x_{2q} \leqslant y_{2q} - 1$. Thus $x - y \leqslant -\beta^{-2q} + \sum_{i \geqslant 2q+1} x_i (-\beta)^{-i} - \sum_{i \geqslant 2q+1} y_i (-\beta)^{-i} < 0$, since $\sum_{i \geqslant 1} x_{2q+i} (-\beta)^{-i}$ and $\sum_{i \geqslant 1} y_{2q+i} (-\beta)^{-i}$ are in $I_{-\beta}$. The case $k = 2q+1$ is similar. The converse is immediate. □

Example 1. In base -2, $3 = (111\centerdot)_{-2}$, $4 = (100\centerdot)_{-2}$ and $111 \prec 100$.

A word $(x_i)_{i \geqslant 1}$ is said to be $(-\beta)$-*admissible* if there exists a real number $x \in I_{-\beta}$ such that $\mathsf{d}_{-\beta}(x) = (x_i)_{i \geqslant 1}$. The $(-\beta)$-*shift* $S_{-\beta}$ is the closure of the set of $(-\beta)$-admissible words, and it is a subshift of $A_\beta^{\mathbb{Z}}$.

Define the sequence $\mathsf{d}^*_{-\beta}(\frac{1}{\beta+1})$ as follows:

- if $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1}) = d_1 d_2 \cdots$ is not a periodic sequence with odd period,

$$\mathsf{d}^*_{-\beta}(\frac{1}{\beta+1}) = \mathsf{d}_{-\beta}(\frac{1}{\beta+1}) = 0 d_1 d_2 \cdots$$

– otherwise if $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1}) = (d_1 \cdots d_{2p+1})^\omega$,

$$\mathsf{d}^*_{-\beta}(\frac{1}{\beta+1}) = (0d_1 \cdots d_{2p}(d_{2p+1}-1))^\omega.$$

Theorem 3 (Ito-Sadahiro [7]). *A word $(w_i)_{i\geqslant 1}$ is $(-\beta)$-admissible if and only if for each $n \geqslant 1$*

$$\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1}) \preceq w_n w_{n+1} \cdots \prec \mathsf{d}^*_{-\beta}(\frac{1}{\beta+1}).$$

A word $(w_i)_{i\in\mathbb{Z}}$ is an element of the $(-\beta)$-shift if and only if for each n

$$\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1}) \preceq w_n w_{n+1} \cdots \preceq \mathsf{d}^*_{-\beta}(\frac{1}{\beta+1}).$$

Theorem 3 can be restated as follows.

Lemma 1. *Let $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1}) = d_1 d_2 \cdots$ and let*

$$S = \{(w_i)_{i\in\mathbb{Z}} \in A_\beta^{\mathbb{Z}} \mid \forall n,\ d_1 d_2 \cdots \preceq w_n w_{n+1} \cdots\}.$$

If $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1})$ is not a periodic sequence with odd period, then $S_{-\beta} = S$.
If $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1})$ is a periodic sequence of odd period, $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1}) = (d_1 \cdots d_{2p+1})^\omega$, then $S_{-\beta} = S \cap S'$ where

$$S' = \{(w_i)_{i\in\mathbb{Z}} \in A_\beta^{\mathbb{Z}} \mid \forall n,\ w_n w_{n+1} \cdots \preceq (0d_1 \cdots d_{2p}(d_{2p+1}-1))^\omega\}.$$

Theorem 4. *The $(-\beta)$-shift is a system of finite type if and only if $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1})$ is purely periodic.*

Proof. If $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1})$ is purely periodic with an even period, the result follows from Theorem 3, Lemma 1 and Proposition 3. If $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1})$ is purely periodic with an odd period, the result follows from Theorem 3, Lemma 1, Proposition 3, Remark 1, and the fact that the intersection of two finite sets is finite. □

By Theorem 3, Lemma 1, Proposition 2, Remark 1, and the fact that the intersection of two regular sets is again regular the following result follows.

Theorem 5 (Ito-Sadahiro [7]). *The $(-\beta)$-shift is a sofic system if and only if $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1})$ is eventually periodic.*

Example 2. Let $G = \frac{1+\sqrt{5}}{2}$; then $\mathsf{d}_{-G}(-\frac{G}{G+1}) = 10^\omega$, and the $(-G)$-shift is a sofic system which is not of finite type.
Let $\beta = G^2 = \frac{3+\sqrt{5}}{2}$. Then $\mathsf{d}_{-\beta}(-\frac{\beta}{\beta+1}) = (21)^\omega$ and the $(-\beta)$-shift is of finite type: the set of minimal forbidden factors is $X(S_{-\beta}) = \{20\}$.

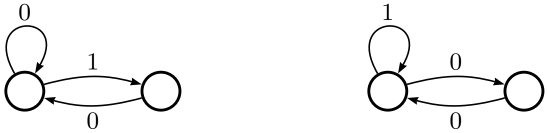


Fig. 2. Finite automata for the G-shift (left) and for the $(-G)$-shift (right)

Example 3. The automaton in Fig. 2 (right) recognizing the $(-G)$-shift is obtained by minimizing the result of the construction of Proposition 1. Remark that it is the automaton which recognizes the celebrated even shift (see [9]).

This example suggests that the entropy of the $-\beta$-shift is the same as the entropy of the β-shift. Using results from Fotiades and Boudourides [3], we can prove the following result.

Proposition 5. *The entropy of the $(-\beta)$-shift is equal to* $\log \beta$*.*

4.2 The Pisot Case

We first prove that the classical result saying that if β is a Pisot number, then every element of $\mathbb{Q}(\beta) \cap [0, 1]$ has an eventually periodic β-expansion is still valid for the base $-\beta$.

Theorem 6. *If β is a Pisot number, then every element of $\mathbb{Q}(\beta) \cap I_{-\beta}$ has an eventually periodic $(-\beta)$-expansion.*

Proof. Let $M_\beta(X) = X^d - a_1 X^{d-1} - \cdots - a_d$ be the minimal polynomial of β and denote by $\beta = \beta_1, \ldots, \beta_d$ the conjugates of β. Let x be arbitrarily fixed in $\mathbb{Q}(\beta) \cap I_{-\beta}$. Since $\mathbb{Q}(\beta) = \mathbb{Q}(-\beta)$, x can be expressed as $x = q^{-1} \sum_{i=0}^{d-1} p_i (-\beta)^i$ with q and p_i in $\mathbb{Z}$, $q > 0$ as small as possible in order to have uniqueness.

Let $(x_i)_{i \geqslant 1}$ be the $(-\beta)$-expansion of x, and write

$$r_n = r_n^{(1)} = r_n^{(1)}(x) = \frac{x_{n+1}}{-\beta} + \frac{x_{n+2}}{(-\beta)^2} + \cdots = (-\beta)^n \left(x - \sum_{k=1}^{n} x_k (-\beta)^{-k} \right).$$

Since $r_n = T_{-\beta}^n(x)$ belongs to $I_{-\beta}$ then $|r_n| \leqslant \frac{\beta}{\beta+1} < 1$. For $2 \leqslant j \leqslant d$, let

$$r_n^{(j)} = r_n^{(j)}(x) = (-\beta_j)^n \left(q^{-1} \sum_{i=0}^{d-1} p_i (-\beta_j)^i - \sum_{k=1}^{n} x_k (-\beta_j)^{-k} \right).$$

Let $\eta = \max\{|\beta_j| \mid 2 \leqslant j \leqslant d\}$: since β is a Pisot number, $\eta < 1$. Since $x_k \leqslant \lfloor \beta \rfloor$ we get

$$|r_n^{(j)}| \leqslant q^{-1} \sum_{i=0}^{d-1} |p_i| \eta^{n+i} + \lfloor \beta \rfloor \sum_{k=0}^{n-1} \eta^k$$

and since $\eta < 1$, $\max_{1 \leqslant j \leqslant d} \{\sup_n \{|r_n^{(j)}|\}\} < \infty$.

We need a technical result. Set $R_n = (r_n^{(1)}, \ldots, r_n^{(d)})$ and let B the matrix $B = ((-\beta_j)^{-i})_{1 \leqslant i,j \leqslant d}$.

Lemma 2. *Let $x = q^{-1}\sum_{i=0}^{d-1} p_i(-\beta)^i$. For every $n \geqslant 0$ there exists a unique d-uple $Z_n = (z_n^{(1)}, \dots, z_n^{(d)})$ in $\mathbb{Z}^d$ such that $R_n = q^{-1}Z_nB$.*

Proof. By induction on n. First, $r_1 = -\beta x - x_1$, thus

$$r_1 = q^{-1}\left(\sum_{i=0}^{d-1} p_i(-\beta)^{i+1} - qx_1\right) = q^{-1}\left(\frac{z_1^{(1)}}{-\beta} + \cdots + \frac{z_1^{(d)}}{(-\beta)^d}\right)$$

using the fact that $(-\beta)^d = -a_1(-\beta)^{d-1} + a_2(-\beta)^{d-2} + \cdots + (-1)^d a_d$. Now, $r_{n+1} = -\beta r_n - x_{n+1}$, hence

$$r_{n+1} = q^{-1}\left(z_n^{(1)} + \frac{z_1^{(2)}}{-\beta} + \cdots + \frac{z_n^{(d)}}{(-\beta)^{d-1}} - qx_{n+1}\right) = q^{-1}\left(\frac{z_{n+1}^{(1)}}{-\beta} + \cdots + \frac{z_{n+1}^{(d)}}{(-\beta)^d}\right)$$

since $z_n^{(1)} - qx_{n+1} \in \mathbb{Z}$. Thus for every n there exists $(z_n^{(1)}, \dots, z_n^{(d)})$ in $\mathbb{Z}^d$ such that

$$r_n = q^{-1}\sum_{k=1}^{d} z_n^{(k)}(-\beta)^{-k}.$$

Since the latter equation has integral coefficients and is satisfied by $-\beta$, it is also satisfied by $-\beta_j$, $2 \leqslant j \leqslant d$, and

$$r_n^{(j)} = (-\beta_j)^n\left(q^{-1}\sum_{i=0}^{d-1}\bar{p}_i(-\beta_j)^i - \sum_{k=1}^{n} x_k(-\beta_j)^{-k}\right) = q^{-1}\sum_{k=1}^{d} z_n^{(k)}(-\beta_j)^{-k}.$$

□

We go back to the proof of Theorem 6. Let $V_n = qR_n$. The $(V_n)_{n\geqslant 1}$ have bounded norm, since $\max_{1\leqslant j\leqslant d}\{\sup_n\{|r_n^{(j)}|\}\} < \infty$. As the matrix B is invertible, for every $n \geqslant 1$,

$$\|Z_n\| = \|(z_n^{(1)}, \dots, z_n^{(d)})\| = \max\{|z_n^{(j)}| \ : 1 \leqslant j \leqslant d\} < \infty$$

so there exist p and $m \geqslant 1$ such that $Z_{m+p} = Z_p$, hence $r_{m+p} = r_p$ and the $(-\beta)$-expansion of x is eventually periodic. □

As a corollary we get the following result.

Theorem 7. *If β is a Pisot number then the $(-\beta)$-shift is a sofic system.*

The *normalization* in base $-\beta$ is the function which maps any $(-\beta)$-representation on an alphabet C of digits of a given number of $I_{-\beta}$ onto the admissible $(-\beta)$-expansion of that number.

Let $C = \{-c, \dots, c\}$, where $c \geqslant \lfloor\beta\rfloor$ is an integer. Denote

$$Z_{-\beta}(2c) = \left\{(z_i)_{i\geqslant 0} \in \{-2c, \dots, 2c\}^{\mathbb{N}} \;\middle|\; \sum_{i\geqslant 0} z_i(-\beta)^{-i} = 0\right\}.$$

The set $Z_{-\beta}(2c)$ is recognized by a countable infinite automaton $\mathcal{A}_{-\beta}(2c)$: the set of states $Q(2c)$ consists of all $s \in \mathbb{Z}[\beta] \cap [-\frac{2c}{\beta-1}, \frac{2c}{\beta-1}]$. Transitions are of the form $s \xrightarrow{e} s'$ with $e \in \{-c, \ldots, c\}$ such that $s' = -\beta s + e$. The state 0 is initial; every state is terminal.

Let $M_\beta(X)$ be the minimal polynomial of β, and denote by $\beta = \beta_1, \ldots, \beta_d$ the conjugates of β. We define a norm on the discrete lattice of rank d, $\mathbb{Z}[X]/(M_\beta)$, as

$$||P(X)|| = \max_{1 \leqslant i \leqslant d} |P(\beta_i)|.$$

Proposition 6. *If β is a Pisot number then the automaton $\mathcal{A}_{-\beta}(2c)$ is finite for every $c \geqslant \lfloor \beta \rfloor$.*

Proof. Every state s in $Q(2c)$ is associated with the label of the shortest path $f_0 f_1 \cdots f_n$ from 0 to s in the automaton. Thus $s = f_0(-\beta)^n + f_1(-\beta)^{n-1} + \cdots + f_n = P(\beta)$, with $P(X)$ in $\mathbb{Z}[X]/(M_\beta)$. Since $f_0 f_1 \cdots f_n$ is a prefix of a word of $Z_{-\beta}(2c)$, there exists $f_{n+1} f_{n+2} \cdots$ such that $(f_i)_{i \geqslant 0}$ is in $Z_{-\beta}(2c)$. Thus $s = |P(\beta)| < \frac{2c}{\beta-1}$. For every conjugate β_i, $2 \leqslant i \leqslant d$, $|\beta_i| < 1$, and $|P(\beta_i)| < \frac{2c}{1-|\beta_i|}$. Thus every state of $Q(2c)$ is bounded in norm, and so there is only a finite number of them. □

The *redundancy transducer* $\mathcal{R}_{-\beta}(c)$ is similar to $\mathcal{A}_{-\beta}(2c)$. Each transition $s \xrightarrow{e} s'$ of $\mathcal{A}_{-\beta}(2c)$ is replaced in $\mathcal{R}_{-\beta}(c)$ by a set of transitions $s \xrightarrow{a|b} s'$, with $a, b \in \{-c, \ldots, c\}$ and $a - b = e$. Thus one obtains the following proposition.

Proposition 7. *The redundancy transducer $\mathcal{R}_{-\beta}(c)$ recognizes the set*

$$\big\{(x_1 x_2 \cdots, y_1 y_2 \cdots) \in C^{\mathbb{N}} \times C^{\mathbb{N}} \mid \sum_{i \geqslant 1} x_i (-\beta)^{-i} = \sum_{i \geqslant 1} y_i (-\beta)^{-i} \big\}.$$

If β is a Pisot number, then $\mathcal{R}_{-\beta}(c)$ is finite.

Theorem 8. *If β is a Pisot number, then normalization in base $-\beta$ on any alphabet C is realizable by a finite transducer.*

Proof. The normalization is obtained by keeping in $\mathcal{R}_{-\beta}(c)$ only the outputs y that are $(-\beta)$-admissible. By Theorem 7 the set of admissible words is recognizable by a finite automaton $\mathcal{D}_{-\beta}$. The finite transducer $\mathcal{N}_{-\beta}(c)$ doing the normalization is obtained by making the intersection of the output automaton of $\mathcal{R}_{-\beta}(c)$ with $\mathcal{D}_{-\beta}$. □

Proposition 8. *If β is a Pisot number, then the conversion from base $-\beta$ to base β is realizable by a finite transducer. The result is β-admissible.*

Proof. Let $x \in I_{-\beta}$, $x \geqslant 0$, such that $\mathsf{d}_{-\beta}(x) = x_1 x_2 x_3 \cdots$. Denote $\bar{a}$ the signit digit $(-a)$. Then $\overline{x_1} x_2 \overline{x_3} \cdots$ is a β-representation of x on the alphabet $\widetilde{A_{-\beta}} = \{-\lfloor \beta \rfloor, \ldots, \lfloor \beta \rfloor\}$. Thus the conversion is equivalent to the normalization in base β on the alphabet $\widetilde{A_{-\beta}}$, and when β is a Pisot number, it is realizable by a finite transducer by [4]. □

Remark 2. In the case where the base is a negative integer, conversion from base b to base $-b$ is realizable by a finite right sequential transducer. In a forthcoming paper we show that conversion from base β to base $-\beta$ — with the result in non-admissible form — is realizable by a finite left sequential transducer when β is a Pisot number.

References

1. Bertrand, A.: Développements en base de Pisot et répartition modulo 1. C. R. Acad. Sci. Paris Sér A-B 285, A419–A421 (1977)
2. Eilenberg, S.: Automata, Languages and Machines, vol. A. Academic Press, London (1974)
3. Fotiades, N., Boudourides, M.: Topological conjugacies of piecewise monotone interval maps. Int. J. Math. Math. Sci. 25, 119–127 (2001)
4. Frougny, C.: Representations of numbers and finite automata. Math. Systems Theory 25, 37–60 (1992)
5. Frougny, C.: On-line finite automata for addition in some numeration systems. Theoretical Informatics and Applications 33, 79–101 (1999)
6. Grünwald, V.: Intorno all'aritmetica dei sistemi numerici a base negativa con particolare riguardo al sistema numerico a base negativo-decimale per lo studio delle sue analogie coll'aritmetica ordinaria (decimale). Giornale di Matematiche di Battaglini 367, 203–221 (1885)
7. Ito, S., Sadahiro, T.: $(-\beta)$-expansions of real numbers. In: INTEGERS (to appear)
8. Knuth, D.E.: The Art of Computer Programming, Seminumerical Algorithms, 2nd edn., vol. 2. Addison-Wesley, Reading (1988)
9. Lind, D., Marcus, B.: An introduction to symbolic dynamics and coding. Cambridge University Press, Cambridge (1995)
10. Lothaire, M.: Algebraic combinatorics on words. Encyclopedia of Mathematics and its Applications, vol. 90. Cambridge University Press, Cambridge (2002)
11. Parry, W.: On the β-expansions of real numbers. Acta Math. Acad. Sci. Hungar. 11, 401–416 (1960)
12. Rényi, A.: Representations for real numbers and their ergodic properties. Acta Math. Acad. Sci. Hungar. 8, 477–493 (1957)
13. Sakarovitch, J.: Eléments de théorie des automates, Vuibert (2003); English translation: Elements of Automata Theory, Cambridge University Press, Cambridge (to appear)
14. Schmidt, K.: On periodic expansions of Pisot numbers and Salem numbers. Bull. London Math. Soc. 12, 269–278 (1980)

Crucial Words for Abelian Powers*

Amy Glen, Bjarni V. Halldórsson, and Sergey Kitaev

The Mathematics Institute, Reykjavík University
Kringlan 1, 103 Reykjavík, Iceland
amy.glen@gmail.com, bjarnivh@ru.is, sergey@ru.is

Abstract. Let $k \geq 2$ be an integer. An *abelian k-th power* is a word of the form $X_1X_2 \cdots X_k$ where X_i is a permutation of X_1 for $2 \leq i \leq k$. In this paper, we consider *crucial words* for abelian k-th powers, i.e., finite words that avoid abelian k-th powers, but which cannot be extended to the right by any letter of their own alphabets without creating an abelian k-th power. More specifically, we consider the problem of determining the minimal length of a crucial word avoiding abelian k-th powers. This problem has already been solved for abelian squares by Evdokimov and Kitaev [6], who showed that a *minimal crucial word* over an n-letter alphabet $\mathcal{A}_n = \{1, 2, \ldots, n\}$ avoiding *abelian squares* has length $4n - 7$ for $n \geq 3$. Extending this result, we prove that a minimal crucial word over $\mathcal{A}_n$ avoiding abelian cubes has length $9n - 13$ for $n \geq 5$, and it has length 2, 5, 11, and 20 for $n = 1, 2, 3,$ and 4, respectively. Moreover, for $n \geq 4$ and $k \geq 2$, we give a construction of length $k^2(n-1) - k - 1$ of a crucial word over $\mathcal{A}_n$ avoiding abelian k-th powers. This construction gives the minimal length for $k = 2$ and $k = 3$.

Keywords: pattern avoidance; abelian square-free word; abelian cube-free word; abelian power; crucial word; Zimin word.

MSC (2000): 05D99; 68R05; 68R15.

1 Introduction

Let $\mathcal{A}_n = \{1, 2, \ldots, n\}$ be an n-letter alphabet and let $k \geq 2$ be an integer. A word W over $\mathcal{A}_n$ contains a *k-th power* if W has a factor of the form $X^k = XX \cdots X$ (k times) for some non-empty word X. A k-th power is *trivial* if X is a single letter. For example, the word $V = 13243232323243$ over $\mathcal{A}_4$ contains the (non-trivial) 4-th power $(32)^4 = 32323232$. A word W contains an *abelian k-th power* if W has a factor of the form $X_1X_2 \cdots X_k$ where X_i is a permutation of X_1 for $2 \leq i \leq k$. The cases $k = 2$ and $k = 3$ give us (*abelian*) *squares* and *cubes*, respectively. For instance, the preceding word V contains the abelian square 43232 32324 and the word 123 312 213 is an abelian cube. A word W is (*abelian*) *k-power-free* if W *avoids* (abelian) k-th powers, i.e., if W does not contain any

* The first and third authors were supported by the Icelandic Research Fund (grant nos. 09003801/1 and 060005012/3).

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 264–275, 2009.

(abelian) k-th powers. For example, the word 1234324 is abelian cube-free, but not abelian square-free since it contains the abelian square 234 324.

A word W is *crucial* with respect to a given set of *prohibited words* (or simply *prohibitions*) if W avoids the prohibitions, but Wx does not avoid the prohibitions for any letter x occurring in W. A *minimal crucial word* is a crucial word of the shortest length. For example, the word $W = 21211$ (of length 5) is crucial with respect to abelian cubes since it is abelian cube-free and the words $W1$ and $W2$ end with the abelian cubes 111 and 21 21 12, respectively. Actually, W is a minimal crucial word over $\{1, 2\}$ with respect to abelian cubes. Indeed, one can easily verify that there do not exist any crucial abelian cube-free words on two letters of length less than 5.

Abelian squares were first introduced by Erdős [4], who asked whether or not there exist words of arbitrary length over a fixed finite alphabet that avoid patterns of the form XX' where X' is a permutation of X (i.e., abelian squares). This question has since been solved in the affirmative in a series of papers from 1968 to 1992 (see [5,9,7] and also [2]). Problems of this type were also considered by Zimin [10], who used the following sequence of words as a key tool.

The *Zimin word* Z_n over $\mathcal{A}_n$ is defined recursively as follows: $Z_1 = 1$ and $Z_n = Z_{n-1}nZ_{n-1}$ for $n \geq 2$. The first four Zimin words are:

$$Z_1 = 1,\ Z_2 = 121,\ Z_3 = 1213121,\ Z_4 = 121312141213121.$$

The *k-generalized Zimin word* $Z_{n,k} = X_n$ is defined as

$$X_1 = 1^{k-1} = 11\cdots 1,\ X_n = (X_{n-1}n)^{k-1}X_{n-1} = X_{n-1}nX_{n-1}n\cdots nX_{n-1}$$

where the number of 1's, as well as the number of n's, is $k-1$. Thus $Z_n = Z_{n,2}$. It is easy to see (by induction) that each $Z_{n,k}$ avoids (abelian) k-th powers and has length $k^n - 1$. Moreover, it is known that $Z_{n,k}$ gives the length of a minimal crucial word avoiding k-th powers.

However, much less is known in the case of abelian powers. Crucial abelian square-free words (also called *right maximal abelian square-free words*) of exponential length are given in [3] and [6], and it is shown in [6] that a minimal crucial abelian square-free word over an n-letter alphabet has length $4n-7$ for $n \geq 3$.

In this paper, we extend the study of crucial abelian k-power-free words to the case of $k > 2$. In particular, we provide a complete solution to the problem of determining the length of a minimal crucial abelian cube-free word (the case $k = 3$) and we conjecture a solution in the general case. More precisely, we show that a minimal crucial word over $\mathcal{A}_n$ avoiding abelian cubes has length $9n - 13$ for $n \geq 5$ (Corollary 1), and it has length 2, 5, 11, and 20 for $n = 1, 2, 3$, and 4, respectively. For $n \geq 4$ and $k \geq 2$, we give a construction of length $k^2(n-1)-k-1$ of a crucial word over $\mathcal{A}_n$ avoiding abelian k-th powers (see Theorem 5). This construction gives the minimal length for $k = 2$ and $k = 3$, and we conjecture that this is also true for any $k \geq 4$ and sufficiently large n. We also provide a rough lower bound for the length of minimal crucial words over $\mathcal{A}_n$ avoiding abelian k-th powers, for $n \geq 5$ and $k \geq 4$ (see Theorem 6).

For a crucial word X over $\mathcal{A}_n$, we let $X = X_i\varDelta_i$ where $\varDelta_i$ is the factor of minimal length such that $\varDelta_i i$ is a prohibition for $i \in \mathcal{A}_n$. Note that we can rename letters, if needed, so we can assume that for any minimal crucial word X, one has

$$\varDelta_1 \subset \varDelta_2 \subset \cdots \subset \varDelta_n = X$$

where "$\subset$" means (proper) *right factor* (or *suffix*). In other words, for each $i = 2, 3, \ldots, n$, we have $\varDelta_i = Y_i\varDelta_{i-1}$ for some non-empty Y_i. In what follows we will use X_i and Y_i as stated above. We note that the definitions imply:

$$X = X_i\varDelta_i = X_iY_i\varDelta_{i-1} = X_{n-1}Y_{n-1}Y_{n-2}\cdots Y_2\varDelta_1,$$

for any $i = 2, 3, \ldots, n-1$. Furthermore, in the case of crucial words avoiding abelian k-th powers, we write $\varDelta_i i = \varOmega_{i,1}\varOmega_{i,2}\cdots\varOmega_{i,k}$, where the k *blocks* $\varOmega_{i,j}$ are equal up to permutation, and we denote by $\varOmega'_{i,k}$ the block $\varOmega_{i,k}$ without the rightmost i.

Hereafter, we let $\ell_k(n)$ denote the length of a minimal crucial word over $\mathcal{A}_n$ avoiding abelian k-th powers. The length of a word W is denoted by $|W|$, and we denote by $|W|_x$ the number of occurrences of a letter x in W. The *Parikh vector* of a word W over $\mathcal{A}_n$ is defined by

$$\mathcal{P}(W) := (|W|_1, |W|_2, \ldots, |W|_n).$$

Clearly, if W is an abelian k-th power, then $|W|_x \equiv 0 \pmod{k}$ for all letters x occurring in W.

2 Crucial Words for Abelian Cubes

2.1 An Upper Bound for $\ell_3(n)$

The fact that the 3-generalized Zimin word $Z_{n,3}$ is crucial with respect to abelian cubes already gives us an upper bound of $3^n - 1$ for $\ell_3(n)$. In Theorem 1 (below) we improve this upper bound to $3 \cdot 2^{n-1} - 1$. We then give a construction of a crucial abelian cube-free word over $\mathcal{A}_n$ of length $9n - 13$, which coincides with the lower bound given in Theorem 3 of Sec. 2.2 for $n \geq 5$.

Theorem 1. *One has that $\ell_3(n) \leq 3 \cdot 2^{n-1} - 1$.*

Proof. We construct a crucial abelian cube-free word $X = X_n$ iteratively as follows. Set $X_1 = 11$ and assume X_{n-1} has been constructed. Then do the following:

1. Increase all letters of X_{n-1} by 1 to obtain X'_{n-1}.
2. Insert 1 after (to the right of) each letter of X'_{n-1} and adjoin one extra 1 to the right of the resulting word to get X_n.

For example, $X_2 = 21211$, $X_3 = 31213121211$, etc. It is easy to verify that $|X_n| = 3\cdot 2^{n-1}-1$. We show by induction that X_n avoids abelian cubes, whereas $X_n x$ does not avoid abelian cubes for any $x \in \mathcal{A}_n$. Both claims are trivially true for $n = 1$. Now take $n \geq 2$. If X_n contains an abelian cube, then removing all 1's from it, we would deduce that X_{n-1} must also contain an abelian cube, contradicting the fact that X_{n-1} contains no abelian cubes.

It remains to show that extending X_n to the right by any letter x from $\mathcal{A}_n$ creates an abelian cube. If $x = 1$ then we get 111 from the construction of X_n. On the other hand, if $x > 1$ then we swap the rightmost 1 with the rightmost x in Xx, thus obtaining a word where every other letter is 1; removing all 1's and decreasing each of the remaining letters by 1, we have $X_{n-1}(x-1)$, which contains an abelian cube (by the induction hypothesis). □

Remark 1. We observe that the "greedy" construction used in the proof of the above theorem actually yields *minimal* crucial abelian cube-free words over $\mathcal{A}_n$ of lengths 2, 5, 11 for $n = 1, 2, 3$, respectively (verified by exhaustive search). For $n = 4$, one can also verify that a minimal crucial word avoiding abelian cubes has length 20. For example, the word 41213124213121312211 is a minimal crucial word with respect to abelian cubes.

A construction giving the best possible upper bound for $n \geq 5$ can be easily described by examples, and we do this below (for $n = 4, 5, 6, 7$; the construction does not work for $n \leq 3$). We also provide a general description. The pattern in the construction is easy to recognize.

An optimal construction for crucial abelian cube-free words. The construction of the word E_n for $n = 4, 5, 6, 7$ works as follows. We use spaces to separate the blocks $\Omega_{n,1}$, $\Omega_{n,2}$, and $\Omega'_{n,3}$ in $E_n = \Delta_n$.

$$E_4 = 34423311\ 34231134\ 3233411$$

$$E_5 = 45534423311\ 45342311345\ 4323344511$$

$$E_6 = 56645534423311\ 56453423113456\ 5432334455611$$

$$E_7 = 67756645534423311\ 67564534231134567\ 6543233445566711$$

In general, the block $\Omega_{n,1}$ in $E_n = \Delta_n = \Omega_{n,1}\Omega_{n,2}\Omega'_{n,3}$ is built by adjoining the factors $i(i+1)(i+1)$ for $i = n-1, n-2, \ldots, 2$, followed by two 1's. The block $\Omega_{n,2}$ is built by adjoining the following factors: $i(i+1)$ for $i = n-1, n-2, \ldots, 2$, followed by 11, and then the factor $34\cdots(n-1)n$. Finally, the block $\Omega'_{n,3}$ is built by adjoining the factors $(n-1)(n-2)\cdots 32$, then xx for $3 \leq x \leq n-1$, followed by n, and finally two 1's.

We have $E_n = \Omega_{n,1}\Omega_{n,2}\Omega'_{n,3}$ where $\Omega_{n,3} = \Omega'_{n,3}n$, and by construction each $\Omega_{n,i}$ contains two 1's, one 2, two n's, and three x's for $x = 3, \ldots, n-1$. That is, for each $i = 1, 2, 3$, the Parikh vector of the block $\Omega_{n,i}$ is given by $\mathcal{P}(\Omega_{n,i}) = (2, 1, 3, 3, \ldots, 3, 2)$. Hence, $\mathcal{P}(E_n) = (6, 3, 9, 9, \ldots, 9, 5)$, and therefore $|E_n| = 6 + 3 + 9(n-3) + 5 = 9n - 13$. Moreover, for all $n \geq 4$, the word E_n is crucial with respect to abelian cubes. We omit the proof of this latter fact since it is very similar to the proof of Theorem 5 (to follow), from which the fact can actually be deduced by setting $k = 3$ (in view of Remark 4, later). Thus, a minimal crucial word avoiding abelian cubes has length at most $9n - 13$ for $n \geq 4$. That is:

Theorem 2. *For $n \geq 4$, we have $\ell_3(n) \leq 9n - 13$.*

Proof. Theorem 5 with $k = 3$. □

2.2 A Lower Bound for $\ell_3(n)$

If $X = \Delta_n$ is a crucial word over $\mathcal{A}_n$ with respect to abelian cubes, then clearly the number of occurrences of each letter except n must be divisible by 3, whereas the number of occurrences of n is 2 modulo 3. We sort in non-decreasing order the number of occurrences of the letters $1, 2, \ldots, n-1$ in X to get a non-decreasing sequence of numbers $(a_1 \leq a_2 \leq \cdots \leq a_{n-1})$. Notice that a_i does not necessarily correspond to the letter i. We denote by a_0 the number of occurrences of the letter n. Also note that a_0 can be either larger or smaller than a_1. By definitions, $|X| = \sum_{i=0}^{n-1} a_i$.

For example, the abelian cube-free crucial word E_n of length $9n-13$ in Sec. 2.1 has the following sequence of a_i's: $(a_0, a_1, \ldots, a_{n-1}) = (5, 3, 6, 9, \ldots, 9)$. In this subsection, we prove that this sequence cannot be improved for $n \geq 5$, meaning that, e.g., 5 cannot be replaced by 2, and/or 6 cannot be replaced by 3, and/or 9('s) cannot be replaced by 3('s) or 6('s), no matter what construction we use to form a crucial word. This is a direct consequence of Lemmas 2–5 (below) and is recorded in Theorem 3. In the rest of this section we use, without explanation, the following two facts that are easy to see from the definitions. For any letter x in a crucial abelian cube-free word X over $\mathcal{A}_n$:

1. $|\Delta_x|_x \equiv 2 \pmod 3$ and $|\Delta_x|_y \equiv 0 \pmod 3$ for any other letter y occurring in X.
2. If $x + 1$ occurs in X, then we have $\Delta_{x+1} = Y_{x+1}\Delta_x$ where $|Y_{x+1}|_{x+1} \equiv 2 \pmod 3$, $|Y_{x+1}|_x \equiv 1 \pmod 3$, and $|Y_{x+1}|_y \equiv 0 \pmod 3$ for any other letter y occurring in X.

The following fact will also be useful.

Lemma 1. *Suppose X is a crucial abelian cube-free word over $\mathcal{A}_n$ containing letters x and y such that $x < y < n$ and $|X|_x = |X|_y = 6$. Then Δ_x cannot contain 5 occurrences of the letter x.*

Proof. Suppose to the contrary that (under the hypotheses of the lemma) Δ_x contains 5 occurrences of the letter x. Let $A_1 = Y_nY_{n-1}\cdots Y_{y+1}$ and $A_2 = Y_yY_{y-1}\cdots Y_{x+1}$ so that $X = A_1A_2\Delta_x$. Then $|A_1A_2|_x = 1$ and $|A_1A_2|_y \geq 3$, contradicting the fact that each of the blocks $\Omega_{n,1}$, $\Omega_{n,2}$, and $\Omega'_{n,3}$ in $X = \Delta_n = \Omega_{n,1}\Omega_{n,2}\Omega'_{n,3}$ must each receive two x's and two y's.

Lemma 2. *For a crucial abelian cube-free word X over $\mathcal{A}_n$, the sequence of a_i's cannot contain $3, 3$. That is, $(a_1, a_2) \neq (3, 3)$.*

Proof. Suppose that x and y are letters such that $x < y < n$ and $|X|_x = |X|_y = 3$. Let $A_1 = Y_nY_{n-1}\cdots Y_{y+1}$ and $A_2 = Y_yY_{y-1}\cdots Y_{x+1}$ so that we have $X = A_1A_2\Delta_x$. Then we must have the following distribution of x's and y's in X: $|A_1|_y = 1$, $|A_2|_y = 2$, $|A_2|_x = 1$, and $|\Delta_x|_x = 2$. However, we get a contradiction, since each of the blocks $\Omega_{n,2}$ and $\Omega'_{n,3}$ in $X = \Delta_n = \Omega_{n,1}\Omega_{n,2}\Omega'_{n,3}$ must receive one copy of x and one copy of y, which is impossible (no x can exist between the two rightmost y's). □

Lemma 3. *For a crucial abelian cube-free word X over $\mathcal{A}_n$, the sequence of a_i's cannot contain* $6, 6, 6$.

Proof. Suppose that x, y, z are three letters such that $x < y < z < n$ and $|X|_x = |X|_y = |X|_z = 6$. Let $A_1 = Y_nY_{n-1}\cdots Y_{z+1}$, $A_2 = Y_zY_{z-1}\cdots Y_{y+1}$, and $A_3 = Y_yY_{y-1}\cdots Y_{x+1}$ so that $X = A_1A_2A_3\Delta_x$. Then the minimal requirements on the A_i are as follows: $|A_1|_z \geq 1$, $|A_2|_z \geq 2$, $|A_2|_y \geq 1$, $|A_3|_y \geq 2$, and $|A_3|_x \geq 1$. Moreover, applying Lemma 1 to x and y, we have $|\Delta_x|_2 = 2$. And apply the same lemma to the letters y and z guarantees that A_1A_2 contains 4 y's (in particular, Δ_x does not contain any y's).

Looking at $X = \Delta_n = \Omega_{n,1}\Omega_{n,2}\Omega'_{n,3}$, we see that for each $i = 1, 2, 3$, $|\Omega_{n,i}|_x = |\Omega_{n,i}|_y = |\Omega_{n,i}|_z = 2$. Thus, in A_3, we must have the following order of letters: x, y, y and the boundary between $\Omega_{n,2}$ and $\Omega'_{n,3}$ must be between x and y in A_3. But then Δ_x entirely belongs to $\Omega'_{n,3}$, so it cannot contain any z's (if it would do so, Δ_x would then contain 3 z's which is impossible). On the other hand, we must have $|A_3|_z = 3$ for $\Omega'_{n,3}$ to receive 2 z's. Thus, Δ_y contains 2 y's, 3 z's, and 3 x's, which is impossible by Lemma 2 applied to the word Δ_y with two letters occurring exactly 3 times each. (Alternatively, one can see, due to the considerations above, that no z can be between the two rightmost x's, contradicting the structure of Δ_y). □

Lemma 4. *For a crucial abelian cube-free word X over $\mathcal{A}_n$, the sequence of a_i's cannot contain* $3, 6, 6$.

Proof. Suppose that x, y, and z are letters such that $|X|_x = 3$ and $|X|_y = |X|_z = 6$. We consider three cases covering all the possibilities up to renaming y and z.

Case 1: $z < y < x < n$. One can see that Δ_y does not contain x, but it contains at least 3 z's contradicting the fact that each of the blocks $\Omega_{n,1}$, $\Omega_{n,2}$, and $\Omega'_{n,3}$ must receive 1 x and 2 z's.

Case 2: $x < z < y < n$. We let $A = Y_nY_{n-1}\cdots Y_{z+1}$ so that $X = A\Delta_z$. All three x's must be in Δ_z, while A must contain at least 3 y's contradicting the fact that each of the blocks $\Omega_{n,1}$, $\Omega_{n,2}$, and $\Omega'_{n,3}$ must receive 1 x and 2 y's.

Case 3: $z < x < y < n$. We let $A_1 = Y_nY_{n-1}\cdots Y_{y+1}$, $A_2 = Y_yY_{y-1}\cdots Y_{x+1}$, and $A_3 = Y_xY_{x-1}\cdots Y_{z+1}$ so that $X = A_1A_2A_3\Delta_z$. The minimal requirements on the A_i and Δ_z are as follows: $|A_1|_y \geq 1$, $|A_2|_y \geq 2$, $|A_2|_x = 1$, $|A_3|_x = 2$, $|A_3|_z \geq 1$, and $|\Delta_z|_z \geq 2$. Now Δ_z cannot contain 3 y's, for otherwise, considering the structure of Δ_x, it would not be possible to distribute x's and y's in a proper way. However, if A_3 contains 3 y's then, so as not to contradict the structure of Δ_x (no proper distribution of y's and z's would exist), Δ_x must contain 3 z's, which contradicts to the structure of $X = \Delta_n = \Omega_{n,1}\Omega_{n,2}\Omega'_{n,3}$ (no proper distribution of y's and z's would exist among the blocks $\Omega_{n,1}$, $\Omega_{n,2}$, and $\Omega'_{n,3}$, each of which is supposed to contain exactly 2 occurrences of y and 2 occurrences of z). Thus, there are no y's in Δ_x, contradicting the structure of Δ_n (no proper distribution of y's and x's would exist among the blocks $\Omega_{n,1}$,$\Omega_{n,2}$, and $\Omega'_{n,3}$). □

Lemma 5. *For a crucial abelian cube-free word X over $\mathcal{A}_n$,*

$$(a_0, a_1, a_2, a_3, a_4) \neq (2, 3, 6, 9, 9).$$

Proof. Suppose $|X|_n = 2$ and assume that $|X|_t = 3$ for some other letter t. If $t \neq n-1$, then all three occurrences of t are in Δ_{n-1}, whereas the two occurrences of n are in Y_n (recall that $X = \Delta_n = Y_n\Delta_{n-1}$). This contradicts the fact that $|\Omega_{n,1}|_n = |\Omega_{n,1}|_t = 1$. Thus, $t = n - 1$ and $|X|_{n-1} = 3$.

Now, assuming x, y, and z are three letters, with $x < y < z < n - 1$, occurring $\{6, 9, 9\}$ times in X (we do not specify which letter occurs how many times). Then, as in the proof of Lemma 3, we deduce that Δ_z belongs entirely to the block $\Omega'_{n,3}$. Moreover, the block $\Omega'_{n,3}$ has $\{2, 3, 3\}$ occurrences of letters x, y, z (in some order). However, if x or y occur twice in $\Omega'_{n,3}$, they occur twice in Δ_z, contradicting the structure of Δ_z. Thus z must occur twice in $\Omega'_{n,3}$, and the letters x and y each occur 3 times in $\Omega'_{n,3}$. But then it is clear that x and y must each occur 3 times in Δ_z, contradicting the fact that x and z should be distributed properly in Δ_z, by Lemma 2. □

Theorem 3. *For $n \geq 5$, we have $\ell_3(n) \geq 9n - 13$.*

Proof. This is a direct consequence of the preceding four lemmas, which tell us that any attempt to decrease numbers in the sequence $(5, 3, 6, 9, 9, \ldots, 9)$ corresponding to E_n will lead to a prohibited configuration. □

Corollary 1. *For $n \geq 5$, we have $\ell_3(n) = 9n - 13$.*

Proof. The result follows immediately from Theorems 2 and 3. □

Remark 2. Recall from Remark 1 that $\ell_3(n) = 2, 5, 11, 20$ for $n = 1, 2, 3, 4$, respectively. For instance, the word 42131214231211321211 is a minimal crucial abelian cube-free word of length 20 ($= 2 + 3 + 6 + 9$). This can be proved using similar arguments as in the proofs of the Lemmas 2–5.

3 Crucial Words for Abelian k-th Powers

3.1 An Upper Bound for $\ell_k(n)$ and a Conjecture

The following theorem is a direct generalization of Theorem 1 and is a natural approach to obtaining an upper bound that improves $k^n - 1$ given by the k-generalized Zimin word $Z_{n,k}$.

Theorem 4. *For $k \geq 3$, we have $\ell_k(n) \leq k \cdot (k-1)^{n-1} - 1$.*

Proof. We proceed as in the proof of Theorem 1, with the only difference being that we begin with $X_1 = 1^{k-1}$ and put $(k-2)$ 1's to the right of each letter except for the last $(k-2)$ letters, after which we put $(k-1)$ 1's instead. □

We now proceed directly to the construction of a crucial abelian k-power-free word $D_{n,k}$ that we believe to be optimal.

A construction of a crucial abelian k-power-free word $D_{n,k}$, where $n \geq 4$ and $k \geq 2$. As we shall see, the following construction of the word $D_{n,k}$ is optimal for $k = 2, 3$. We believe that it is also optimal for any $k \geq 4$ and sufficiently large n (see Conjecture 1).

As our basis for the construction of the word $D_{n,k}$, we use the following word D_n, which is constructed as follows, for $n = 4, 5, 6, 7$. (As previously, we use spaces to separate the blocks $\Omega_{n,1}$ and $\Omega'_{n,2}$ in $D_n = \Delta_n$.)

$$D_4 = 34231\ 3231$$

$$D_5 = 4534231\ 432341$$

$$D_6 = 564534231\ 54323451$$

$$D_7 = 67564534231\ 6543234561$$

In general, the first block $\Omega_{n,1}$ in $D_n = \Delta_n = \Omega_{n,1}\Omega'_{n,2}$ is built by adjoining the factors $i(i+1)$ for $i = n-1, n-2, \ldots, 2$, followed by the letter 1. The second block $\Omega'_{n,2}$ is built by adjoining the factors $(n-1)(n-2)\cdots 432$, then $34\cdots(n-2)(n-1)$, and finally the letter 1.

Remark 3. The above construction coincides with the construction given in [6, Theorem 5] for a minimal crucial abelian square-free word over $\mathcal{A}_n$ of length $4n-7$. In fact, the word D_n can be obtained from the minimal crucial abelian cube-free word E_n (defined in Sec. 2.1) by removing the second block in E_n and deleting the rightmost copy of each letter except 2 in the first and third blocks of E_n.

Now we illustrate each step of the construction for the word $D_{n,k}$ using $D_{4,3}$ as an example. The construction can be explained directly, but we introduce it recursively, obtaining $D_{n,k}$ from $D_{n,k-1}$ for $n \geq 4$, and using the crucial abelian square-free word $D_{n,2} := D_n$ as the basis. For $n = 4$, we have

$$D_{4,2} = \Omega_{4,1}\Omega'_{4,2} = 34231\ 3231.$$

Assume that $D_{n,k-1} = \Omega_{n,1}\Omega_{n,2}\cdots\Omega'_{n,k-1}$ is constructed and implement the following steps to obtain $D_{n,k}$:

1. Duplicate $\Omega_{n,1}$ in $D_{n,k-1}$ to obtain the word

$$D'_{n,k-1} = \Omega_{n,1}\Omega_{n,1}\Omega_{n,2}\cdots\Omega'_{n,k-1}.$$

   For $n = 4$ and $k = 3$, $D'_{4,2} = 34231\ 34231\ 3231$.

2. Append to the second $\Omega_{n,1}$ in $D'_{n,k-1}$ the factor $134 \cdots n$ (in our example, 134; in fact, any permutation of $\{1, 3, 4, \ldots, n\}$ would work at this place) to obtain $\Omega_{n,2}$ in $D_{n,k}$. In each of the remaining blocks $\Omega_{n,i}$ in $D'_{n,k-1}$, duplicate the rightmost occurrence of each letter x, where $1 \leq x \leq n-1$ and $x \neq 2$. Finally, in the last block of $D'_{n,k}$ insert the letter n immediately before the leftmost 1 to obtain the word $D_{n,k}$. For $n = 4$ and $k = 3$, we have

$$D_{4,3} = \mathbf{34}4\mathbf{23}3\mathbf{1}1\ \mathbf{34231}134\ \mathbf{323}34\mathbf{1}1,$$

where the bold letters form the word $D'_{4,2}$ from which $D_{4,3}$ is derived.

We provide five more examples here, namely $D_{5,3}$, $D_{5,4}$, $D_{4,4}$, $D_{4,5}$, and $D_{6,4}$, respectively, so that the reader can check their understanding of the construction:

$$45534423311\ 45342311345\ 4323344511;$$

$$455534442333111\ 455344233111345\ 453423111334455\ 43233344455111;$$

$$34442333111\ 34423311134\ 34231113344\ 3233344111;$$

$$34444233331111\ 34442333111134\ 34423311113344\ 34231111333444\ 3233334441111;$$

$$5666455534442333111\ 5664553442331113456\ 5645342311133445566\ 543233344455566111.$$

Remark 4. By construction, $D_{n,3} = E_n$ for all $n \geq 4$.

Theorem 5. *For $n \geq 4$ and $k \geq 2$, we have $\ell_k(n) \leq k^2(n-1) - k - 1$.*

Proof. Fix $n \geq 4$ and $k \geq 2$. We have

$$D_{n,k} = \Omega_{n,1}\Omega_{n,2} \cdots \Omega_{n,k-1}\Omega'_{n,k}$$

where $\Omega_{n,k} = \Omega'_{n,k}n$, and by construction each $\Omega_{n,i}$ contains $(k-1)$ occurrences of the letter 1, one occurrence of the letter 2, $(k-1)$ occurrences of the letter n, and k occurrences of the letter x for $x = 3, 4, \ldots, n-1$. That is, for each $i = 1, 2, \ldots, k$, the Parikh vector of the block $\Omega_{n,i}$ is given by

$$\mathcal{P}(\Omega_{n,i}) = (k-1, 1, k, k, \ldots, k, k-1), \tag{1}$$

and hence $\mathcal{P}(D_{n,k}) = (k(k-1), k, k^2, k^2, \ldots, k^2, k(k-1) - 1)$. Thus,

$$|D_{n,k}| = k(k-1) + k + k^2(n-3) + k(k-1) - 1 = k^2(n-1) - k - 1.$$

We will now prove that $D_{n,k}$ is crucial with respect to abelian k-th powers; whence the result. The following facts, which are easily verified from the construction of $D_{n,k}$, will be useful in the proof.

Fact 1. In every block $\Omega_{n,i}$, the letter 3 has occurrences before and after the single occurrence of the letter 2.

Fact 2. In every block $\Omega_{n,i}$, all $(k-1)$ of the 1's occur after the single occurrence of the letter 2 (as the factor 1^{k-1}).

Fact 3. For all i with $2 \leq i \leq k-1$, the block $\Omega_{n,i}$ ends with n^{i-1} and the other $(k-1-i+1)$ n's occur (together as a string) before the single occurrence of the letter 2 in $\Omega_{n,i}$. In particular, there are exactly $k-2$ occurrences of the letter n between successive 2's in $D_{n,k}$.

Freeness: First we prove that $D_{n,k}$ is abelian k-power-free. Obviously, by construction, $D_{n,k}$ is not an abelian k-th power (as the number of occurrences of the letter n is not a multiple of k) and $D_{n,k}$ does not contain any *trivial* k-th powers, i.e., k-th powers of the form x^k for some letter x. Moreover, each block $\Omega_{n,i}$ is abelian k-power-free. For if not, then according to the Parikh vector of $\Omega_{n,i}$ (see (1)), at least one of the $\Omega_{n,i}$ must contain an abelian k-th power consisting of exactly k occurrences of the letter x for all $x = 3, 4, \ldots, n-1$, and no occurrences of the letters 1, 2, and n. But, by construction, this is impossible because, for instance, the letter 3 has occurrences before and after the letter 2 in each of the blocks $\Omega_{n,i}$ in $D_{n,k}$ (by Fact 1).

Now suppose, by way of contradiction, that $D_{n,k}$ contains a non-trivial abelian k-th power, say P. Then it follows from the preceding paragraph that P overlaps at least two of the blocks $\Omega_{n,i}$ in $D_{n,k}$. We first show that P cannot overlap three or more of the blocks in $D_{n,k}$. For if so, then P must contain at least one of the blocks, and hence P must also contain all k of the 2's. Furthermore, all of the 1's in each block occur after the letter 2 (by Fact 2), so there are $(k-1)^2 = k^2-2k+1$ occurrences of the letter 1 between the leftmost and rightmost 2's in $D_{n,k}$. Thus, P must contain all $k(k-1) = k^2 - k$ of the 1's. Hence, since $\Omega'_{n,k}$ ends with 1^{k-1}, we deduce that P must end with the word

$$W = 23^{k-1}1^{k-1}\Omega_{n,2}\cdots\Omega_{n,k-1}\Omega'_{n,k},$$

where $|W|_n = k$, $|W|_3 = k(k-1) + (k-1) = k^2 - 1$, and $|W|_x = k(k-1)$ for $x = 4, \ldots, n-1$. It follows that P must contain all k^2 of the 3's. But then, since

$$D_{n,k} = (n-1)n^{k-1}\cdots 34^{k-1}W$$

(by construction), we deduce that P must contain all k^2 of the 4's that occur in $D_{n,k}$, and hence all k^2 of the 5's, and so on. That is, P must contain all k^2 occurrences of the letter x for $x = 3, \ldots, n-1$; whence, since $D_{n,k}$ begins with the letter $n-1$, we have $P = \Omega_{n,1}\Omega_{n,2}\cdots\Omega_{n,k} = D_{n,k}$, a contradiction.

Thus, P overlaps exactly two adjacent blocks in $D_{n,k}$, in which case P cannot contain the letter 2; otherwise P would contain all k of the 2's, and hence would overlap all of the blocks in $D_{n,k}$, which is impossible (by the preceding arguments). Hence, P lies strictly between two successive occurrences of the letter 2 in $D_{n,k}$. But then P cannot contain the letter n as there are exactly $k-2$ occurrences of the letter n between successive 2's in $D_{n,k}$ (by Fact 3). Therefore, since the blocks $\Omega_{n,i}$ with $2 \leq i \leq k-1$ end with the letter n, it follows that P overlaps the blocks $\Omega_{n,1}$ and $\Omega_{n,2}$. Now, by construction, $\Omega_{n,1}$ ends with 1^{k-1}, and hence P contains k of the $2(k-1) = 2k-2$ occurrences of the letter 1 in $\Omega_{n,1}\Omega_{n,2}$. But then P must contain the letter 2 because $\Omega_{n,1}$ contains exactly $(k-1)$ occurrences of the letter 1 (as a suffix) and all $(k-1)$ of the 1's in $\Omega_{n,2}$ occur after the letter 2 (by Fact 2); a contradiction.

We have now shown that $D_{n,k}$ is abelian k-power-free. It remains to show that $D_{n,k}x$ ends with an abelian k-th power for each letter $x = 1, 2, \ldots, n$.

Cruciality: By construction, $D_{n,k}n$ is clearly an abelian k-th power. It is also easy to see that $D_{n,k}1$ ends with the (abelian) k-th power $\Delta_1 1 := 1^k$. Furthermore, for all $m = n, n-1, \ldots, 4$, we deduce from the construction that

$$\begin{aligned}
\Omega_{m,1} &= (m-1)m^{k-1}\Omega_{m-1,1},\\
\Omega_{m,2} &= (m-1)m^{k-2}\Omega_{m-1,2}m,\\
&\vdots\\
\Omega_{m,k-2} &= (m-1)m^2\Omega_{m-1,k-2}m^{k-3},\\
\Omega_{m,k-1} &= m(m-1)\Omega_{m-1,k-1}m^{k-2},\\
\Omega'_{m,k} &= (m-1)\Omega'_{m-1,k}[1^{k-1}]^{-1}(m-1)m^{k-2}1^{k-1},
\end{aligned}$$

where $\Omega'_{m-1,k}[1^{k-1}]^{-1}$ indicates the deletion of the suffix 1^{k-1} of $\Omega'_{m-1,k}$.

Consequently, for $x = n-1, n-2, \ldots, 3$, the word $D_{n,k}x$ ends with the abelian k-th power $\Delta_x x$ where Δ_x is such that

$$\Delta_{x+1} = x(x+1)^{k-1}\Delta_x \quad \text{with } \Delta_n := D_{n,k}.$$

□

Observe that $|D_{n,2}| = 4n - 7$ and $|D_{n,3}| = 9n - 13$. Hence, since $D_{n,k}$ is a crucial abelian k-power-free word (by the proof of Theorem 5), it follows from [6, Theorem 5] and Corollary 1 that the words $D_{n,2}$ and $D_{n,3}$ are minimal crucial words over $\mathcal{A}_n$ avoiding abelian squares and abelian cubes, respectively. That is, for $k = 2, 3$, the word $D_{n,k}$ gives the length of a minimal crucial word over $\mathcal{A}_n$ avoiding abelian k-th powers. In the case of $k \geq 4$, we make the following conjecture.

Conjecture 1. For $k \geq 4$ and sufficiently large n, the length of a minimal crucial word over $\mathcal{A}_n$ avoiding abelian k-th powers is given by $k^2(n-1) - k - 1$.

3.2 A Lower Bound for $\ell_k(n)$

A trivial lower bound for $\ell_k(n)$ is $nk-1$ as all letters except n must occur at least k times, whereas n must occur at least $k-1$ times. We give here the following slight improvement of the trivial lower bound, which must be rather imprecise though.

Theorem 6. *For $n \geq 5$ and $k \geq 4$, we have $\ell_k(n) \geq k(3n-4) - 1$.*

Proof. Assuming that X is a crucial word over the n-letter alphabet $\mathcal{A}_n$ with respect to abelian k-th powers ($k \geq 4$), we see that adjoining any letter from $\mathcal{A}_n$ to the right of X must create a cube as a factor from the right. In particular, adjoining n from the right side leads to creating a cube of length at least $9n-13$ (by Lemmas 2–5). This cube will be $\Omega_{n,k-2}\Omega_{n,k-1}\Omega'_{n,k}$ in X and thus $\Omega_{n,i}$, for $1 \leq i \leq k-1$, will have length at least $3n-4$, whereas $\Omega'_{n,k}$ has length at least $3n-5$, which yields the result. □

4 Further Research

1. Prove or disprove Conjecture [1]. Notice that the general construction uses a greedy algorithm for going from $k-1$ to k, which does not work for going from $n-1$ to n for a fixed k. However, we believe that the conjecture is true.

2. A word W over $\mathcal{A}_n$ is *maximal* with respect to a given set of prohibitions if W avoids the prohibitions, but xW and Wx do not avoid the prohibitions for any letter $x \in \mathcal{A}_n$. For example, the word 323121 is a maximal abelian square-free word over $\{1,2,3\}$ of minimal length. Clearly, the length of a minimal crucial word with respect to a given set of prohibitions is at most the length of a shortest maximal word. Thus, obtaining the length of a minimal crucial word we get a lower bound for the length of a shortest maximal word.

   Can we use our approach to tackle the problem of finding maximal words of minimal length? In particular, Korn [8] proved that the length $\ell(n)$ of a shortest maximal abelian square-free word over $\mathcal{A}_n$ satisfies $4n-7 \leq \ell(n) \leq 6n-10$ for $n \geq 6$, while Bullock [1] refined Korn's methods to show that $6n-29 \leq \ell(n) \leq 6n-12$ for $n \geq 8$. Can our approach improve Bullock's result (probably too much to ask when taking into account how small the gap is), or can it provide an alternative solution?

References

1. Bullock, E.M.: Improved bounds on the length of maximal abelian square-free words. Electron. J. Combin. 11, #R17 (2004)
2. Carpi, A.: On the number of abelian square-free words on four letters. Discrete Appl. Math. 81, 155–167 (1998)
3. Cummings, L.J., Mays, M.: A one-sided Zimin construction. Electron. J. Combin. 8, #R27 (2001)
4. Erdős, P.: Some unsolved problems. Magyar Tud. Akad. Mat. Kutató Int. Közl. 6, 221–254 (1961)
5. Evdokimov, A.A.: Strongly asymmetric sequences generated by a finite number of symbols. Dokl. Akad. Nauk SSSR 179, 1268–1271 (1968); Soviet Math. Dokl. 9, 536–539 (1968)
6. Evdokimov, A.A., Kitaev, S.: Crucial words and the complexity of some extremal problems for sets of prohibited words. J. Combin. Theory Ser. A 105, 273–289 (2004)
7. Keränen, V.: Abelian squares are avoidable on 4 letters. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 41–52. Springer, Heidelberg (1992)
8. Korn, M.: Maximal abelian square-free words of short length. J. Combin. Theory Ser. A 102, 207–211 (2003)
9. Pleasants, P.A.B.: Non-repetitive sequences. Proc. Cambridge Philos. Soc. 68, 267–274 (1970)
10. Zimin, A.: Blocking sets of terms. Mat. Sb (N.S.) 119 (161), 363–375, 447 (1982); Math. USSR Sbornik 47 353–364 (1984)

Tight Bounds on the Descriptional Complexity of Regular Expressions

Hermann Gruber and Markus Holzer

Institut für Informatik, Universität Giessen,
Arndtstr. 2, D-35392 Giessen, Germany
{hermann.gruber,holzer}@informatik.uni-giessen.de

Abstract. We improve on some recent results on lower bounds for conversion problems for regular expressions. In particular we consider the conversion of planar deterministic finite automata to regular expressions, study the effect of the complementation operation on the descriptional complexity of regular expressions, and the conversion of regular expressions extended by adding intersection or interleaving to ordinary regular expressions. Almost all obtained lower bounds are optimal, and the presented examples are over a binary alphabet, which is best possible.

1 Introduction

It is well known that regular expressions are equally expressive as finite automata. In contrast to this equivalence, a classical result due to Ehrenfeucht and Zeiger states that finite automata, even deterministic ones, can sometimes allow exponentially more succinct representations than regular expressions [4]. Although they obtained a tight lower bound on expression size, their examples used an alphabet of growing size.

Reducing the alphabet size remained an open challenge [5] until the recent advent of new proof techniques, see [8,9,12]—indeed most of our proofs in this paper rely on the recently established relation between regular expression size and star height of regular languages [9]. Although this resulted in quite a few new insights into the nature of regular expressions, see also [7,10,11], proving tight lower bounds for small alphabets remains a challenging task, and not all bounds in the mentioned references are both tight and cover all alphabet sizes. In this work, we close some of the remaining gaps: in the case of converting planar finite automata to regular expressions, we prove the bound directly, by finding a witness language over a binary alphabet. For the other questions under consideration, namely the effect of complementation and of extending regular expression syntax by adding an intersection or interleaving operator, proceeding in this way appears more difficult. Yet, sometimes it proves easier to find witness languages over larger alphabets. For this case, we also devise a new set of encodings which are economic and, in some precise sense, robust with respect to both the Kleene star and the interleaving operation. This extends the scope of known proof techniques, and allows us to give a definitive answer to some questions regarding the descriptional complexity of regular expressions that were not

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 276–287, 2009.

Table 1. Comparing the lower bound results for conversion problems of deterministic finite automata (DFA), regular expressions (RE), and regular expressions with additional operations (RE($\cdot$)), where $\cap$ denotes intersection, $\neg$ complementation, and ш the interleaving or shuffle operation on formal languages. Entries with a bound in $\Theta(\cdot)$ indicate that the result is best possible, i.e., refers to a lower bound matching a known upper bound.

Conversion	known results		this paper with $\|\Sigma\| = 2$	
planar DFA to RE	$2^{\Theta(\sqrt{n})}$	for $\|\Sigma\| = 4$ [9]	$2^{\Theta(\sqrt{n})}$	[Thm. 3]

$\neg$ RE to RE	$2^{2^{\Omega(\sqrt{n \log n})}}$
	$2^{2^{\Omega(n)}}$
		for $\|\Sigma\| = 2$ [9]
		for $\|\Sigma\| = 4$ [8]
			$2^{2^{\Theta(n)}}$	[Thm. 6]

RE($\cap$) to RE	$2^{2^{\Omega(\sqrt{n})}}$	for $\|\Sigma\| = 2$ [7]	$2^{2^{\Theta(n)}}$	[Thm. 7]
RE(ш) to RE	$2^{2^{\Omega(\sqrt{n})}}$	for $\|\Sigma\|$ const. [7]	$2^{2^{\Omega(n/\log n)}}$ $2^{2^{\Theta(n)}}$ for $\|\Sigma\| = O(n)$	[Thm. 14] [Thm. 8]

yet settled completely in previous work [5,7,8,9]; also note that these problems become easy in the case of unary alphabets [5]. Our main results are summarized and compared to known results in Table 1.

2 Basic Definitions

We introduce some basic notions in formal language and automata theory—for a thorough treatment, the reader might want to consult a textbook such as [15]. In particular, let Σ be a finite alphabet and Σ^* the set of all words over the alphabet Σ, including the empty word ε. The length of a word w is denoted by $|w|$, where $|\varepsilon| = 0$. A *(formal) language* over the alphabet Σ is a subset of Σ^*.

The *regular expressions* over an alphabet Σ are defined recursively in the usual way:[1] $\emptyset$, ε, and every letter a with $a \in \Sigma$ is a regular expression; and when r_1 and r_2 are regular expressions, then $(r_1 + r_2)$, $(r_1 \cdot r_2)$, and $(r_1)^*$ are also regular expressions. The language defined by a regular expression r, denoted by $L(r)$, is defined as follows: $L(\emptyset) = \emptyset$, $L(\varepsilon) = \{\varepsilon\}$, $L(a) = \{a\}$, $L(r_1 + r_2) = L(r_1) \cup L(r_2)$, $L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$, and $L(r_1^*) = L(r_1)^*$. The *size* or *alphabetic width* of a regular expression r over the alphabet Σ, denoted by $\mathrm{alph}(r)$, is defined as the total number of occurrences of letters of Σ in r. For a regular language L, we define its alphabetic width, $\mathrm{alph}(L)$, as the minimum alphabetic width among all regular expressions describing L.

Our arguments on lower bounds for the alphabetic width of regular languages is based on a recent result that utilizes the star height of regular

[1] For convenience, parentheses in regular expressions are sometimes omitted and the concatenation is simply written as juxtaposition. The priority of operators is specified in the usual fashion: concatenation is performed before union, and star before both product and union.

languages [9]. Here the star height of a regular language is defined as follows: for a regular expression r over Σ, the star height, denoted by $h(r)$, is a structural complexity measure inductively defined by: $h(\emptyset) = h(\varepsilon) = h(a) = 0$, $h(r_1 \cdot r_2) = h(r_1 + r_2) = \max(h(r_1), h(r_2))$, and $h(r_1^*) = 1 + h(r_1)$. The star height of a regular language L, denoted by $h(L)$, is then defined as the minimum star height among all regular expressions describing L. The next theorem establishes the aforementioned relation between alphabetic width and star height of regular languages [9]:

Theorem 1. *Let $L \subseteq \Sigma^*$ be a regular language. Then* $\mathrm{alph}(L) \geq 2^{\frac{1}{3}(h(L)-1)} - 1$.

The star height of a regular language appears to be more difficult to determine than its alphabetic width, see, e.g., [13]. Fortunately, the star height can be determined more easily for bideterministic regular languages: A DFA is *bideterministic*, if it has a single final state, and if the NFA obtained by reversing all transitions and exchanging the roles of initial and final state is again deterministic—notice that, by construction, this NFA in any case accepts the reversed language. A regular language L is *bideterministic* if there exists a bideterministic finite automaton accepting L. For these languages, the star height can be determined from the digraph structure of the minimal DFA: the *cycle rank* of a digraph $G = (V, E)$, denoted by $cr(G)$, is inductively defined as follows: (1) If G is acyclic, then $cr(G) = 0$. (2) If G is strongly connected, then $cr(G) = 1 + \min_{v \in V}\{cr(G - v)\}$, where $G - v$ denotes the graph with the vertex set $V \setminus \{v\}$ and appropriately defined edge set. (3) If G is not strongly connected, then $cr(G)$ equals the maximum cycle rank among all strongly connected components of G. For a given finite automaton A, let its cycle rank, denoted by $cr(A)$, be defined as the cycle rank of the underlying digraph. Eggan's Theorem states that the star height of a regular language equals the minimum cycle rank among all NFAs accepting it [3]. Later, McNaughton [18] proved the following:

Theorem 2 (McNaughton's Theorem). *Let L be a bideterministic language, and let A be the minimal trim, i.e., without a dead state, deterministic finite automaton accepting L. Then $h(L) = cr(A)$.*

In fact, the minimality requirement in the above theorem is not needed, since every bideterministic finite automaton in which all states are useful is already a trim minimal deterministic finite automaton. Here, a state is useful if it is both reachable from the start state, and if some final state is reachable from it.

3 Lower Bounds on Regular Expression Size

This section consists of three parts. First we show an optimal bound converting planar deterministic finite automata to equivalent regular expressions and then we present our results on the alphabetic width on complementing regular expression and on regular expressions with intersection and interleaving. While the former result utilizes a characterization of cycle rank in terms of a cops and robbers game given in [9], the latter two results are mainly based on star-height-preserving morphisms.

3.1 Converting Planar DFAs into Regular Expressions

Recently, it was shown that for planar finite automata, one can construct equivalent regular expressions of size at most $2^{O(\sqrt{n})}$, for all alphabet sizes polynomial in n [5]. This is a notable improvement over the general case, since conversion from n-state deterministic finite automata to equivalent regular expressions was shown to be of order $2^{\Theta(n)}$ in [9]. Also in [9], for alphabet size at least four a lower bound on the conversion of planar deterministic finite automata to regular expressions of $2^{\Theta(\sqrt{n})}$ was proven. We improve this result to alphabets of size two, using a characterization of cycle rank in terms of a cops and robber game from [9].

Theorem 3. *There is an infinite family of languages L_n over a binary alphabet acceptable by n-state planar deterministic finite automata, such that* $\mathrm{alph}(L_n) = 2^{\Omega(\sqrt{n})}$.

Proof. By Theorems 1 and 2, it suffices to find an infinite family of bideterministic finite automata A_k of size $O(k^2)$ such that the digraph underlying A_k has cycle rank $\Omega(k)$. The deterministic finite automata A_k witnessing the claimed lower bound are inspired by a family of digraphs G_k defined in [16]. These graphs each admit a planar drawing as the union of k concentric equally directed $2k$-cycles, which are connected to each other by $2k$ radial directed k-paths, the first k of which are directed inwards, while the remaining k are directed outwards; see Figure 1 for illustration. Formally, for $k \geq 1$, let $G_k = (V, E)$ be the graph with vertex set $V = \{\, u_{i,j} \mid 1 \leq i, j \leq k \,\} \cup \{\, v_{i,j} \mid 1 \leq i, j \leq k \,\}$, and whose edge set can be partitioned into a set of directed $2k$-cycles C_i, and two sets of directed k-paths P_i and Q_i with $1 \leq i \leq k$. Here each C_i admits a walk visiting the vertices $u_{i,1}, u_{i,2}, \ldots, u_{i,k}, v_{i,1}, v_{i,2}, \ldots, v_{i,k}$ in order, each P_i admits a walk visiting the vertices $u_{1,i}, u_{2,i}, \ldots, u_{k,i}$ in order, and Q_i admits a walk visiting the vertices $v_{k,i}, v_{k-1,i}, \ldots, v_{k,1}$ in order.

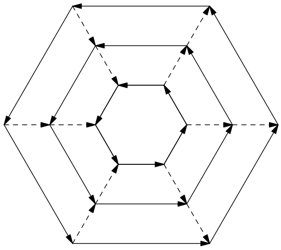


Fig. 1. A drawing of the graph G_3. When viewed as automaton A_3, the solid (dashed, respectively) arrows indicate a-transitions (b-transitions, respectively).

Fix $\{a, b\}$ as a binary input alphabet. If we interpret the edges in G_k belonging to the cycles C_i as a-transitions, the edges belonging to the paths P_i and Q_i as b-transitions, interpret the vertices as states and choose a single initial and a single final state (both arbitrarily), we obtain a finite automaton A_k with $O(k^2)$ states whose underlying digraph is G_k. It is easily observed that A_k is bideterministic; thus it only remains to show that for the underlying digraph G_k the identity $cr(G_k) = \Omega(k)$ holds.

To this end, we use the cops and robber game characterization of graphs having cycle rank k given in [9]. A game quite similar to the mentioned one is studied in [16]; there a lower bound of k on the number of required cops on G_k is proved. It is not hard to prove that the lower bound carries over and $cr(G_k)$ is at least $k-1$. □

3.2 Operations on Regular Expressions: Alphabetic Width of Complementation

As noted in [5], the naive approach to complement regular expressions, of first converting the given expression into a nondeterministic finite automaton, determinizing, complementing the resulting deterministic finite automaton, and converting back to a regular expression gives a doubly exponential upper bound of $2^{2^{O(n)}}$. The authors of [5] also gave a lower bound of $2^{\Omega(n)}$, and stated as an open problem to find tight bounds. A doubly-exponential lower bound was found in [8], for alphabets of size at least four. Their witness language is a 4-symbol encoding of the set of walks in an n-vertex complete digraph. They gave a very short regular expression describing the complement of the encoded set, and provided a direct and technical proof showing that the encoded language requires large regular expressions, carefully adapting the approach originally taken by Ehrenfeucht and Zeiger [4]. Resulting from an independent approach pursued by the authors, in [9] a roughly doubly-exponential lower bound of $2^{2^{O(\sqrt{n \log n})}}$ was given for the binary alphabet.

Now it appears tempting to encode the language from [8] using a star-height-preserving morphism to further reduce the alphabet size, as done in [9] for a similar problem. Unfortunately, the proof from [8] does not offer any clue about the star height of the witness language, and thus we cannot mix these proof techniques. At least, it is known [2] that the *preimage* of the encoded language has large star height:

Theorem 4 (Cohen). *Let J_n be the complete digraph on n vertices with self-loops, where each edge (i,j) carries a unique label a_{ij}. Let W_n denote the set of all walks $a_{i_0 i_1} a_{i_1 i_2} \cdots a_{i_{r-2} i_{r-1}} a_{i_{r-1} i_r}$ in J_n, including the empty walk ε. Then the star height of language W_n equals n.*

To obtain a tight lower bound for binary alphabets, here we use a similar encoding as in [8], but make sure that the encoding is a star-height-preserving morphism. Here a morphism ρ *preserves star height*, if the star height of each regular language L equals the star height of the homomorphic image $\rho(L)$. The existence of such encodings was already conjectured in [3]. A full characterization of star-height-preserving morphisms was established later in [14], which reads as follows:

Theorem 5 (Hashiguchi/Honda). *A morphism $\rho : \Gamma^* \to \Sigma^*$ preserves star height if and only if (1) ρ is injective, (2) ρ is both prefix-free and suffix-free, that is, no word in $\rho(\Gamma)$ is prefix or suffix of another word in $\rho(\Gamma)$, and (3) ρ*

has the non-crossing property, that is, for all $v, w \in \rho(\Gamma)$ holds: If v can be decomposed as $v = v_1 v_2$, with $v_1, v_2 \neq \varepsilon$, and w as $w = w_1 w_2$, with $w_1, w_2 \neq \varepsilon$, such that both cross-wise concatenations $v_1 w_2$ and $w_1 v_2$ are again in $\rho(\Gamma)$, then this implies $v_1 = w_1$ or $v_2 = w_2$.

Observe that the given lower bound matches the aforementioned upper bound on the problem under consideration.

Theorem 6. *There exists an infinite family of languages L_n over a binary alphabet Σ with* $\mathrm{alph}(L_n) = O(n)$, *such that* $\mathrm{alph}(\Sigma^* \setminus L_n) = 2^{2^{\Omega(n)}}$.

Proof. We will first prove the theorem for alphabet size 3, and then use a star-height-preserving morphism to further reduce the alphabet size to binary. Let W_{2^n} be the set of walks in a complete 2^n-vertex digraph as defined in Theorem 4. Let $E = \{\, a_{ij} \mid 0 \leq i, j \leq 2^n - 1 \,\}$ denote the edge set of this graph, and let $\Sigma = \{0, 1, \$\}$.

Now define the morphism $\rho : E^* \to \Sigma^*$ by $\rho(a_{ij}) = \mathrm{bin}(i) \cdot \mathrm{bin}(j) \cdot \mathrm{bin}(i) \cdot \mathrm{bin}(j)\$$, where $\mathrm{bin}(i)$ denotes the usual n-bit binary encoding of the number i. To see that ρ is star-height-preserving, one has to verify the properties of Theorem 5, which is an easy exercise. Our witness language for ternary alphabets is the complement of the set $L_n = \rho(W_{2^n})$. To establish the theorem for ternary alphabets, we give a regular expression of size $O(n)$ describing the complement of L_n; a lower bound of $2^{2^{\Omega(n)}}$ then immediately follows from Theorems 1 and 4 since the morphism ρ preserves star height. As for the witness language given in [8], our expression is a union of some local consistency tests: Every nonempty word in L_n falls apart into blocks of binary digits of each of length $4n$, separated by occurrences of the symbol \$, and takes the form

$$(\mathrm{bin}(i_0)\,\mathrm{bin}(i_1))^2\$\,(\mathrm{bin}(i_1)\,\mathrm{bin}(i_2))^2\,\$\cdots\$(\mathrm{bin}(i_{r-1})\,\mathrm{bin}(i_r))^2\$.$$

Thus, word w is not in L_n if and only if we have at least one of the following cases: (i) The word w has no prefix in $\{0,1\}^{4n}\$$, or w contains an occurrence of \$ not immediately followed by a word in $\{0,1\}^{4n}\$$; (ii) the region around the boundary of some pair of adjacent blocks in w is not of the form $\mathrm{bin}(i)\$\,\mathrm{bin}(i)$; or (iii) some block does not contain the pattern $(\mathrm{bin}(i)\,\mathrm{bin}(j))^2$, in the sense that inside the block some pair of bits at distance $2n$ does not match. It is not hard to encode these conditions into a regular expression of size $O(n)$.

To further decrease the alphabet size to binary, we use the star height-preserving morphism $\sigma = \{0 \mapsto a^1 b^3, 1 \mapsto a^2 b^2, \$ \mapsto a^3 b^1\}$, which already proved useful in [9]. Then $\sigma(L_n)$ has star height 2^n and thus again has alphabetic width at least $2^{2^{\Omega(n)}}$. For an upper bound on the alphabetic width of its complement, note first that every word w that is in $\sigma(\Sigma^*)$ but not in $\sigma(L_n)$ matches the morphic image under σ of the expression r_n given above; and $\sigma(r_n)$ still has alphabetic width $O(n)$. The words in the complement of $\sigma(L_n)$ not covered by the expression $\sigma(r_n)$ are precisely those not in $\sigma(\{0,1,\$\}^*)$, and the complement of the latter set can be described by a regular expression of constant size. The union of these two expressions gives a regular expression of size $O(n)$ as desired. □

3.3 Regular Expressions with Intersection and Interleaving

It is known that extending the syntax with an intersection operator can provide an exponential gain in succinctness over nondeterministic finite automata. For instance, in [6] it is shown that the set of palindromes of length n can be described by regular expressions with intersection of size $O(n)$. On the other hand, it is well known that the number of states of a nondeterministic finite automaton accepting P_n has $\Omega(2^n)$ states [19]. Of course, it appears more natural to compare the gain in succinctness of such extended regular expressions to ordinary regular expressions rather than to finite automata. There a $2^{2^{O(n)}}$ doubly exponential upper bound readily follows by combining standard constructions [7]. Yet a roughly doubly-exponential lower bound of $2^{2^{\Omega(\sqrt{n})}}$, for alphabets of growing size, was found only recently in [8], and a follow-up paper [7] shows that this can be reached already for binary alphabets. Here we finally establish a tight doubly-exponential lower bound, which even holds for binary alphabets.

Theorem 7. *There is an infinite family of languages L_n over a binary alphabet admitting regular expressions with intersection of size $O(n)$, such that* $\mathrm{alph}(L_n) = 2^{2^{\Omega(n)}}$.

Proof. First, we show that the set of walks $W_{2^n} \subseteq E^*$ defined in Theorem 4 allows a compact representation using regular expressions with intersection. First we define $M = \{\, a_{i,j} \cdot a_{j,k} \mid 0 \leq i, j, k \leq 2^n - 1 \,\}$ and then observe that the set *Even* of all nonempty walks of even length, i.e., total number of seen edges, in the graph J_n can be written as $\mathit{Even} = M^* \cap (E \cdot M^* \cdot E)$, while the the set *Odd* of all nonempty walks of odd length is $\mathit{Odd} = (E \cdot M^*) \cap (M^* \cdot E)$. Thus, we have $W_{2^n} = \mathit{Even} \cup \mathit{Odd} \cup \{\varepsilon\}$. This way of describing W_{2^n} appears to be a long shot from our goal; it uses a large alphabet and does not even reach a linear-exponential gain in succinctness over ordinary regular expressions—a similar statement appears, already over thirty years ago, in [4]. In order to get the desired result, we present a binary encoding τ that preserves star height and allows a representation of the encoded sets $\tau(M)$ and $\tau(E)$ by regular expressions with intersection each of size $O(n)$. Let $\tau : E^* \to \{0,1\}^*$ be the morphism defined by $\tau(a_{i,j}) = \mathrm{bin}(i) \cdot \mathrm{bin}(j) \cdot \mathrm{bin}(j)^R \cdot \mathrm{bin}(i)^R$, for $0 \leq i, j \leq 2^{n-1}$. To see that τ preserves star height, we have to check the properties given in Theorem 5, which is an easy exercise. Thus, by Theorems 1 and 4, the set $\tau(W_{2^n})$ has alphabetic width at least $2^{2^{\Omega(n)}}$.

It remains to give expressions with intersection of size $O(n)$ for the set $\tau(W_{2^n})$. Since $\tau(W_{2^n}) = \tau(\mathit{Even}) \cup \tau(\mathit{Odd}) \cup \{\varepsilon\}$, the morphism commutes with concatenation, union, and Kleene star, and, being injective, also with intersection, it suffices to give regular expressions with intersection for $\tau(E)$ and $\tau(M)$ of size $O(n)$. To this end, we we make use of an observation from [6], namely that the sets of palindromes of length $2m$ admit regular expressions with intersection of size $O(m)$. A straightforward extension of that idea gives a size $O(m+n)$ regular expression with intersection for $S_{m,n} = \{\, vwv^R \in \{0,1\}^* \mid |v| = m, |w| = n \,\}$, where m and n are fixed nonnegative integers.

Finally, observe that the set $\tau(E) = \{\, ww^R \in \{0,1\}^* \mid |w| = 2n \,\}$ is equal to $S_{n,0}$ and that the set $\tau(M)$ can be written as $\{\, uvv^R u^R vww^R v^R \mid |u| = |v| = |w| = n \,\}$, or $\{0,1\}^{2n} \cdot S_{n,n} \cdot \{0,1\}^{3n} \cap \tau(E) \cdot \tau(E)$. The latter set can be described by a regular expression with intersection of size $O(n)$ again, and the proof is complete. □

The interleaving of languages is another basic language operation known to preserve regularity. Regular expressions extended with interleaving were first studied in [17], with focus on the computational complexity of word problems. They also showed that regular expressions extended with an interleaving operator can be exponentially more succinct than nondeterministic finite automata [17]. Very recently, it was shown in [7] that regular expressions with interleaving can be roughly doubly-exponentially more succinct than regular expressions: converting such expressions into ordinary regular expressions can cause a blow-up in required expression size of $2^{2^{\Omega(\sqrt{n})}}$, for constant alphabet size. This bound is close to an easy upper bound of $2^{2^{O(n)}}$ that follows from standard constructions, see, e.g., [7] for details. If we take alphabets of growing size into account, the lower bound can be increased to match this trivial upper bound. The language witnessing that bound is in fact of very simple structure.

Theorem 8. *There is an infinite family of languages L_n over an alphabet of size $O(n)$ having regular expressions with interleaving of size $O(n)$, such that* $\mathrm{alph}(L_n) = 2^{2^{\Omega(n)}}$.

Proof. We consider the language L_n described by the shuffle regular expression

$$r_n = (a_1 b_1)^* \mathbin{ш} (a_2 b_2)^* \mathbin{ш} \cdots \mathbin{ш} (a_n b_n)^*$$

of size $O(n)$ over the alphabet $\Gamma = \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n\}$. To give a lower bound on the alphabetic width of L_n, we estimate first the star height of L_n. The language L_n can be accepted by a 2^n-state partial bideterministic finite automaton $A = (Q, \Sigma, \delta, q_0, F)$, whose underlying digraph forms a symmetric n-dimensional hypercube: The set of states is $Q = \{0,1\}^n$, the state $q_0 = 0^n$ is the initial state, and is also the only final state, i.e., $F = \{0^n\}$. For $1 \le i \le n$, the partial transition function δ is specified by $\delta(p, a_i) = q$ and $\delta(q, b_i) = p$, for all pairs of states (p, q) of the form $(x0y, x1y)$ with $x \in \{0,1\}^{i-1}$ and $y \in \{0,1\}^{n-i}$. It can be readily verified that this partial deterministic finite automaton is reduced and bideterministic. Therefore, the star height of L_n coincides with the cycle rank of the n-dimensional symmetric Cartesian hypercube. For a symmetric graph G, the cycle rank of G coincides with its (undirected) elimination tree height, which is in turn bounded below by the (undirected) pathwidth of G. Many structural properties of the n-dimensional hypercube are known, and among these is the recently established fact [1] that its pathwidth equals $\sum_{i=0}^{n-1} \binom{i}{\lceil i/2 \rceil} = \Theta(2^{n-1/2 \log n})$, where the latter estimate uses Stirling's approximation. Using Theorem 1, we obtain $\mathrm{alph}(L_n) = 2^{\Omega(2^{n-1/2 \log n})} = 2^{2^{\Omega(n)}}$, as desired. □

For a similar result using binary alphabets, we will encode the above witness language in binary using a star-height-preserving morphism. Some extra care has to be taken, however. The ideal situation one might hope for is to find for each $\Gamma = \{a_1, a_2, \ldots, a_n\}$ a suitable star-height-preserving morphism $\rho : \Gamma^* \to \{0,1\}^*$ such that $\rho(x \sqcup\!\sqcup\, y) = \rho(x) \sqcup\!\sqcup\, \rho(y)$, for all $x, y \in \Gamma^*$. This aim however appears to be a bit too ambitious. In all cases we have tried, the right-hand side of the above equation can contain words which are not even valid codewords. In [7] this difficulty is avoided altogether by simulating regular expressions with intersection by those with interleaving, using a trick from [17]. The drawback here is that the simulation takes place at the expense of introducing an extra symbol and polynomially increased size of the resulting expression with interleaving. To overcome this difficulty, Warmuth and Haussler devised a particular encoding [20], which they called *shuffle resistant*, that has the above property once we restrict our attention to codewords. Inspired by a property of this encoding proved later by Mayer and Stockmeyer [17, Prop. 3.1], we are led to define in general a shuffle resistant encoding as follows:

Definition 9. *An injective morphism $\rho : \Gamma^* \to \Sigma^*$, for some alphabets Γ and Σ, is* shuffle resistant *if $\rho(L(r)) = L(\rho(r)) \cap \rho(\Gamma)^*$, for each regular expression r with interleaving over Γ.*

The following is proved in [17, Prop. 3.1] for the encoding proposed by Warmuth and Haussler in [20]:

Theorem 10. *Let $\Gamma = \{a_1, a_2, \ldots, a_n\}$ and $\Sigma = \{a, b\}$. The morphism $\rho : \Gamma^* \to \Sigma^*$, which maps a_i to $a^{i+1}b^i$ is shuffle resistant.*

Incidentally, this encoding also preserves star height. The drawback is, however, that $\text{alph}(h(r)) = \Theta(|\Sigma| \,\text{alph}(r))$, for r a regular expression with interleaving. We now present a general family of more economic encodings, into alphabets of size at least 3, that enjoy similar properties.

Theorem 11. *Let Γ and Σ be two alphabets, and $\$$ be a symbol not in Σ. If $\rho : \Gamma^* \to (\Sigma \cup \{\$\})^*$ is an injective morphism with $\rho(\Gamma) \subseteq \Sigma^k\$$, for some integer k, then ρ is shuffle resistant.*

Proof. We need to show that for each such morphism ρ, the equality $\rho(L(r)) = L(\rho(r)) \cap \rho(\Gamma)^*$ holds for all regular expressions r with interleaving over Γ. The outline of the proof is roughly the same as the proof for Theorem 10 as sketched in [17]. The proof is by induction on the operator structure of r, using the stronger inductive hypothesis that

$$L(\rho(r)) \subseteq \rho(L(r)) \cup E, \quad \text{with} \quad E = (\rho(\Gamma))^* \Sigma^{\geq k+1} (\Sigma \cup \$)^* \tag{1}$$

Roughly speaking, the "error language" E specifies that the first error occurring in a word in $L(\rho(r))$ but not in $(\rho(\Gamma))^*$ must consist in a sequence of too many consecutive symbols from Σ.

The base cases are easily established, and also the induction step is easy for the regular operators concatenation, union, and Kleene star. The more difficult part is to show that if two expressions r_1 and r_2 satisfy Equation (1), then this also holds for $r = r_1 \sqcup\!\sqcup\, r_2$. To prove this implication, it suffices to show the following claim:

Claim 12. For all words u, v in $\rho(\Gamma)^* \cup E$ and for each word z in $u \sqcup\!\sqcup\, v$ the following holds: If both $z \in (\Sigma^k\$)^*$ and $u, v \in \rho(\Gamma)^*$, then $z \in \rho\left(\rho^{-1}(u) \sqcup\!\sqcup\, \rho^{-1}(v)\right)$. Otherwise, $z \in E$.

Proof. We prove the claim by induction on the length of z. The base case with $|z| = 0$ is clear. For the induction step, assume $|z| > 0$ and consider the prefix y consisting of the first $k + 1$ letters of z. Such a prefix always exists if z is obtained from shuffling two nonempty words from $\rho(\Gamma)^* \cup E$. The cases where u or v is empty are trivial. Observe first that it is impossible to obtain a prefix in $\Sigma^{<k}\$$ by shuffling two prefixes u' and v' of the words u and v. Also, a prefix in $\Sigma^{>k}$ always completes to a word $z \in E$. It remains to consider the case z has a prefix y in $\Sigma^k\$$. To obtain such a prefix, two prefixes u' and v' have to be shuffled, with $(u', v') \in (\Sigma^j) \times (\Sigma^{k-j}\$)$ or $(u', v') \in (\Sigma^j\$) \times (\Sigma^{k-j})$. But since these are prefixes of words in $\rho(\Gamma)^* \cup E$, the index j can take on only the values $j = 0$ and $j = k$. Thus, if $y \in \Sigma^k\$$, then y is indeed in $\rho(\Gamma)$, and y is obtained by observing exclusively the first $k+1$ letters of u, or exclusively the first $k+1$ letters of v. Hence at least one of the subcases $y^{-1}z \in (y^{-1}u) \sqcup\!\sqcup\, v$ and $y^{-1}z \in u \sqcup\!\sqcup\, (y^{-1}v)$ holds. We only consider the first subcase, for the second one a symmetric argument applies.

It is not hard to see that we can apply the induction hypothesis to this subcase: Because $y \in \rho(\Gamma)$ and $u \in \rho(\Gamma)^* \cup E$, the word $y^{-1}u$ is again in the set $\rho(\Gamma)^* \cup E$. Having furthermore $|y^{-1}z| < |z|$, the induction hypothesis readily implies that claimed statement also holds for the word $z = y(y^{-1}z)$. This completes the proof of the claim. □

Having established the claim, completing the proof of the statement $L(\rho(r)) \subseteq \rho(L(r)) \cup E$ is a rather easy exercise. □

The existence of economic shuffle resistant binary encodings that furthermore preserve star height is given by the next theorem—we omit the proof because of limited space.

Theorem 13. *Let Γ be an alphabet. There exists a morphism $\rho : \Gamma^* \to \{0, 1\}^*$ such that (1) $|\rho(a)| = O(\log |\Gamma|)$, for every symbol $a \in \Gamma$, and (2) the morphism ρ is shuffle resistant and preserves star height.* □

For regular expressions with interleaving we show that the conversion to ordinary regular expressions induces a $2^{2^{\Omega(n/\log n)}}$ lower bound for binary input alphabet.

Theorem 14. *There is an infinite family of languages L_n over a binary alphabet admitting regular expressions with interleaving of size $O(n)$, such that* $\mathrm{alph}(L_n) = 2^{2^{\Omega(n/\log n)}}$.

Proof. Our witness language will be described by the expression

$$\rho(r_n) = (\rho(a_1)\rho(b_1))^* \text{ ш } (\rho(a_2)\rho(b_2))^* \text{ ш } \cdots \text{ ш } (\rho(a_n)\rho(b_n))^*,$$

obtained by applying the morphism ρ from Theorem 13 to the expression r_n used in the proof of Theorem 8. This expression has size $O(n \log n)$, and to prove the theorem, it will suffice to establish that $L(\rho(r_n))$ has alphabetic width at least $2^{2^{\Omega(n)}}$.

Recall from the proof of Theorem 8 that the star height of $L(r_n)$ is bounded below by $2^{\Omega(n)}$. Since ρ preserves star height, the same bound applies to the language $\rho(L(r_n))$. By Theorem 1, we thus have

$$\operatorname{alph}(\rho(L(r_n))) = 2^{2^{\Omega(n)}}. \tag{2}$$

Unfortunately, this bound applies to $\rho(L(r_n))$ rather than to $L(\rho(r_n))$. At least, as we know from Theorem 13 that ρ is a shuffle resistant encoding, these two sets are related by

$$L(\rho(r_n)) \cap \rho(\Gamma)^* = \rho(L(r_n)), \tag{3}$$

with $\Gamma = \{a_1, b_1, \ldots, a_n, b_n\}$.

To derive a similar lower bound on the language $L(\rho(r_n))$, we use the upper bound $2^{O(n(1+\log m))}$ from [11] on the alphabetic width of the intersection for regular languages of alphabet width m and n, respectively, for $m \geq n$. To this end, let $\alpha(n)$ denote the alphabetic width of $L(\rho(r_n))$. We show first that $\alpha(n) > \operatorname{alph}(\rho(\Gamma)^*)$. Assume the contrary. By Theorem 13, the set $\rho(\Gamma)^*$ admits a regular expression of size $O(n \log n)$. Assuming $\alpha(n) \leq \operatorname{alph}(\rho(\Gamma)^*)$, the upper bound on the alphabetic width of intersection implies that $\rho(L(r_n)) = L(\rho(r_n)) \cap \rho(\Gamma^*)$ admits a regular expression of size $2^{O(n \log^2 n)}$. But this clearly contradicts Inequality (2). Thus, $\alpha(n) > \operatorname{alph}(\rho(\Gamma)^*)$. Applying the upper bound for intersection to the left-hand side of Equation (3), we obtain

$$\operatorname{alph}(\rho(L(r_n))) = \operatorname{alph}(L(\rho(r_n)) \cap \rho(\Gamma^*)) = 2^{O(n \log n \log \alpha(n))}. \tag{4}$$

Inequalities (2) and (4) now together imply that there exist positive constants c_1 and c_2 such that, for n large enough, holds $2^{2^{c_1 n}} \leq 2^{c_2 n \log n \log \alpha(n)}$. Taking double logarithms on both sides and rearranging terms, we obtain $c_1 n - O(\log n) \leq \log\log \alpha(n)$. Since the the left-hand side is in $\Omega(n)$, we thus have $\operatorname{alph}(L(\rho(r_n))) = \alpha(n) = 2^{2^{\Omega(n)}}$, and the proof is complete. □

References

1. Chandran, L.S., Kavitha, T.: The treewidth and pathwidth of hypercubes. Discrete Mathematics 306(3), 359–365 (2006)
2. Cohen, R.S.: Star height of certain families of regular events. Journal of Computer and System Sciences 4(3), 281–297 (1970)
3. Eggan, L.C.: Transition graphs and the star height of regular events. Michigan Mathematical Journal 10, 385–397 (1963)

4. Ehrenfeucht, A., Zeiger, H.P.: Complexity measures for regular expressions. Journal of Computer and System Sciences 12(2), 134–146 (1976)
5. Ellul, K., Krawetz, B., Shallit, J., Wang, M.-W.: Regular expressions: New results and open problems. Journal of Automata, Languages and Combinatorics 10(4), 407–437 (2005)
6. Fürer, M.: The complexity of the inequivalence problem for regular expressions with intersection. In: de Bakker, J.W., van Leeuwen, J. (eds.) ICALP 1980. LNCS, vol. 85, pp. 234–245. Springer, Heidelberg (1980)
7. Gelade, W.: Succinctness of regular expressions with interleaving, intersection and counting. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 363–374. Springer, Heidelberg (2008)
8. Gelade, W., Neven, F.: Succinctness of the complement and intersection of regular expressions. In: Albers, S., Weil, P. (eds.) STACS 2008, Dagstuhl Seminar Proceedings, vol. 08001, pp. 325–336. IBFI Schloss Dagstuhl, Wadern (2008)
9. Gruber, H., Holzer, M.: Finite automata, digraph connectivity, and regular expression size. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 39–50. Springer, Heidelberg (2008)
10. Gruber, H., Holzer, M.: Language operations with regular expressions of polynomial size. Theoretical Computer Science (accepted for publication) (2009)
11. Gruber, H., Holzer, M.: Provably shorter regular expressions from deterministic finite automata (Extended abstract). In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257. Springer, Heidelberg (2008)
12. Gruber, H., Johannsen, J.: Optimal lower bounds on regular expression size using communication complexity. In: Amadio, R. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 273–286. Springer, Heidelberg (2008)
13. Hashiguchi, K.: Algorithms for determining relative star height and star height. Information and Computation 78(2), 124–169 (1988)
14. Hashiguchi, K., Honda, N.: Homomorphisms that preserve star height. Information and Control 30(3), 247–266 (1976)
15. Hopcroft, J.E., Ullman, J.D.: Introduction to Automata Theory, Languages and Computation. Addison-Wesley, Reading (1979)
16. Johnson, T., Robertson, N., Seymour, P.D., Thomas, R.: Directed tree-width. Journal of Combinatorial Theory, Series B 82(1), 138–154 (2001)
17. Mayer, A.J., Stockmeyer, L.J.: Word problems - This time with interleaving. Information and Computation 115(2), 293–311 (1994)
18. McNaughton, R.: The loop complexity of pure-group events. Information and Control 11(1/2), 167–176 (1967)
19. Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars, and formal systems. In: IEEE Symposium on Switching and Automata Theory, pp. 188–191. IEEE Computer Society Press, Los Alamitos (1971)
20. Warmuth, M.K., Haussler, D.: On the complexity of iterated shuffle. Journal of Computer and System Sciences 28(3), 345–358 (1984)

Subshifts, Languages and Logic

Emmanuel Jeandel[1] and Guillaume Theyssier[2]

[1] Laboratoire d'informatique fondamentale de Marseille (LIF)
Aix-Marseille Université, CNRS
39 rue Joliot-Curie, 13 453 Marseille Cedex 13, France
Emmanuel.Jeandel@lif.univ-mrs.fr
[2] LAMA (Université de Savoie, CNRS)
Campus Scientifique, 73376 Le Bourget-du-lac cedex France
guillaume.theyssier@univ-savoie.fr

Abstract. We study the Monadic Second Order (MSO) Hierarchy over infinite pictures, that is tilings. We give a characterization of existential MSO in terms of tilings and projections of tilings. Conversely, we characterise logic fragments corresponding to various classes of infinite pictures (subshifts of finite type, sofic subshifts).

1 Introduction

There is a close connection between words and monadic second-order (MSO) logic. Büchi and Elgot proved for finite words that MSO-formulas correspond exactly to regular languages. This relationship was developed for other classes of labeled graphs; trees or infinite words enjoy a similar connection. See [20,13] for a survey of existing results. Colorings of the entire plane, i.e tilings, represent a natural generalization of biinfinite words to higher dimensions, and as such enjoy similar properties. We plan to study in this paper tilings for the point of view of monadic second-order logic.

Tilings and logic have a shared history. The introduction of tilings can be traced back to Hao Wang [21], who introduced his celebrated tiles to study the (un)decidability of the ∀∃∀ fragment of first order logic. The undecidability of the domino problem by his PhD Student Berger [3] lead then to the undecidability of this fragment [5]. Seese [10,18] used the domino problem to prove that graphs with a decidable MSO theory have a bounded tree width. Makowsky[12,15] used the construction by Robinson [16] to give the first example of a finitely axiomatizable super-stable theory that is super-stable. More recently, Oger [14] gave generalizations of classical results on tilings to locally finite relational structures. See the survey [2] for more details.

Previously, a finite variant of tilings, called tiling pictures, was studied [6,7]. Tiling pictures correspond to colorings of a *finite* region of the plane, this region being bordered by special '#' symbols. It is proven for this particular model that language recognized by EMSO-formulas correspond exactly to so-called finite tiling systems, i.e. projections of finite tilings.

The equivalent of finite tiling systems for infinite pictures are so-called *sofic subshifts* [22]. A *sofic subshift* represents intuitively local properties and ensures

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 288–299, 2009.

that every point of the plane behaves in the same way. As a consequence, there is no general way to enforce that some specific color, say □ appears at least once. Hence, some simple first-order existential formulas have no equivalent as sofic subshift (and even subshift). This is where the border of # for finite pictures play an important role: Without such a border, results on finite pictures would also stumble on this issue.

We deal primarily in this article with subshifts. See [1] for other acceptance conditions (what we called subshifts of finite type correspond to A-acceptance in this paper).

Finally, note that all decision problems in our context are non-trivial : To decide if a universal first-order formula is satisfiable (the domino problem, presented earlier) is not recursive. Worse, it is Σ^1_1-hard to decide if a tiling of the plane exists where some given color appears infinitely often [9,1]. As a consequence, the satisfiability of MSO-formulas is at least Σ^1_1-hard.

2 Symbolic Spaces and Logic

2.1 Configurations

Let $d \geq 1$ be a fixed integer and consider the discrete lattice $\mathbb{Z}^d$. For any finite set Q, a Q-configuration is a function from $\mathbb{Z}^d$ to Q. Q may be seen as a set of *colors* or *states*. An element of $\mathbb{Z}^d$ will be called a *cell*. A configuration will usually be denoted C, M or N.

Fig. 1 shows an example of two different configurations of $\mathbb{Z}^2$ over a set Q of 5 colors. As a configuration is infinite, only a finite fragment of the configurations is represented in the figure. The reader has to use his imagination to decide what colors do appear in the rest of the configuration. We choose not to represent which cell of the picture is the origin $(0, 0)$ (we use only translation invariant properties).

A *pattern* is a partial configuration. A pattern $P : X \to Q$ where $X \subseteq \mathbb{Z}^2$ occurs in $C \in Q^{\mathbb{Z}^d}$ at position z_0 if

$$\forall z \in X,\ C(z_0 + z) = P(z).$$

We say that P occurs in C if it occurs at some position in C. As an example the pattern P of Fig 2 occurs in the configuration M but not in N (or more accurately not on the finite fragment of N depicted in the figure). A finite pattern

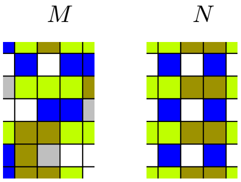


Fig. 1. Two configurations

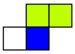


Fig. 2. A pattern P. P appears in M but presumably not in N

is a partial configuration of finite domain. All patterns in the following will be finite. The *language* $\mathcal{L}(C)$ of a configuration C is the set of finite patterns that occur in C. We naturally extend this notion to sets of configurations.

A *subshift* is a natural concept that captures both the notion of *uniformity* and *locality*: the only description "available" from a configuration C is the finite patterns it contains, that is $\mathcal{L}(C)$. Given a set $\mathcal{F}$ of patterns, let $X_{\mathcal{F}}$ be the set of all configurations where no patterns of $\mathcal{F}$ occurs.

$$X_{\mathcal{F}} = \{C | \mathcal{L}(C) \cap \mathcal{F} = \emptyset\}$$

$\mathcal{F}$ is usually called the set of forbidden patterns or the *forbidden language.* A set of the form $X_{\mathcal{F}}$ is called a *subshift.*

A subshift can be equivalently defined by topology considerations. Endow the set of configurations $Q^{\mathbb{Z}^d}$ with the product topology: A sequence $(C_n)_{n\in\mathbb{N}}$ of configurations converges to a configuration C if the sequence ultimately agree with C on every $z \in \mathbb{Z}^2$. Then a subshift is a closed subset of $Q^{\mathbb{Z}^d}$ also closed by shift maps.

A *subshift of finite type* (or *tiling*) correspond to a finite set $\mathcal{F}$: it is the set of configurations C such that no pattern in $\mathcal{F}$ occurs in C. If all patterns of $\mathcal{F}$ are of diameter n, this means that we only have to see a configuration through a window of size n to know if it is a tiling, hence the locality.

Given two state sets Q_1 and Q_2, a projection is a map $\pi : Q_1 \to Q_2$. We naturally extend it to $\pi : Q_1^{\mathbb{Z}^d} \to Q_2^{\mathbb{Z}^d}$ by $\pi(C)(z) = \pi(C(z))$. A *sofic subshift* of state set Q_2 is the image by some projection π of some subshift of finite type of state set Q_1. It is also a subshift (clearly closed by shift maps, and topologically closed because projections are continuous maps on a compact space). A sofic subshift is a natural object in tiling theory, although quite never mentioned explicitly. It represents the concept of *decoration*: some of the tiles we assemble to obtain the tilings may be decorated, but we forgot the decoration when we observe the tiling.

2.2 Structures

From now on, we restrict to dimension 2. A configuration will be seen in this article as an infinite structure. The signature τ contains four unary maps North, South, East, West and a predicate P_c for each color $c \in Q$.

A configuration M will be seen as a structure $\mathfrak{M}$ in the following way:

- The elements of $\mathfrak{M}$ are the points of $\mathbb{Z}^2$.
- North is interpreted by $\mathsf{North}^{\mathfrak{M}}((x, y)) = (x, y + 1)$, East is interpreted by $\mathsf{East}^{\mathfrak{M}}((x, y)) = (x + 1, y)$. $\mathsf{South}^{\mathfrak{M}}$ and $\mathsf{West}^{\mathfrak{M}}$ are interpreted similarly
- $P_c^{\mathfrak{M}}((x, y))$ is true if and only if the point at coordinate (x, y) is of color c, that is if $M(x, y) = c$.

As an example, the configuration M of Fig. 1 has three consecutive cells with the color 🟩. That is, the following formula is true:

$$\mathfrak{M} \models \exists z, P_{\text{🟩}}(z) \wedge P_{\text{🟩}}(\mathsf{East}(z)) \wedge P_{\text{🟩}}(\mathsf{East}(\mathsf{East}(z)))$$

As another example, the following formula states that the configuration has a vertical period of 2 (the color in the cell (x, y) is the same as the color in the cell $(x, y+2)$). The formula is false in the structure $\mathfrak{M}$ and true in the structure $\mathfrak{N}$ (if the reader chose to color the cells of N not shown in the picture correctly):

$$\forall z, \begin{cases} P_{\text{🟫}}(z) \implies P_{\text{🟫}}(\mathsf{North}(\mathsf{North}(z))) \\ P_{\square}(z) \implies P_{\square}(\mathsf{North}(\mathsf{North}(z))) \\ P_{\text{🟦}}(z) \implies P_{\text{🟦}}(\mathsf{North}(\mathsf{North}(z))) \\ P_{\text{🟩}}(z) \implies P_{\text{🟩}}(\mathsf{North}(\mathsf{North}(z))) \\ P_{\text{▨}}(z) \implies P_{\text{▨}}(\mathsf{North}(\mathsf{North}(z))) \end{cases}$$

2.3 Monadic Second-Order Logic

This paper studies connection between subshifts (seen as structures as explained above) and monadic second order sentences. First order variables (x, y, z, ...) are interpreted as points of $\mathbb{Z}^2$ and (monadic) second order variables (X, Y, Z, ...) as subsets of $\mathbb{Z}^2$.

Monadic second order formulas are defined as follows:

- a term is either a first-order variable or a function (South, North, East, West) applied to a term;
- atomic formulas are of the form $t_1 = t_2$ or $X(t_1)$ where t_1 and t_2 are terms and X is either a second order variable or a color predicate;
- formulas are build up from atomic formulas by means of boolean connectives and quantifiers $\exists$ and $\forall$ (which can be applied either to first-order variables or second order variables).

A formula is *closed* if no variable occurs free in it. A formula is FO if no second-order quantifier occurs in it. A formula is EMSO if it is of the form

$$\exists X_1, \ldots, \exists X_n, \phi(X)$$

where ϕ is FO. Given a formula $\phi(X_1, \ldots, X_n)$ with no free first-order variable and having only $X_1, \ldots, X_n$ as free second-order variables, a configuration M together with subsets $E_1, \ldots, E_n$ is a model of $\phi(X_1, \ldots, X_n)$, denoted

$$(M, E_1, \ldots, E_n) \models \phi(X_1, \ldots, X_n),$$

if ϕ is satisfied (in the usual sense) when M is interpreted as $\mathfrak{M}$ (see previous section) and E_i interprets X_i.

2.4 Definability

This paper studies the following problems: Given a formula ϕ of some logic, what can be said of the configurations that satisfy ϕ? Conversely, given a subshift, what kind of formula can characterise it?

Definition 2.1. *A set S of Q-configurations is defined by ϕ if*

$$S = \left\{ M \in Q^{\mathbb{Z}^2} \middle| \mathfrak{M} \models \phi \right\}$$

Two formulas ϕ and ϕ' are equivalent iff they define the same set of configurations.

A set S is $\mathcal{C}$-definable if it is defined by a formula $\phi \in \mathcal{C}$.

Note that a definable set is always closed by shift (a shift between 2 configurations induces an isomorphism between corresponding structures). It is not always closed: The set of {🟩, □}-configurations defined by the formula $\phi : \exists z, P_{\text{🟩}}(z)$ contains all configurations except the all-white one, hence is not closed.

When we are dealing with MSO formulas, the following remark is useful: second-order quantifiers may be represented as projection operations on sets of configurations. We formalize now this notion.

If $\pi : Q_1 \mapsto Q_2$ is a projection and S is a set of Q_1-configurations, we define the two following operators:

$$E(\pi)(S) = \left\{ M \in (Q_2)^{\mathbb{Z}^2} \middle| \exists N \in (Q_1)^{\mathbb{Z}^2}, \pi(N) = M \wedge N \in S \right\}$$

$$A(\pi)(S) = \left\{ M \in (Q_2)^{\mathbb{Z}^2} \middle| \forall N \in (Q_1)^{\mathbb{Z}^2}, \pi(N) = M \implies N \in S \right\}$$

Note that A is a dual of E, that is $A(\pi)(S) = {}^cE(\pi)({}^cS)$ where ^{c} represents complementation.

Proposition 2.2

- *A set S of Q-configurations is EMSO-definable if and only if there exists a set S' of Q' configurations and a map $\pi : Q' \mapsto Q$ such that $S = E(\pi)(S')$ and S' is FO-definable.*
- *The class of MSO-definable sets is the closure of the class of FO-definable sets by the operators E and A.*

Proof (Sketch). We prove here only the first item.

- Let $\phi = \exists X, \psi$ be a EMSO formula that defines a set S of Q-configurations. Let $Q' = Q \times \{0,1\}$ and π be the canonical projection from Q' to Q.
  Consider the formula ψ' obtained from ψ by replacing $X(t)$ by $\vee_{c \in Q} P_{(c,1)}(t)$ and $P_c(t)$ by $P_{(c,0)}(t) \vee P_{(c,1)}(t)$.
  Let S' be a set of Q' configurations defined by ψ'. Then is it clear that $S = E(\pi)(S')$. The generalization to more than one existential quantifier is straightforward.

- Let $S = E(\pi)(S')$ be a set of Q configurations, and S' FO-definable by the formula ϕ. Denote by $c_1 \dots c_n$ the elements of Q'. Consider the formula ϕ' obtained from ϕ where each P_{c_i} is replaced by X_i. Let

$$\psi = \exists X_1, \dots, \exists X_n, \begin{cases} \forall z, \vee_i X_i(z) \\ \forall z, \wedge_{i \neq j} (\neg X_i(z) \vee \neg X_j(z)) \\ \forall z, \wedge_i \left(X_i(z) \implies P_{\pi(c_i)}(z) \right) \\ \phi' \end{cases}$$

  Then ψ defines S. Note that the formula ψ constructed above is of the form $\exists X_1, \dots, \exists X_n (\forall z, \psi'(z)) \wedge \phi'$. This will be important later. □

Second-order quantifications will then be regarded in this paper either as projections operators or sets quantifiers.

3 Hanf Locality Lemma and EMSO

The first-order logic has a property that makes it suitable to deal with tilings and configurations: it is local. This is illustrated by Hanf's lemma [8,4,11].

Definition 3.1. *Two Q-configurations M and N are (n, k)-equivalent if for each Q-pattern P of size n:*

- *If P appears in M less than k times, then P appears the exact same number of times in M and in N*
- *If P appears in M more than k times, then P appears in N more than k times*

This notion is indeed an equivalence relation. Given n and k, it is clear that there is only finitely many equivalence classes for this relation.

The Hanf's local lemma can be formulated in our context as follows:

Theorem 3.2. *For every FO formula ϕ, there exists (n, k) such that*

$$\text{if } M \text{ and } N \text{ are } (n,k) \text{ equivalent, then } \mathfrak{M} \models \phi \iff \mathfrak{N} \models \phi$$

Corollary 3.3. *Every FO-definable set is a (finite) union of some (n, k)-equivalence classes.*

This is theorem 3.3 in [7], stated for finite configurations. Lemma 3.5 in the same paper gives a proof of Hanf's Local Lemma in our context.

Given (P, k) we consider the set $S_{=k}(P)$ of all configurations such that the pattern P occurs exactly k times (k may be taken equal to 0). The set $S_{\geq k}(P)$ is the set of all configurations such that the pattern P occurs more than k times.

We may rephrase the preceding corollary as:

Corollary 3.4. *Every FO-definable set is a positive combination (i.e. unions and intersections) of some $S_{=k}(P)$ and some $S_{\geq k}(P)$*

Theorem 3.5. *Every EMSO-definable set can be defined by a formula of the form:*

$$\exists X_1, \dots, \exists X_n,\ \left(\forall z_1,\, \phi_1(z_1, X_1, \dots, X_n)\right) \wedge \left(\exists z_1, \dots, \exists z_p,\, \phi_2(z_1 \dots z_p, X_1, \dots, X_n)\right),$$

where ϕ_1 and ϕ_2 are quantifier-free formulas.

See [20, Corollary 4.1] or [19, Corollary 4.2] for a similar result. This result is an easy consequence of [17, Theorem 3.2] (see also the corrigendum).

4 Logic Characterization of SFT and Sofic Subshifts

We start by a characterization of subshifts of finite type (SFTs, i.e tilings). The problem with SFTs is that they are closed neither by projection nor by union. As a consequence, the corresponding class of formulas is not very interesting:

Theorem 4.1. *A set of configurations is a SFT if and only if it is defined by a formula of the form $\forall z,\, \psi(z)$ where ψ is quantifier-free.*

Proof. Let $P_1 \dots P_n$ be patterns. To each P_i we associate the quantifier-free formula $\phi_{P_i}(z)$ which is true if and only if P_i appears at the position z. Then the subshifts that forbids patterns $P_1 \dots P_n$ is defined by the formula:

$$\forall z, \neg\phi_{P_1(z)} \wedge \dots \wedge \neg\phi_{P_n(z)}$$

Conversely, let ψ be a quantifier-free formula. Each term t_i in ψ is of the form $f_i(z)$ where f_i is some combination of the functions $\mathsf{North}, \mathsf{South}, \mathsf{East}$ and West, each f_i thus representing somehow some vector z_i ($f_i(z) = z + z_i$). Let Z be the collection of all vectors z_i that appear in the formula ψ. Now the fact that ψ is true at the position z only depends on the colors of the configuration at points $(z + z_1), \dots, (z + z_n)$, i.e. on the *pattern* of domain Z that occurs at position z. Let $\mathcal{P}$ be the set of patterns of domain Z that makes ψ false. Then the set S defined by ψ is the set of configurations where no patterns in $\mathcal{P}$ occurs, hence a SFT. □

Theorem 4.2. *A set S is a sofic subshift if and only if it is definable by a formula of the form*

$$\exists X_1, \dots, \exists X_n, \forall z_1, \dots, \forall z_p,\, \psi(X_1, \dots, X_n, z_1 \dots z_p)$$

where ψ is quantifier-free. Moreover, any such formula is equivalent to a formula of the same form but with a single universal quantifier ($p = 1$).

Note that the real difficulty in the proof of this theorem is to treat the only binary predicate, the equality (=). The reader might try to find a sofic subshift corresponding to the following formula before reading the proof:

$$\forall x, y, \Big(P_{🟩}(x) \wedge P_{🟦}(\mathsf{East}(y))\Big) \implies x = y$$

Proof. A sofic subshift being a projection of a SFT, one direction of the first assertion follows from the previous theorem and proposition 2.2.

Let $\mathcal{C}$ be the class of formulas of the form:

$$\exists X_1, \ldots, \exists X_n, \forall z_1, \ldots, \forall z_p,\ \psi(X_1, \ldots, X_n, z_1 \ldots z_{p_i})$$

Now we prove by induction on the number p of universal quantifiers that each formula of $\mathcal{C}$ is equivalent to a formula with only one universal quantifier. There is nothing to prove for $p = 1$. For $p > 1$, we rewrite the formula in conjunctive normal form:

$$\exists X_1, \ldots, \exists X_n, \forall z_1, \ldots, \forall z_p,\ \wedge_i \psi_i(X_1, \ldots, X_n, z_1 \ldots z_p)$$

where ψ_i is disjunctive. This is equivalent to

$$\exists X_1, \ldots, \exists X_n, \wedge_i \forall z_1, \ldots, \forall z_p,\ \psi_i(X_1, \ldots, X_n, z_1 \ldots z_p) \equiv \exists X_1, \ldots, \exists X_n, \wedge_i \eta_i$$

Now we treat each η_i separately. ψ_i is a disjunction of four types of formulas:

$$\bullet P_c(f(x)) \qquad \bullet \neg P_c(f(x)) \qquad \bullet f(x) = y \qquad \bullet f(x) \neq y$$

because terms are made only of bijective functions (compositions of North, South, East, West). We may suppose the last case never happens: $\forall x, y, z f(x) \neq y \vee \psi(x, y, z)$ is equivalent to $\forall x, z, \psi(x, f(x), z)$. We may rewrite

$$\psi_i(z_1 \ldots z_p) \equiv \epsilon(z_p) \vee z_p = f(z_{k_1}) \vee \cdots \vee z_p = f(z_{k_m}) \vee \theta(z_1 \ldots z_{p-1})$$

(we forgot the second-order variables to simplify notations).

We may suppose that no formula is of the form $z_p = z_p$. Now is the key argument: Suppose that there are strictly more that m values of z such that $\epsilon(z)$ is false. Then given $z_1 \ldots z_{p-1}$ we may find a z_p such that the formula $\epsilon(z_p) \vee (z_p = f(z_{k_1})) \vee \cdots \vee (z_p = f(z_{k_m}))$ is false. That is, if there are more than m values of z so that $\epsilon(z)$ is false, then $\forall z_1, \ldots, \forall z_{p-1},\ \theta(z_1 \ldots z_{p-1})$ must be true.

As a consequence, our formula η_i is equivalent to the disjunction of the formula $\forall z_1, \ldots, \forall z_{p-1},\ \theta(z_1 \ldots z_{p-1})$ and the formula

$$\exists S_1, \ldots, \exists S_m, \begin{cases} \Psi_i \\ \forall z, \vee_i S_i(z) \iff \neg \epsilon(z) \\ \forall z_1, \ldots, \forall z_{p-1},\ S_1(f(z_{k_1})) \vee \cdots \vee S_m(f(z_{k_m})) \vee \theta(z_1 \ldots z_{p-1}) \end{cases}$$

where Ψ_i express that S_i has at most one element and is defined as follows:

$$\Psi_i \overset{def}{=} \exists A, \forall x \begin{cases} A(x) \iff A(\mathsf{North}(x)) \wedge A(\mathsf{East}(x)) \\ S_i(x) \iff A(x) \wedge \neg A(\mathsf{South}(x)) \wedge \neg A(\mathsf{West}(x)) \end{cases}$$

Simplifying notations, our formula η_i is equivalent to

$$\forall z_1, \ldots, \forall z_{p-1},\ \theta(z_1 \ldots z_{p-1}) \vee \exists P_1, \ldots, \exists P_q \forall z_1, \ldots, \forall z_{p-1},\ \zeta(z_1 \ldots z_{p-1})$$

which is equivalent to

$$\exists X, \exists P_1, \ldots, \exists P_q \forall z_1, \ldots, \forall z_{p-1}, \begin{cases} X(z_1) \iff X(\mathsf{North}(z_1)) \\ X(z_1) \iff X(\mathsf{East}(z_1)) \\ X(z_1) \implies \theta(z_1, \ldots z_{p-1}) \\ \neg X(z_1) \implies \zeta(z_1, \ldots z_{p-1}) \end{cases}$$

Now report this new formula instead of η_i to obtain a formula

$$\exists X_1, \ldots, \exists X_n, \wedge_i \exists R_1, \ldots, \exists R_{q_i}, \forall z_1, \ldots, \forall z_{p-1}, \theta_i(z_1 \ldots z_{p_i}, R_1 \ldots R_{q_i})$$

equivalent to

$$\exists X_1, \ldots, \exists X_n, \exists R_{11}, \ldots, \exists R_{kq_k}, \forall z_1, \ldots, \forall z_{p-1}, \wedge_i \theta_i(z_1 \ldots z_{p_i}, R_{i1} \ldots R_{iq_i})$$

We finally obtain a formula of $\mathcal{C}$ with $p-1$ universal quantifiers, and we may conclude by induction.

To finish the proof, a formula of $\mathcal{C}$ with 1 universal quantifier defines indeed a sofic subshift (use the proof of theorem 4.1. to conclude that this formula defines a projection of a SFT, hence a sofic subshift). □

5 Separation Result

Theorems 3.5. and 4.2. above suggest that EMSO-definable subshifts are not necessarily sofic. We will show in this section that the set of EMSO-definable subshifts is indeed strictly larger than the set of sofic subshifts. The proof is based on the analysis of the computational complexity of forbidden languages. It is well-known that sofic subshifts have a recursively enumerable forbidden language. The following theorem shows that the forbidden language of an MSO-definable subshift can be arbitrarily high in the arithmetical hierarchy.

This is not surprising since arbitrary Turing computation can be defined via first order formulas (using tilesets) and second order quantifiers can be used to simulate quantification of the arithmetical hierarchy. However, some care must be taken to ensure that the set of configurations obtained is a subshift.

Theorem 5.1. *Let E be an arithmetical set. Then there is an MSO-definable subshift with forbidden language $\mathcal{F}$ such that E reduces to $\mathcal{F}$ (for many-one reduction).*

Proof (sketch). Suppose that the complement of E is defined as the set of integers m such that $\exists x_1, \forall x_2, \ldots, \exists/\forall x_n, R(m, x_1, \ldots, x_n)$ where R is a recursive relation. We first build a formula ϕ defining the set of configurations representing a successful computation of R on some input $m, x_1, \ldots, x_n$. Consider 3 colors c_l, c and c_r and additional second order variables $X_1, \ldots, X_n$ and $S_1, \ldots, S_n$. The input $(m, x_1, \ldots, x_n)$ to the computation is encoded in unary on an horizontal segment using colors c_l and c_r and variables S_i as separators, precisely: first an occurrence of c_l then m occurrences of c, then an occurrence of c_r and, for each successive $1 \leq i \leq n$, x_i positions in X_i before a position of S_i. Let ϕ_1 be the FO formula expressing the following:

1. there is exactly 1 occurrence of c_l and 1 of c_r and all S_i are singletons;
2. starting from an occurrence c_l and going east until reaching S_n, the only possible successions of states are those forming a valid input as explained above.

Now, the computation of R on any input encoded as above can be simulated via tiling constraints in the usual way. Consider sufficiently many new second order variables $Y_1, \ldots, Y_p$ to handle the computation and let ϕ_2 be the FO formula expressing that:

1. a valid computation starts at the north of an occurrence of c_l;
2. there is exactly one occurrence of the halting state (represented by some Y_i) in the whole configuration.

We define ϕ by $\exists X_1, \forall X_2, \ldots, \exists/\forall X_n, \exists S_1, \ldots, \exists S_n, \exists Y_1, \ldots, \exists Y_p, \phi_1 \wedge \phi_2$.

Finally let ψ be the following FO formula: $(\forall z, \neg P_{c_l}) \vee (\forall z, \neg P_{c_r})$. Let X be the set defined by $\phi \vee \psi$. By construction, a finite (unidimensional) pattern of the form $c_l c^m c_r$ appears in some configuration of X if and only if $m \notin E$. Therefore E is many-one reducible to the forbidden language of X.

To conclude the proof it is sufficient to check that X is closed. To see this, consider a sequence $(C_n)_n$ of configurations of X converging to some configuration C. C has at most one occurrence of c_l and one occurrence of c_r. If one of these two states does not occur in C then $C \in X$ since ψ is verified. If, conversely, both c_l and c_r occur (once each) then any pattern containing both occurrences also occurs in some configuration C_n verifying ϕ. But ϕ is such that any modification outside the segment between c_l and c_r in C_n does not change the fact that ϕ is satisfied provided no new c_l and c_r colors are added. Therefore ϕ is also satisfied by C and $C \in X$. □

The construction of the theorem gives the claimed separation result for subshifts of EMSO if we choose E to be the set of non-halting Turing machines.

Corollary 5.2. *There are EMSO-definable subshifts which are not sofic.*

6 A Characterization of EMSO

EMSO-definable sets are projections of FO-definable sets (proposition 2.2). Besides, sofic subshifts are projections of subshifts of finite type (or tilings). Previous results show that the correspondence sofic↔EMSO fails. However, we will show in this section how EMSO can be characterized through projections of "locally checkable" configurations.

Corollary 3.4. expresses that FO-definable sets are essentially captured by counting occurrences of patterns up to some value. The key idea in the following is that this counting can be achieved by local checkings (equivalently, by tiling constraints), provided it is limited to a finite and explicitly delimited region. This idea was successfully used in [7] in the context of picture languages: pictures are rectangular finite patterns with a border made explicit using a special state (which occurs all along the border and nowhere else). We will proceed here quite differently. Instead of putting special states on borders of some rectangular zone, we will simply require that two special subsets of states Q_0 and Q_1 are present in the configuration: we call a (Q_0, Q_1)-*marked configuration* any configuration that contains both a color $q \in Q_0$ and some color $q' \in Q_1$ somewhere. By extension, given a subshift Σ over Q and two subsets $Q_0 \subseteq Q$ and $Q_1 \subseteq Q$, the *doubly-marked set* Σ_{Q_0,Q_1} is the set of (Q_0, Q_1)-marked configurations of Σ. Finally, a *doubly-marked set of finite type* is a set Σ_{Q_0,Q_1} for some SFT Σ and some Q_0, Q_1.

Lemma 6.1. *For any finite pattern P and any $k \geq 0$, $S_{=k}(P)$ is the projection of some doubly-marked set of finite type. The same result holds for $S_{\geq k}(P)$.*

Moreover, any positive combination (union and intersection) of projections of doubly-marked sets of finite type is also the projection of some doubly-marked sets of finite type.

Theorem 6.2. *A set is EMSO-definable if and only if it is the projection of a doubly-marked set of finite type.*

Proof. First, a doubly-marked set of finite type is an FO-definable set because SFT are FO-definable (theorem 4.1.) and the restriction to doubly-marked configurations can be expressed through a simple existential FO formula. Thus the projection of a doubly-marked set of finite type is EMSO-definable.

The opposite direction follows immediately from proposition 2.2 and corollary 3.4. and the lemma above. □

At this point, one could wonder whether considering simply-marked set of finite type is sufficient to capture EMSO via projections. In fact the presence of 2 markers is necessary in the above theorem: considering the set Σ_{Q_0,Q_1} where Σ is the full shift $Q^{\mathbb{Z}^2}$ and Q_0 and Q_1 are distinct singleton subsets of Q, a simple compactness argument allows to show that it is not the projection of any simply-marked set of finite type.

7 Open Problems

- Is the second order alternation hierarchy strict for MSO (considering our model-theoretic equivalence)?
- One can prove that theorem 4.1. also holds for formulas of the form:

$$\forall X_1 \dots \forall X_n, \forall z,\ \psi(z, X_1 \dots X_n)$$

  where ψ is quantifier-free. Hence, adding universal second-order quantifiers does not increase the expression power of formulas of theorem 4.1.. More generally, let $\mathcal{C}$ be the class of formulas of the form

$$\forall X_1, \exists X_2, \dots, \forall/\exists X_n, \forall z_1, \dots, \forall z_p, \phi(X_1, \dots, X_n, z_1, \dots, z_p).$$

  One can check that any formula in $\mathcal{C}$ defines a subshift. Is the second-order quantifiers alternation hierarchy strict in $\mathcal{C}$? On the contrary, do all formulas in $\mathcal{C}$ represent sofic subshifts ?

References

1. Altenbernd, J.-H., Thomas, W., Wohrle, S.: Tiling systems over infinite pictures and their acceptance conditions. In: Developments in Language Theory (2003)
2. Ballier, A., Jeandel, E.: Tilings and model theory. In: First Symposium on Cellular Automata Journées Automates Cellulaires (2008)
3. Berger, R.: The undecidability of the domino problem. Memoirs American Mathematical Society 66, 1966 (1966)
4. Ebbinghaus, H.-D., Flum, J.: Finite Model Theory. Springer Monographs in Mathematics. Springer, Berlin (1995)
5. Börger, Y.G.E., Grädel, E.: The classical decision problem. Springer, Telos (1996)
6. Giammarresi, D., Restivo, A.: Two-Dimensional Languages. In: Handbook of Formal Languages, Beyond Words, vol. 3. Springer, Heidelberg (1997)
7. Giammarresi, D., Restivo, A., Seibert, S., Thomas, W.: Monadic Second-Order Logic over Rectangular Pictures and Recognizability by Tiling Systems. Information and Computation 125, 32–45 (1996)
8. Hanf, W.: Model-theoretic methods in the study of elementary logic. In: Symposium in the Theory of Models, pp. 132–145 (1965)
9. Harel, D.: Recurring Dominoes: Making the Highly Undecidable Highly Understandable. Annals of Discrete Mathematics 24, 51–72 (1985)
10. Kuske, D., Lohrey, M.: Logical aspects of Cayley-graphs: the group case. Annals of Pure and Applied Logic 131(1-3), 263–286 (2005)
11. Libkin, L.: Elements of Finite Model Theory. Texts in Theoretical Computer Science, an EATCS Series. Springer, Heidelberg (2004)
12. Makowsky, J.A.: On some conjectures connected with complete sentences. Fund. Math. 81, 193–202 (1974)
13. Matz, O., Schweikardt, N.: Expressive power of monadic logics on words, trees, pictures, and graphs. In: Gradel, E., Flum, J., Wilke, T. (eds.) Logic and Automata: History and Perspectives. Texts in Logic and Games, pp. 531–552. Amsterdam University Press (2007)
14. Oger, F.: Algebraic and model-theoretic properties of tilings. Theoretical computer science 319, 103–126 (2004)
15. Poizat, B.: Une théorie finiement axiomatisable et superstable. Groupe d'étude des théories stables 3(1), 1–9 (1980-1882)
16. Robinson, R.M.: Undecidability and Nonperiodicity for Tilings of the Plane. Inventiones Math. 12 (1971)
17. Schwentick, T., Barthelmann, K.: Local Normal Forms for First-Order Logic with Applications to Games and Automata. Discrete Mathematics and Theoretical Computer Science 3, 109–124 (1999)
18. Seese, D.: The structure of the models of decidable monadic theories of graphs. Annals of pure and applied logic 53(2), 169–195 (1991)
19. Thomas, W.: On logics, tilings, and automata. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 441–454. Springer, Heidelberg (1991)
20. Thomas, W.: Languages, Automata, and Logic. In: Thomas, W. (ed.) Handbook of Formal Languages, Beyond Words, vol. 3. Springer, Heidelberg (1997)
21. Wang, H.: Proving theorems by pattern recognition ii. Bell system technical journal 40, 1–41 (1961)
22. Weiss, B.: Subshifts of finite type and sofic systems. Monatshefte für Mathematik 77, 462–474 (1973)

Magic Numbers and Ternary Alphabet*

Galina Jirásková

Mathematical Institute, Slovak Academy of Sciences,
Grešákova 6, 040 01 Košice, Slovakia
jiraskov@saske.sk

Abstract. A number α, in the range from n to 2^n, is magic for n with respect to a given alphabet size s, if there is no minimal nondeterministic finite automaton of n states and s input letters whose equivalent minimal deterministic finite automaton has α states. We show that in the case of a ternary alphabet, there are no magic numbers. For all n and α satisfying that $n \leqslant \alpha \leqslant 2^n$, we describe an n-state nondeterministic automaton with a three-letter input alphabet that needs α deterministic states.

1 Introduction

At the 3rd Conference on Developments in Language Theory, Iwama, Kambayashi, and Takaki [10] stated the question of whether there always exists a minimal nondeterministic finite automaton (nfa) of n states whose equivalent minimal deterministic finite automaton (dfa) has α states for all integers n and α satisfying that $n \leqslant \alpha \leqslant 2^n$. The question has also been considered by Iwama, Matsuura, and Paterson in [11], where a number Z with $n \leqslant Z \leqslant 2^n$ is called "magic", if there is no nfa of n states that needs Z deterministic states. In these two papers it is shown that if $\alpha = 2^n - 2^k$ or $\alpha = 2^n - 2^k - 1$, where $0 \leqslant k \leqslant n/2 - 2$, or if $\alpha = 2^n - k$, where $2 \leqslant k \leqslant 2n - 2$ and some coprimality condition holds, then the number α is not magic. The authors defined corresponding automata over a two-letter alphabet, and have mentioned that if we allow more input symbols, the problem becomes easier.

In the case when the alphabet size is allowed to grow exponentially with n, appropriate automata have been described for all values of n and α in [12]. It has been shown that a $2n$-letter alphabet would be enough, however, in this case, automata were given implicitly. For a binary alphabet, all numbers from n to $n^2/2$ have been shown to be non-magic. The explicit constructions of automata using $n+2$ input symbols have been presented by Geffert [6], who also considered a binary case and proved that all numbers from n to $2^{n^{1/3}}$ are non-magic.

The problem has been solved for a fixed four-letter alphabet by Jirásek, Jirásková, and Szabari [14] by describing, for all n and α with $n \leqslant \alpha \leqslant 2^n$, a minimal nondeterministic finite automaton of n states with a four-letter input alphabet that needs α deterministic states. This means that in the case of a four-letter alphabet, there are no magic numbers.

* Research supported by VEGA grant 2/0111/09.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 300–311, 2009.

Let us note that in the unary case, all numbers from $\mathrm{e}^{(1+o(1))\cdot\sqrt{n\ln n}}$ to 2^n are magic since every n-state unary nfa can be simulated by an $\mathrm{e}^{(1+o(1))\cdot\sqrt{n\ln n}}$-state dfa [15,4,7]. Moreover, it has been shown by Geffert [7] that there are much more magic than non-magic numbers in the range from n to $\mathrm{e}^{(1\pm o(1))\cdot(\sqrt{n\ln n})}$ in the unary case.

Here we continue this research and study a ternary case. We show that neither in this case do magic numbers exist, and give the explicit constructions of appropriate automata with a three-letter input alphabet. Surprisingly, the constructions and proofs are even easier than in the case of a four-letter alphabet. The question of whether or not there are some magic numbers for a binary alphabet remains open. Some partial results for the binary case have recently been obtained by Matsuura and Saito in [16] and by the author in [13]. The first paper shows that, with some exceptions, all numbers from 2^n to $2^n - 4n$ are non-magic, while in the the second one, all numbers from n to $2^{n/3}$ are proved to be non-magic in the binary case.

To conclude this section we mention two more related works. Magic numbers for symmetric difference nfa's have been studied by Zijl [20], and similar problems for nonterminal complexity of some operations on context-free languages have been investigated by Dassow and Stiebe [5].

2 Preliminaries

In this section, we give some basic definitions, notations, and preliminary results used throughout the paper. For further details, we refer to [18,19].

Let Σ be a finite alphabet and Σ^* the set of all strings over the alphabet Σ including the empty string ε. The length of a string w is denoted by $|w|$. 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 quintuple $M = (Q, \Sigma, \delta, q_0, F)$, 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 quintuple $M = (Q, \Sigma, \delta, q_0, F)$, 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. The transition function 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 of strings $L(M) = \{w \in \Sigma^* \mid \delta(q_0, w) \cap F \neq \varnothing\}$.

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 its states are reachable from the initial state, and *(ii)* no two of its different states are equivalent (states p and q are said to be equivalent if for all strings w in Σ^*, the state $\delta(p, w)$ is accepting iff the state $\delta(q, w)$ is accepting). Each regular language has a unique minimal dfa, up to isomorphism. However, the same result does not hold for nfa's.

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" [17] in the following way. Every state of the dfa M' is a subset of the state set Q. The initial state of the dfa M' is the set $\{q_0\}$. The transition function δ' is defined by $\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$ for each state R in 2^Q and each symbol a in Σ. A state R in 2^Q is an accepting state of the dfa M' if it contains an accepting state of the nfa M. We call the automaton M' the subset automaton corresponding to the nfa M. Notice that the subset automaton need not be minimal since some of its states may be unreachable or equivalent.

To prove that an nfa is minimal we use a fooling-set lower-bound technique [1,2,3,8,9]. After defining a fooling set, we recall 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 regular language* L *if for every* i *and* j *in* $\{1, 2, \ldots, n\}$,

(1) the string $x_i y_i$ *is in the language* L, *and*

(2) 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 ([2], Lemma 1). *Let* $\mathcal{A}$ *be a fooling set for a regular language* L. *Then every nfa for the language* L *needs at least* $|\mathcal{A}|$ *states.* □

The following lemma from [14] shows that each integer can be expressed as a sum of powers of two decreased by one if, possibly, the smallest summand can be taken twice. We will use this lemma later in our constructions.

Lemma 2 ([14], Lemma 2). *Let* k *be a positive integer. Then for each integer* m *such that* $1 \leqslant m < 2^k$, *one of the following three cases holds:*

$$m = 2^k - 1,$$
$$m = (2^{k_1} - 1) + (2^{k_2} - 1) + \cdots + (2^{k_{\ell-1}} - 1) + (2^{k_\ell} - 1),$$
$$m = (2^{k_1} - 1) + (2^{k_2} - 1) + \cdots + (2^{k_{\ell-1}} - 1) + 2 \cdot (2^{k_\ell} - 1),$$

where $1 \leqslant \ell \leqslant k - 1$, *and* $k - 1 \geqslant k_1 > k_2 > \cdots > k_\ell \geqslant 1$. □

3 Main Results

The aim of this section is to show that in the case of a three-letter alphabet, there are no magic numbers, which means that each value in the range from n to 2^n can be obtained as the size of the minimal dfa equivalent to a minimal n-state nfa defined over a three-letter alphabet. Let us start with an example.

Example 1. Consider the five-state nfa A_5 with the input alphabet $\{a, b, c\}$ shown in Fig. 1. The automaton has states 0,1,2,3,4, of which state 4 is the initial and the sole accepting state. On inputs a and b, each state i goes to state $i + 1$, except for state 4 which goes to itself. Moreover, on input b, each state goes to state 0. There is only one transition on input c, which goes from state 3 to state 4.

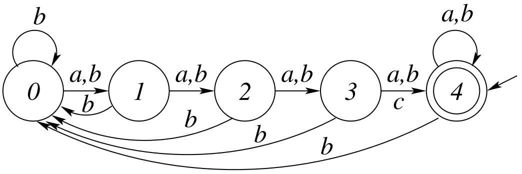


Fig. 1. The nondeterministic finite automaton A_5

Consider the subset automaton A'_5 corresponding to the nfa A_5. Fig. 2 shows the dfa A'_5. To keep the figure transparent, we label the states of the dfa without commas and brackets, and we omit transitions in some states.

We can see that state 4, by each string in $\{a, b\}^*$, goes to a subset containing state 4. Moreover, each such subset is reachable from state 4 by a string over $\{a, b\}$. We prove this property in a general case in Lemma 3.

Next, consider states $\{1\}, \{1, 2\}, \{1, 2, 3\}$, depicted by a red (grey) color in the figure. Notice that from state $\{1\}$, seven subsets of $\{0, 1, 2, 3\}$, that is, subsets not containing state 4, can be reached by a string over $\{a, b\}$. Three more subsets can be reached from state $\{1, 2\}$, and one more subset is reachable from state $\{1, 2, 3\}$ (and even one more from state $\{0, 1, 2, 3\}$).

Thus, if we would like to have a five-state nfa requiring, for example, $1+16+7+3+1 = 28$ deterministic states, let us call it $B_{5,28}$, we can construct it from the nfa A_5 by adding transitions on symbol c from states 0, 1, and 2 to subsets $\{1\}$,$\{1, 2\}$, and $\{1, 2, 3\}$, respectively. The nondeterministic finite automaton $B_{5,28}$ is minimal since the set $\{(\varepsilon, ba^3c), (b, a^3c), (ba, a^2c), (ba^2, ac), (ba^3, c)\}$ is a fooling set of size five for the language $L(B_{5,28})$. Next, the empty string is accepted by the nfa $B_{5,28}$ only from state 4, while the string $a^{3-i}c$ is accepted only from state i for all $i = 0, 1, 2, 3$. This means that all reachable sets in the subset automaton are pairwise inequivalent.

The following subsets are reachable in the subset automaton $B'_{5,28}$:

- the empty set,
- all subsets containing state 4 (16 subsets),
- state $\{0, 4\}$ goes by c to state $\{1\}$, from which 7 subsets of $\{0, 1, 2, 3\}$ can be reached,
- state $\{1, 4\}$ goes by c to state $\{1, 2\}$, from which 3 more subsets of $\{0, 1, 2, 3\}$ can be reached,
- state $\{2, 4\}$ goes by c to a new subset $\{1, 2, 3\}$,
- no other subset is reachable.

Hence the total number of reachable states is $1 + 16 + 7 + 3 + 1 = 28$.

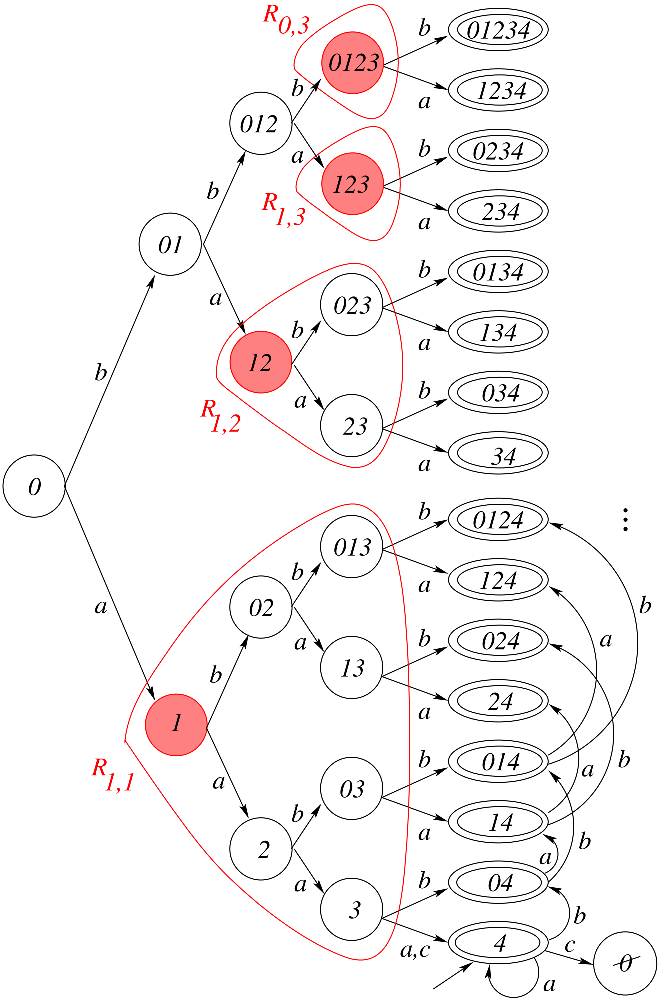


Fig. 2. The deterministic finite automaton A'_5

If we would like to get a five-state nfa $B_{5,17+7+7}$ requiring 31 deterministic states, we would define $0 \xrightarrow{c} \{1\}$, $1 \xrightarrow{c} \{0,1\}$, and $3 \xrightarrow{c} \{4\}$, while in the case of an nfa $B_{5,17+7+3+3}$ requiring 30 deterministic states we would have $0 \xrightarrow{c} \{1\}$, $1 \xrightarrow{c} \{1,2\}$, $2 \xrightarrow{c} \{0,1,2\}$, and $3 \xrightarrow{c} \{4\}$. A problem could arise if we would like to reach $17+7+3+1+1$ states in the subset automaton, however, in this case we will use $3 \xrightarrow{c} \{0,1,2,3\}$ instead of $3 \xrightarrow{c} \{4\}$, and the inequivalence still will be guaranteed. In such a way, using Lemma 2, we can define a five-state nfa $B_{5,17+m}$ that needs $17+m$ deterministic states for each m with $1 \leqslant m \leqslant 15$. □

Let us now generalize the above example. Our first aim is to define a ternary k-state nondeterministic automaton $B_{k,\beta}$ that needs β deterministic states, for each β greater than 2^{k-1}. We will use these automata later in our constructions to get the whole range of complexities from n to 2^n.

To this aim define a ternary k-state nfa $A_k = (Q_k, \{a, b, c\}, \delta, k-1, \{k-1\})$, where $Q_k = \{0, 1, \ldots, k-1\}$, and for each i in Q_k,

$$\delta(i,a) = \begin{cases} \{i+1\}, \text{ if } 0 \leqslant i \leqslant k-2, \\ \{k-1\}, \text{if } i = k-1, \end{cases}$$

$$\delta(i,b) = \begin{cases} \{0, i+1\}, \text{ if } 0 \leqslant i \leqslant k-2, \\ \{0, k-1\}, \text{if } i = k-1, \end{cases}$$

$$\delta(i,c) = \begin{cases} \{k-1\}, & \text{if } i = k-2, \\ \varnothing, & \text{if } i \neq k-2, \end{cases}$$

that is, on inputs a and b, each state i goes to state $i+1$ except for state $k-1$ which goes to itself. Moreover, on symbol b, each state also goes to state 0. Transitions on input c are defined only in state $k-2$, which goes to state $k-1$ on c. The automaton A_k is depicted in Fig. 3.

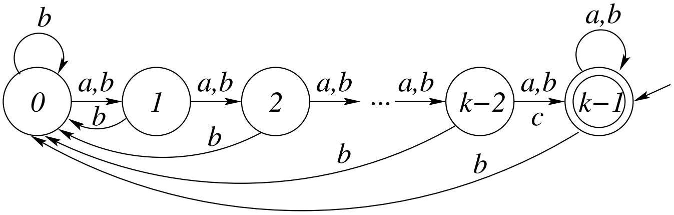


Fig. 3. The nondeterministic finite automaton A_k

Let $A'_k = (2^{Q_k}, \{a, b, c\}, \delta', \{k-1\}, F')$ be the subset automaton corresponding to the nfa A_k. Recall that the subset automaton contains all 2^k states, some of which are unreachable. Since the initial state $k-1$ of the nfa A_k goes to itself on symbols a and b, and the only transition on c goes to state $k-1$, all nonempty subsets of Q_k that do not contain state $k-1$ are unreachable in the dfa A'_k. On the other hand, let us show that all the other subsets are reachable.

Lemma 3. *All subsets of Q_k containing state $k-1$ are reachable in the subset automaton A'_k from the initial state $k-1$ via a string over the binary alphabet $\{a, b\}$. The empty set is reachable as well.*

Proof. The empty set is reachable since transitions on c are not defined in state $k-1$. We prove by induction on the size of subsets that each subset containing state $k-1$ is reachable. The singleton $\{k-1\}$ is reachable since it is the initial state of the subset automaton A'_k. Let $1 \leqslant t \leqslant k-1$ and assume that each subset of size t containing state $k-1$ is reachable via a string over $\{a, b\}$. Let $\{i_1, i_2, \ldots, i_t, k-1\}$ be a subset of size $t+1$ such that $0 \leqslant i_1 < i_2 < \cdots < i_t < k-1$. Then we have

$$\{i_1, i_2, \ldots, i_t, k-1\} = \delta'(\{i_2 - i_1 - 1, i_3 - i_1 - 1, \ldots, i_t - i_1 - 1, k-1\}, ba^{i_1}),$$

where the latter set of size t is reachable via a string over $\{a, b\}$ by the induction hypothesis. Hence the subset $\{i_1, i_2, \ldots, i_t, k-1\}$ is reachable, which completes our proof. □

If we look at the dfa A'_5 in our Example 1, see Fig. 2, like at a tree rooted in state $\{0\}$, then all (accepting) states in the bottom row can be reached from the initial state $\{4\}$ by a string over $\{a, b\}$, and the empty set can be reached from state $\{4\}$ by c. Hence the dfa has 17 reachable states.

In a general case, the subset automaton A'_k has $2^{k-1}+1$ reachable states. We now continue our constructions by adding new transitions on input c to get nfa's whose corresponding subset automata would have more reachable states. As a result we will be able to construct an nfa requiring β deterministic states for each β between $2^{k-1}+1$ and 2^k.

To this aim consider states $\{1\}, \{1,2\}, \{1,2,3\}, \ldots \{1,2,\ldots,k-2\}$ of the subset automaton A'_k. Notice that $\{1\} \subset \{1,2\} \subset \{1,2,3\} \subset \cdots \subset \{1,2,\ldots,k-2\}$, which is an important property that will be crucial in the proof of our main result. Consider also subsets not containing state $k-1$ that can be reached from these states via strings over $\{a, b\}$, and let us introduce some notation.

Let $1 \leqslant r \leqslant k-2$. Let

$$\mathcal{R}_{0,0} = \{R \subseteq \{0,1,\ldots,k-2\} \mid R = \delta'(\{0\}, w) \text{ for some } w \text{ in } \{a,b\}^*\},$$

$$\mathcal{R}_{0,r} = \{R \subseteq \{0,1,\ldots,k-2\} \mid R = \delta'(\{0,1,2,\ldots,r\}, w) \text{ for some } w \text{ in } \{a,b\}^*\},$$

$$\mathcal{R}_{1,r} = \{R \subseteq \{0,1,\ldots,k-2\} \mid R = \delta'(\{1,2,3,\ldots,r\}, w) \text{ for some } w \text{ in } \{a,b\}^*\},$$

that is, $\mathcal{R}_{0,0}$, $\mathcal{R}_{0,r}$, and $\mathcal{R}_{1,r}$ are the systems of subsets that do not contain state $k-1$ and can be reached via strings over $\{a, b\}$ from states $\{0\}$, $\{0,1,2,\ldots,r\}$, and $\{1,2,3,\ldots,r\}$, respectively. In our Example 1, see Fig. 2, we have $\mathcal{R}_{1,2} = \big\{\{1,2\},\{0,1,2\},\{2,3\}\big\}$ and $\mathcal{R}_{0,3} = \big\{\{0,1,2,3\}\big\}$. We can see that in this example, the systems $\mathcal{R}_{1,1}$, $\mathcal{R}_{1,2}$, and $\mathcal{R}_{1,3}$ are pairwise disjoint and contain $2^3-1, 2^2-1$ and 2^1-1 states, respectively. Let us consider a general case.

Lemma 4. *Let $1 \leqslant s \leqslant r \leqslant k-2$ and let $\mathcal{R}_{0,0}$, $\mathcal{R}_{0,r}$, and $\mathcal{R}_{1,r}$ be the systems of states defined above. Then we have:*

(i) The size of the system $\mathcal{R}_{0,0}$ is $2^{k-1}-1$.
(ii) The size of the system $\mathcal{R}_{0,r}$ and of the system $\mathcal{R}_{1,r}$ is $2^{k-r-1}-1$.
(iii) The systems $\mathcal{R}_{1,s}$ and $\mathcal{R}_{0,r}$ are disjoint.
(iv) If $s < r$, then the systems $\mathcal{R}_{1,s}$ and $\mathcal{R}_{1,r}$ are disjoint.

Proof. The proof of the lemma proceeds in a similar way as in [14, Lemma 4]. We prove, by induction on ℓ, that the set of states that are reachable from states $\{1,2,\ldots,r\}$ and $\{0,1,2,\ldots,r\}$ via strings over $\{a,b\}$ of length ℓ, with $0 \leqslant \ell \leqslant k-r-2$, are

$$\big\{S \cup \{1+\ell, 2+\ell, \ldots, r+\ell\} \mid S \subseteq \{0,1,\ldots,\ell-1\}\big\}$$

and

$$\{S \cup \{\ell, 1+\ell, \ldots, r+\ell\} \mid S \subseteq \{0,1,\ldots,\ell-1\}\},$$

respectively. Both states go to a state containing $k-1$ by every string over $\{a,b\}$ of length greater than $k-r-2$. As a corollary, we get (iii) and (iv). Next, it follows that

$$|\mathcal{R}_{1,r}| = |\mathcal{R}_{0,r}| = 1+2+4+\cdots+2^{k-r-2} = 2^{k-r-1}-1,$$

which proves the lemma. □

Using the results of the above lemma we are now able to give, for each β between 2^{k-1} and 2^k, a construction of a k-state nfa that needs β deterministic states.

Lemma 5. *For all integers k and β such that $2^{k-1}+1 \leqslant \beta \leqslant 2^k$, there exists a minimal ternary k-state nfa $B_{k,\beta}$ whose equivalent minimal dfa has β states.*

Proof. If $\beta = 2^{k-1}+1$, we set $B_{k,\beta} = A_k$, where A_k is the nfa defined on page 305. The nfa is minimal since the set of pairs
$\mathcal{A} = \{(\varepsilon, ba^{k-2}ca\} \cup \{(ba^j, a^{k-2-j}ca) \mid 0 \leqslant j \leqslant k-2\}$
is a fooling set of size k for the language $L(A_k)$. Next, notice that the empty string is accepted by the nfa only from state $k-1$, while for each state i which is less than $k-1$, the string $a^{k-2-i}c$ is accepted only from state i. This means that all reachable states of the corresponding subset automaton are pairwise inequivalent. By Lemma 3, the subset automaton has $2^{k-1}+1$ reachable states.

If $2^{k-1}+1 < \beta \leqslant 2^k$, then $\beta = 2^{k-1}+1+m$ for an integer m such that $1 \leqslant m \leqslant 2^{k-1}-1$. By Lemma 2, one of the following three cases holds for m:

$$m = 2^{k-1}-1 \tag{1}$$

$$m = (2^{k_1}-1)+(2^{k_2}-1)+\cdots+(2^{k_{\ell-1}}-1)+(2^{k_\ell}-1) \tag{2}$$

$$m = (2^{k_1}-1)+(2^{k_2}-1)+\cdots+(2^{k_{\ell-1}}-1)+2\cdot(2^{k_\ell}-1) \tag{3}$$

where $1 \leqslant \ell \leqslant k-2$, and $k-2 \geqslant k_1 > k_2 > \cdots > k_\ell \geqslant 1$.

Construct a k-state nfa $B_{k,\beta} = (Q_k, \{a,b,c\}, \delta_B, k-1, \{k-1\})$ from the nfa A_k by adding transitions on input c depending on m as follows:

- In the case of (1), add the transition on c from state 0 to $\{0\}$.
- In the case of (2), add transitions on c from state $i-1$ to $\{1,2,\ldots,k-k_i-1\}$ for all $i = 1,2,\ldots,\ell$.
- In the case of (3), add the same transitions as in the case of (2), and, moreover, add also the transition on c from state ℓ to $\{0,1,2,\ldots,k-k_\ell-1\}$. If $\ell = k-2$, that is, if $m = (2^{k-2}-1)+(2^{k-3}-1)+\cdots+3+1+1$, then the transition on c from $k-2$ to $k-1$ is replaced with the transition on c from $k-2$ to $\{0,1,\ldots,k-2\}$.

Notice that in all these cases the set $\mathcal{A}$ is a fooling set for the language $L(B_{k,\beta})$, and so the nfa is minimal.

Next, in all cases, the empty string is accepted only from state $k-1$, and for each state i which is less than $k-1$, the string $a^{k-2-i}c$ is accepted only from state i, except for the case of $m = (2^{k-2} - 1) + (2^{k-3} - 1) + \cdots + 3 + 1 + 1$, when the string $a^{k-2-i}ca^{n-2}ca$ is accepted only from state i (notice that in this case, only state $k-2$ goes to state 0 on input c). This means that all reachable states of the corresponding subset automata are pairwise inequivalent.

Set $r_j = k - k_j - 1$ $(1 \leqslant j \leqslant \ell + 1)$ and let $\mathcal{R}$ denote the system consisting of the empty set and all subsets of $\{0, 1, \ldots, k-1\}$ containing state $k-1$. Let us show that the following systems $\mathcal{S}_1$, $\mathcal{S}_2$, and $\mathcal{S}_3$ consist of all reachable subsets of the subset automaton $B'_{k,\beta}$ corresponding to the nfa $B_{k,\beta}$ in cases (1), (2), and (3), respectively (remind that the systems $\mathcal{R}_{0,r}$ and $\mathcal{R}_{1,r}$ are defined before Lemma 4 on page 306):

$$\begin{aligned}
\mathcal{S}_1 &= \mathcal{R} \cup \mathcal{R}_{0,0}, \\
\mathcal{S}_2 &= \mathcal{R} \cup \mathcal{R}_{1,r_1} \cup \mathcal{R}_{1,r_2} \cup \cdots \cup \mathcal{R}_{1,r_\ell}, \\
\mathcal{S}_3 &= \mathcal{R} \cup \mathcal{R}_{1,r_1} \cup \mathcal{R}_{1,r_2} \cup \cdots \cup \mathcal{R}_{1,r_\ell} \cup \mathcal{R}_{0,r_\ell}.
\end{aligned}$$

The empty set is reachable since no transitions on symbol c are defined in the initial state $k-1$. All subsets containing state $k-1$ are reachable by Lemma 3. In the case of (1), state $\{0, k-1\}$ goes to state 0 by c, and then all subsets in $\mathcal{R}_{0,0}$ are reachable from state 0 by strings over $\{a, b\}$. In the case of (2), each state $\{i-1, k-1\}$, $1 \leqslant i \leqslant \ell$, goes by c to state $\{1, 2, \ldots, k - k_i - 1\}$, that is to state $\{1, 2, \ldots, r_i\}$, from which all subsets in $\mathcal{R}_{1,r_i}$ can be reached by strings over $\{a, b\}$. Similarly, in the case of (3), all subsets in $\mathcal{R}_{1,r_i}$, $1 \leqslant i \leqslant \ell$, are reachable, and, moreover, in this case, state $\{\ell, k-1\}$ goes to state $\{0, 1, 2, \ldots, r_\ell\}$, and so all subsets in $\mathcal{R}_{0,r_\ell}$ are reachable as well.

To show that no other subsets are reachable it is enough to prove that each set S in the system $\mathcal{S}_1$ $(\mathcal{S}_2, \mathcal{S}_3)$ goes to a subset in this system by each of the symbols a, b, c. This is quite straightforward for symbols a and b. In the case of symbol c, it is important to notice that we have $\delta_B(k-2, c) = \{k-1\}$ or $\delta_B(k-2, c) = \{0, 1, \ldots, k-2\}$, and $\delta_B(0, c) \subset \delta_B(1, c) \subset \cdots \subset \delta_B(\ell - 1, c)$, and in the case of (3), also $\delta_B(\ell - 1, c) \subset \delta_B(\ell, c)$. In the other states, transitions on c are not defined. It follows that for each subset S of Q_k, the set $\delta_B(S, c)$ is either empty, or contains state $k-1$, or is equal to $\delta_B(j, c)$ for the greatest integer j in $\{0, 1, \ldots, \ell - 1\}$ (in $\{0, 1, \ldots, \ell\}$, respectively) that is in S. In all three cases, the set $\delta(S, c)$ is in $\mathcal{S}_1$ $(\mathcal{S}_2, \mathcal{S}_3$, respectively).

By Lemma 4, the systems $\mathcal{R}_{1,r_1}, \mathcal{R}_{1,r_2}, \ldots, \mathcal{R}_{1,r_\ell}$, and $\mathcal{R}_{0,r_\ell}$ are pairwise disjoint and

$$|\mathcal{R}_{1,r_i}| = |\mathcal{R}_{0,r_i}| = 2^{k-r_i-1} - 1 = 2^{k-(k-k_i-1)-1} - 1 = 2^{k_i} - 1.$$

Hence the subset automaton corresponding to the nfa $B_{k,\beta}$ has $1 + 2^{k-1} + m$ reachable and pairwise inequivalent states, and since $1 + 2^{k-1} + m = \beta$, our proof is complete. □

We are now able to prove our main result.

Theorem 1. *For all integers n and α with $n \leqslant \alpha \leqslant 2^n$, there exists a minimal nondeterministic finite automaton of n states with a three-letter input alphabet whose equivalent minimal deterministic finite automaton has exactly α states.*

Proof. If $\alpha > 2^{n-1}$, then take the n-state nfa $B_{n,\alpha}$ given by Lemma 5. Otherwise, let us find an integer k, $1 \leqslant k \leqslant n-1$, such that $n-(k-1)+2^{k-1} \leqslant \alpha < n-k+2^k$. Then

$$\alpha = n - k + 1 + 2^{k-1} + m$$

for some integer m such that $1 \leqslant m < 2^{k-1}$. Denote $\beta = 1 + 2^{k-1} + m$, hence

$$\alpha = n - k + \beta.$$

Construct an n-state nfa $C_{n,\alpha}$ from the k-state nfa $B_{k,\beta}$ described in the proof of Lemma 5 in the following way. First, add new states $k, k+1, \ldots, n-1$. State $n-1$ will be the initial state, and state $k-1$ will be the sole accepting state of the nfa $C_{n,\alpha}$. Add transitions on symbol b from state j to state $j+1$ for each state j which is greater than $k-1$. Add the transitions on symbols a and c from state k to itself. The nfa $C_{n,\alpha}$ for $\alpha = n-k+1+2^{k-1}$ is shown in Fig. 4.

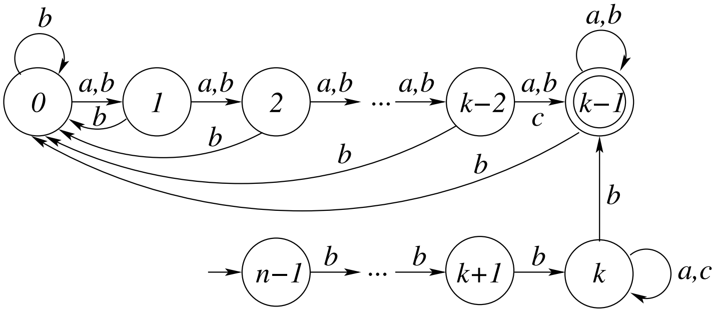


Fig. 4. The nondeterministic finite automaton $C_{n,n-k+1+2^{k-1}}$

Notice that the set of pairs of strings

$$\big\{(b^j, b^{n-k-3-j}a^k cbba^{k-2}ca) \mid 0 \leqslant j \leqslant n-k-3\big\}\cup$$
$$\big\{(b^{n-k-3}a^k cb, ba^{k-2}ca)\big\}\cup \big\{(b^{n-k-3}a^k cbba^i, a^{k-2-i}ca) \mid 0 \leqslant i \leqslant k-2\big\}$$

is a fooling set of size n for the language $L(C_{n,\alpha})$. Hence the nfa $C_{n,\alpha}$ is minimal.

Consider the corresponding subset automaton $C'_{n,\alpha}$. Each of the singletons $\{n-1\}, \{n-2\}, \ldots, \{k-1\}$ is reachable from the initial state $\{n-1\}$ by a string in b^*. Then, all β reachable states of the subset automaton $B'_{k,\beta}$ are also reachable in the dfa $C'_{n,\alpha}$. Moreover, no other subset of $\{0, 1, \ldots, n-1\}$ is reachable. Hence the dfa $C'_{n,\alpha}$ has $n-k+\beta$ reachable states. Since $n-k+\beta = \alpha$, it is enough to show that no two different reachable states are equivalent. Two different subsets of $\{0, 1, \ldots, k-1\}$ can be distinguished by the same string as in the dfa $B'_{k,\beta}$. If $k \leqslant i < j \leqslant n-1$, then the string b^{i-k-1} distinguishes states $\{i\}$ and $\{j\}$. If S is a subset of $\{0, 1, \ldots, k-1\}$ and $k \leqslant i \leqslant n-1$, then the string $b^{i-k}a^k cb$ is accepted from state $\{i\}$ but not from state S. This concludes our proof. □

4 Conclusion

In this paper, we have shown that there are no magic numbers in the ternary case. We have described a minimal n-state nondeterministic finite automaton with a three-letter input alphabet that requires α deterministic states for all integers n and α satisfying that $n \leqslant \alpha \leqslant 2^n$.

The question of whether there are some magic numbers in the binary case remains open. However, after investigating the ternary case, we strongly conjecture that each value in the range from n to 2^n can be reached as the size of the minimal deterministic finite automaton equivalent to a minimal *binary* nondeterministic finite automaton of n-states.

Acknowledgements

I would like to thank Alexander Szabari for his help with the computational verification of some conjectures. I am very grateful to anonymous referees of DLT for their valuable corrections and suggestions. My thanks also go to Mikhail Volkov for his careful reading of the manuscript and useful remarks that have helped me improve the presentation of this paper.

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); Erratum: http://clam.rutgers.edu/~birget/poWordsERR.ps
4. Chrobak, M.: Finite automata and unary languages. Theoret. Comput. Sci. 47, 149–158 (1986); Erratum: Theoret. Comput. Sci. 302, 497–498 (2003)
5. Dassow, J., Stiebe, R.: Nonterminal complexity of some operations on context-free languages. In: Geffert, V., Pighizzini, G. (eds.) 9th International Workshop on Descriptional Complexity of Formal Systems, pp. 162–169. P. J. Šafárik University of Košice, Slovakia (2007)
6. 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)
7. Geffert, V.: Magic numbers in the state hierarchy of finite automata. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 412–423. Springer, Heidelberg (2006)
8. Glaister, I., Shallit, J.: A lower bound technique for the size of nondeterministic finite automata. Inform. Process. Lett. 59, 75–77 (1996)
9. Hromkovič, J.: Communication Complexity and Parallel Computing. Springer, Heidelberg (1997)

10. 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); Preliminary version in: Bozapalidis, S. (ed.) 3rd International Conference on Developments in Language Theory. Aristotle University of Thessaloniki (1997)
11. Iwama, K., Matsuura, A., Paterson, M.: A family of NFAs which need $2^{\mathrm{n}} - \alpha$ deterministic states. Theoret. Comput. Sci. 301, 451–462 (2003)
12. 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)
13. 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. 431–442. Springer, Heidelberg (2008)
14. Jirásek, J., Jirásková, G., Szabari, A.: Deterministic blow-ups of minimal nondeterministic finite automata over a fixed alphabet. Internat. J. Found. Comput. Sci. 16, 511–529 (2005)
15. Lyubich, Yu.I.: Estimates for optimal determinization of nondeterministic autonomous automata. Sibirskii Matematicheskii Zhurnal 5, 337–355 (1964)
16. Matsuura, A., Saito, Y.: Equivalent transformation of minimal finite automata over a two-letter alphabet. In: Campeanu, C., Pighizzini, G. (eds.) 10th International Workshop on Descriptional Complexity of Formal Systems, pp. 224–232. University of Prince Edward Island, Canada (2008)
17. Rabin, M., Scott, D.: Finite automata and their decision problems. IBM Res. Develop. 3, 114–129 (1959)
18. Sipser, M.: Introduction to the theory of computation. PWS Publishing Company, Boston (1997)
19. Yu, S.: Regular languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, ch. 2, vol. I, pp. 41–110. Springer, Heidelberg (1997)
20. Zijl, L.: Magic numbers for symmetric difference NFAs. Internat. J. Found. Computut. Sci. 16, 1027–1038 (2005)

The Pumping Lemma for Well-Nested Multiple Context-Free Languages

Makoto Kanazawa[⋆]

National Institute of Informatics

Abstract. Seki et al. (1991) proved a rather weak pumping lemma for multiple context-free languages, which says that any infinite m-multiple context-free language contains a string that is pumpable at some $2m$ substrings. We prove a pumping lemma of the usual universal form for the subclass consisting of *well-nested* multiple context-free languages. This is the same class of languages generated by non-duplicating macro grammars and by coupled-context-free grammars.

1 Introduction

Call a string z in a language L *k-pumpable* (in L) if there are strings $u_0, \dots, u_k$ and $v_1, \dots, v_k$ that satisfy the following conditions:

$$z = u_0 v_1 u_1 v_2 u_2 \dots u_{k-1} v_k u_k,$$
$$v_1 v_2 \dots v_k \neq \epsilon,$$
$$u_0 v_1^i u_1 v_2^i u_2 \dots u_{k-1} v_k^i u_k \in L \quad \text{for every } i \geq 0.$$

In their paper introducing *multiple context-free grammars*, Seki et al. (1991) proved a rather weak pumping lemma for multiple context-free languages, which says that if L is an infinite m-MCFL, then *some* string in L is $2m$-pumpable. Despite its peculiarly weak existential form, this lemma is quite useful; for example, it can be used to show that the language $\{\, \mathtt{a}_1^n \dots \mathtt{a}_{2m+1}^n \mid n \geq 0 \,\}$ over the alphabet $\{\mathtt{a}_1, \dots, \mathtt{a}_{2m+1}\}$ separates the $(m+1)$-MCFLs from m-MCFLs.

As it happens, Seki et al.'s (1991) proof of their lemma was quite complex, and by some quirk of fate a number of people were led to believe erroneously that a pumping lemma of the more usual universal form has been established for MCFLs: if L is an m-MCFL, *all but finitely many* strings in L are $2m$-pumpable. Radzinski (1991) appealed to it in his attempt to prove that the language $\{\, \mathtt{ab}^{k_1}\mathtt{ab}^{k_2} \dots \mathtt{ab}^{k_n} \mid n \geq 1, k_1 > k_2 > \dots > k_n > 0 \,\}$ is not an MCFL.[1] As a matter of fact, it remains an open question to this day whether the universal pumping lemma holds of all m-MCFLs (Kanazawa and Salvati 2007).

[⋆] I am grateful to Shoh Yamashita and Makoto Tatsuta for helpful discussions.

[1] This language, considered by Radzinski (1991) in connection with the system of Chinese number names, was shown to be non-semilinear by Michaelis and Kracht (1997), so Radzinski's (1991) claim that it is not an MCFL was correct, even though his appeal to the universal pumping lemma for general MCFLs was not justified.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 312–325, 2009.

This question is especially interesting in view of the fact that the class of m-MCFLs is captured by a large number of different formalisms, including among many others *hyperedge replacement grammars* (Engelfriet and Heyker 1991) and *deterministic tree-walking transducers* (Weir 1992).

In this paper, we show that a universal pumping lemma holds for the subclass of MCFLs generated by *well-nested* multiple context-free grammars. It is not difficult to prove that well-nested MCFGs are equivalent to *coupled-context-free grammars* (Hotz and Pitsch 1996) and to *non-duplicating macro grammars* (Fischer 1968).

The principal difficulty in proving a pumping lemma for MCFLs lies in the fact that pumpability of a derivation tree does not translate into pumpability of its string yield. Consider a derivation tree that contains inside it a "pump", i.e., a tree whose frontier consists of terminal nodes plus a single node marked by the same nonterminal as the root. In the case of a CFG, the function of a pump is to take a string and wrap two strings around it; its contribution to the string yield of the whole derivation tree is simply insertion of two substrings into the yield. Iterating the pump i times in the derivation tree results in insertion of i consecutive copies of those substrings, leading to 2-pumpability. In the case of an m-MCFG, in contrast, the function of a pump is to take a k-tuple of strings ($k \leq m$) and return another k-tuple. Each component of the original k-tuple appears somewhere in the resulting k-tuple, but it may move into a different component. If that happens, the effect of the presence of a pump is not merely insertion of $2k$ substrings, but also involves shuffling of the substrings contributed by the subtree below the pump.

Call a pump of an MCFG an *even k-pump* if it maps a k-tuple of strings $(x_1, \ldots, x_k)$ to a k-tuple of the form $(v_1 x_1 v_2, \ldots, v_{2k-1} x_k v_{2k})$. The presence of an even k-pump inside a derivation tree leads to $2k$-pumpability of its string yield, but it is not generally true, even for well-nested MCFGs, that all but finitely many derivation trees contain an even pump. Our strategy for proving the universal pumping lemma for well-nested m-MCFGs is as follows. We show by a series of transformations that for every well-nested m-MCFG G, the set of strings that are the yields of derivation trees of G without even m-pumps is the language of some well-nested $(m-1)$-MCFG. Since well-nested 1-MCFGs are just CFGs, this implies, by induction, that all but finitely many strings in the language of G are $2m$-pumpable.

2 Preliminaries

For $m, n \in \mathbb{N}$ (the set of natural numbers), we use the notation $[m, n]$ to denote $\{ i \in \mathbb{N} \mid m \leq i \leq n \}$.

A *ranked alphabet* is a finite set $\Delta = \bigcup_{n \in \mathbb{N}} \Delta^{(n)}$ such that $\Delta^{(i)} \cap \Delta^{(j)} = \varnothing$ for $i \neq j$. The rank of $f \in \Delta$ is the unique r such that $f \in \Delta^{(r)}$. The set $\mathbb{T}_\Delta$ of *trees* over a ranked alphabet Δ is the smallest set closed under the rule: $f \in \Delta^{(n)}$ and $T_1, \ldots, T_n \in \mathbb{T}_\Delta$ imply $(f T_1 \ldots T_n) \in \mathbb{T}_\Delta$. We also use trees with holes, which are represented by trees over a ranked alphabet Δ augmented with a set

$\mathbf{Y}$ of variables. The notation $\mathbb{T}_\Delta(\mathbf{Y})$ denotes the set of trees over $\Delta \cup \mathbf{Y}$, where the variables in $\mathbf{Y}$ have rank 0. We use expressions like $T[\mathbf{y}_1, \dots, \mathbf{y}_m]$ to denote trees in $\mathbb{T}_\Delta(\mathbf{Y})$ whose variables are among $\mathbf{y}_1, \dots, \mathbf{y}_m$, and write $T[T_1, \dots, T_m]$ for the result of substituting $T_1, \dots, T_m$ for $\mathbf{y}_1, \dots, \mathbf{y}_m$ in $T[\mathbf{y}_1, \dots, \mathbf{y}_m]$. A tree $T \in \mathbb{T}_\Delta(\mathbf{Y})$ is a *simple tree* if each variable in $\mathbf{Y}$ occurs in T at most once.

We assume that the reader is familiar with the notion of *recognizable* set of trees (see Comon et al. 2007 or Gecseg and Steinby 1997). The family of recognizable sets is closed under Boolean operations.

3 Multiple Context-Free Grammars

A *multiple context-free grammar* (Seki et al. 1991) is a context-free grammar on tuples of strings, and is a special kind of *parallel multiple context-free grammar*, which in turn is a special kind of *elementary formal system* (Smullyan 1961, Groenink 1997), a logic program on strings. We use the notation of elementary formal systems, rather than Seki et al.'s (1991), to represent rules of MCFGs.

Let N be a ranked alphabet and Σ be an unranked alphabet. We assume that we are given a fixed infinite supply X of variables ranging over strings. An expression of the form $B(t_1, \dots, t_r)$, where $B \in N^{(r)}$ and $t_1, \dots, t_r$ are strings over $\Sigma \cup X$, is called an *atom* over N, Σ. A *rule* over N, Σ is an expression

$$B(t_1, \dots, t_r) :\!- \, B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n}),$$

where

1. $n \geq 0$,
2. $B \in N^{(r)}$ and $B_i \in N^{(r_i)}$ for all $i \in [1, n]$,
3. $x_{i,j}$ are pairwise distinct variables in X,
4. $t_1, \dots, t_r$ are strings over $\Sigma \cup \{\, x_{i,j} \mid i \in [1, n], j \in [1, r_i] \,\}$ such that each $x_{i,j}$ occurs at most once in $t_1 \dots t_r$.

In a rule

$$B(t_1, \dots, t_r) :\!- \, B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n}),$$

the atom to the left of :– is called the *head* of the rule, and the sequence of atoms to the right of :– is called the *body*. Each atom in the body is called a *subgoal*. The symbol :– is omitted from a rule whose body is empty. Such a rule is called a *terminating rule*.

A *multiple context-free grammar* (MCFG) is a 4-tuple $G = (N, \Sigma, P, S)$, where

1. N is a ranked alphabet of *nonterminals*,
2. Σ is an (unranked) alphabet of *terminals*, disjoint from N,
3. P is a finite set of *rules* over N, Σ, and
4. $S \in N^{(1)}$.

We say that G is an m-MCFG if the rank of nonterminals does not exceed m.[2]

[2] The rank of a nonterminal is called its *dimension* by Seki et al. (1991).

The language of G is $L(G) = \{ w \in \Sigma^* \mid \vdash_G S(w) \}$, where $\vdash_G$ is defined by the following inference schema:

$$\frac{\vdash_G B_1(w_{1,1}, \ldots, w_{1,r_1}) \quad \ldots \quad \vdash_G B_n(w_{n,1}, \ldots, w_{n,r_n})}{\vdash_G B(t_1, \ldots, t_r)\sigma}$$

where $w_{i,j} \in \Sigma^*$, $B(t_1, \ldots, t_r) :\!- B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n})$ is a rule in P, and σ is the substitution that sends $x_{i,j}$ to $w_{i,j}$. If G is an m-MCFG, $L(G)$ is called an *m-MCFL*.

It is also convenient to extend the definition of $\vdash_G$ as follows:

$$\frac{}{B(x_1, \ldots, x_r) \vdash_G B(x_1, \ldots, x_r)}$$

$$\frac{\Gamma_1 \vdash_G B_1(x_{1,1}, \ldots, x_{1,r_1})\sigma \quad \ldots \quad \Gamma_n \vdash_G B_n(x_{n,1}, \ldots, x_{n,r_n})\sigma}{\Gamma_1, \ldots, \Gamma_n \vdash_G B(t_1, \ldots, t_r)\sigma}$$

In the second schema, $B(t_1, \ldots, t_n) :\!- B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n})$ is a rule in P, and $\Gamma_1, \ldots, \Gamma_n$ is a sequence of atoms with no repeated variables.

We call an MCFG $G = (N, \Sigma, P, S)$ *reduced* if the following conditions hold for each nonterminal $B \in N^{(r)}$:

- $\vdash_G B(w_1, \ldots, w_r)$ for some $w_1, \ldots, w_r \in \Sigma^*$, and
- $B(x_1, \ldots, x_r) \vdash_G S(t)$ for some $t \in (\Sigma \cup \{x_1, \ldots, x_r\})^*$.

The following lemma can be shown by a familiar method:

Lemma 1. *For every m-MCFG $G = (N, \Sigma, P, S)$ such that $L(G) \neq \varnothing$, there exists a reduced m-MCFG $G' = (N', \Sigma, P', S)$ such that $L(G) = L(G')$, $N' \subseteq N$, and $P' \subseteq P$.*

A rule is *non-deleting* if it satisfies the strengthened form of the fourth condition on rules:

4′. $t_1, \ldots, t_r$ are strings over $\Sigma \cup \{ x_{i,j} \mid i \in [1, n], j \in [1, r_i] \}$ such that each $x_{i,j}$ occurs exactly once in $t_1 \ldots t_r$.

A *non-deleting* MCFG is one whose rules are all non-deleting. Non-deleting (m-)MCFGs are also known as *(string-based) (m-)linear context-free rewriting systems* (LCFRSs) (Vijay-Shanker et al. 1987, Weir 1988). It is known that for every m-MCFG G, there is a non-deleting m-MCFG G' such that $L(G) = L(G')$ (Seki et al. 1991).

We call a rule $B(t_1, \ldots, t_r) :\!- B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n})$ *non-permuting* if it satisfies the following condition:

- there are no i, j, k such that $1 \leq i \leq n$, $1 \leq j < k \leq r_i$, and $t_1 \ldots t_r \in (\Sigma \cup X)^* x_{i,k} (\Sigma \cup X)^* x_{i,j} (\Sigma \cup X)^*$.

A *non-permuting* MCFG is one whose rules are all non-permuting. Non-deleting non-permuting MCFGs correspond to what Villemonte de la Clergerie (2002a, 2002b) called *ordered simple RCG* and Kracht (2003) called *monotone* LCFRSs.

Theorem 2 (Kracht 2003). *For every m-MCFG G, there is a non-deleting non-permuting m-MCFG G' such that $L(G) = L(G')$.*

Example 3. The following (non-deleting non-permuting) 2-MCFG generates $\text{RESP} = \{\, \mathtt{a}_1^m \mathtt{a}_2^m \mathtt{b}_1^n \mathtt{b}_2^n \mathtt{a}_3^m \mathtt{a}_4^m \mathtt{b}_3^n \mathtt{b}_4^n \mid m, n \geq 0 \,\}$:

$$\begin{array}{ll} & S(x_1 y_1 x_2 y_2) \mathrel{:-} P(x_1, x_2), Q(y_1, y_2). \\ P(\epsilon, \epsilon). & P(\mathtt{a}_1 x_1 \mathtt{a}_2, \mathtt{a}_3 x_2 \mathtt{a}_4) \mathrel{:-} P(x_1, x_2). \\ Q(\epsilon, \epsilon). & Q(\mathtt{b}_1 y_1 \mathtt{b}_2, \mathtt{b}_3 y_2 \mathtt{b}_4) \mathrel{:-} Q(y_1, y_2). \end{array}$$

We call a rule $B(t_1, \dots, t_r) \mathrel{:-} B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n})$ *well-nested* if it is non-deleting and non-permuting and satisfies the following condition:

- there are no i, j, k, i', j', k' such that $1 \leq i, i' \leq n$, $i \neq i'$, $1 \leq j < k \leq r_i$, $1 \leq j' < k' \leq r_{i'}$, and $t_1 \dots t_r \in (\Sigma \cup X)^* x_{ij} (\Sigma \cup X)^* x_{i'j'} (\Sigma \cup X)^* x_{ik} (\Sigma \cup X)^* x_{i'k'} (\Sigma \cup X)^*$.

We say that an MCFG G is *well-nested* if every rule of G is well-nested. A *well-nested MCFL* is the language of some well-nested MCFG. Note that the grammar in Example 3 is not well-nested, because the first rule is not well-nested.

Example 4. The following is an example of a well-nested 2-MCFG:

$$S(x_1 x_2) \mathrel{:-} A(x_1, x_2). \qquad A(\epsilon, \epsilon). \qquad A(\mathtt{a} x_1 \mathtt{b} x_2 \mathtt{c}, \mathtt{d}) \mathrel{:-} A(x_1, x_2).$$

The well-nestedness constraint has been studied by Kuhlmann and Möhl (2007a, 2007b) in the context of *dependency grammars*. Well-nested (m-)MCFGs are essentially the same as *coupled-context-free grammars* (of rank m) (Hotz and Pitsch 1996), and it can be shown that they are equivalent to *non-duplicating macro grammars* (of rank $m - 1$) (Fischer 1968).[3] It is known that well-nested 2-MCFGs are equivalent to *tree-adjoining grammars* (Joshi and Schabes 1997).

Although well-nestedness restricts the generative power of m-MCFGs for each m (Seki and Kato 2008), almost all examples of m-MCFGs that have appeared in the literature have an equivalent well-nested m'-MCFG for some $m' \geq m$. The only exception we are aware of is the 3-MCFG G_{ex} given by Michaelis (2009):

[3] The equivalence between well-nested MCFGs and coupled-context-free grammars is a special case of the equivalence between MCFGs and *local unordered scattered context grammars* (Rambow and Satta 1999). Seki and Kato (2008) prove that all non-duplicating macro grammars—which they call *variable-linear* macro grammars—have equivalent MCFGs. The MCFGs constructed by their proof are well-nested.

The languages generated by non-duplicating macro grammars are the same as the yield images of the tree languages generated by *linear context-free tree grammars* (Kepser and Mönnich 2006).

$$S(x_1x_2x_3) :\!-\; B(x_1,x_2,x_3). \qquad A(\mathtt{a},\mathtt{a},\mathtt{a}). \qquad A(\mathtt{b},\mathtt{b},\mathtt{b}).$$
$$A(x_1\mathtt{a},x_2\mathtt{a},\mathtt{a}x_3) :\!-\; A(x_1,x_2,x_3). \qquad A(x_1\mathtt{b},x_2\mathtt{b},\mathtt{b}x_2) :\!-\; A(x_1,x_2,x_3).$$
$$A(x_1y_1,y_2x_2,y_3x_3) :\!-\; B(x_1,x_2,x_3), B(y_1,y_2,y_3).$$
$$B(\epsilon,\mathtt{[]},\epsilon). \qquad B(x_1,x_2,x_3) :\!-\; A(x_1,x_2,x_3). \qquad B(x_1,\mathtt{[}x_2\mathtt{]},x_3) :\!-\; B(x_1,x_2,x_3).$$

This non-well-nested 3-MCFG generates the following language:[4]

$$\begin{aligned} L(G_{\text{ex}}) = \{\, & w_1 \dots w_n z_n w_n \dots z_1 w_1 z_0 w_n^R \dots w_1^R \mid \\ & n \geq 1,\ w_i \in \{\mathtt{a},\mathtt{b}\}^+ \text{ for } 1 \leq i \leq n, \text{ and } z_n \dots z_0 \in D^*_{\{\mathtt{[},\mathtt{]}\}} \,\}. \end{aligned}$$

According to Staudacher (1993), this language is not an indexed language, which implies that it does not have a non-duplicating macro grammar and hence is not a well-nested MCFL.

In what follows, we will only consider MCFGs that are non-deleting and non-permuting. By a "rule", we will mean a rule that is non-deleting and non-permuting.

4 Derivation Trees of Multiple Context-Free Grammars

We now give our definition of the notion of a *derivation tree* for an MCFG $G = (N, \Sigma, P, S)$. For this purpose, we view the set P of rules as a ranked alphabet. A rule π of the form

$$B(t_1,\dots,t_r) :\!-\; B_1(x_{1,1},\dots,x_{1,r_1}),\dots,B_n(x_{n,1},\dots,x_{n,r_n})$$

belongs to $P^{(n)}$ (i.e., is treated as a symbol of rank n).

Let $\mathbf{Y}$ be a countably infinite set of variables. We use boldface letters $\mathbf{y}, \mathbf{y}_i$, to represent variables in $\mathbf{Y}$, to distinguish them from variables in MCFG atoms, for which we use $x_{i,j}, z_{i,j}, y_i$, etc. We use simple trees in $\mathbb{T}_P(\mathbf{Y})$ as terms in statements of the form

$$\Gamma \vdash_G T : B(t_1,\dots,t_r),$$

where Γ is an expression of the form

$$\mathbf{y}_1 : C_1(z_{1,1},\dots,z_{1,r_1}),\dots,\mathbf{y}_p : C_p(z_{p,1},\dots,z_{p,r_p}).$$

The earlier system for deriving statements of the form $\Gamma \vdash_G B(t_1,\dots,t_r)$, where Γ is a sequence of atoms, is now augmented with trees from $\mathbb{T}_P(\mathbf{Y})$ as follows:

$$\frac{}{\mathbf{y} : B(x_1,\dots,x_n) \vdash_G \mathbf{y} : B(x_1,\dots,x_n)}$$

$$\frac{\Gamma_1 \vdash_G T_1 : B_1(x_{1,1},\dots,x_{1,r_1})\sigma \quad \dots \quad \Gamma_n \vdash_G T_n : B_n(x_{n,1},\dots,x_{n,r_n})\sigma}{\Gamma_1,\dots,\Gamma_n \vdash_G \pi T_1 \dots T_n : B(x_1,\dots,x_r)\sigma}$$

[4] Here, $D^*_{\{\mathtt{[},\mathtt{]}\}}$ refers to the (one-sided) Dyck language over a single pair of brackets $\mathtt{[},\mathtt{]}$.

In the second schema, π is the rule

$$B(x_1, \dots, x_r) \mathrel{:-} B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n}),$$

and the variables (including boldface ones) in $\Gamma_1, \dots, \Gamma_n$ have no repeated occurrences.

A tree $T \in \mathbb{T}_P$ is called a *derivation tree* if

$$\vdash_G T : B(w_1, \dots, w_r)$$

holds for some $B \in N$ and $w_1, \dots, w_r \in \Sigma^*$. The nonterminal B is called the *type* of T, and the tuple $(w_1, \dots, w_r)$ is called its *yield*. Any derivation tree T has a unique type and a unique yield, for which we write $\mathrm{type}(T)$ and $\mathrm{yield}(T)$, respectively. A derivation tree is *complete* if it is of type S. We denote the set of derivation trees of type B by $\mathcal{D}_G^B$. Note that $\mathcal{D}_G^B$ is a recognizable (in fact, local) subset of $\mathbb{T}_P$.

We call a simple tree $T \in \mathbb{T}_P(\mathbf{Y})$ a *non-terminating derivation tree* if

$$\Gamma \vdash_G T : B(t_1, \dots, t_r)$$

holds for some $\Gamma, B, t_1, \dots, t_r$. Note that derivation trees are special cases of non-terminating derivation trees. We call a non-terminating derivation tree T a *derivation tree context* if

$$\mathbf{y} : C(y_1, \dots, y_p) \vdash_G T : B(t_1, \dots, t_r)$$

holds of some $\mathbf{y} \in \mathbf{Y}$, $C \in N^{(p)}$ and $B \in N^{(r)}$.

It is easy to see that if T is a derivation tree and T' is a subtree of T, then T' is also a derivation tree. The same goes with non-terminating derivation trees.

Lemma 5. *If $\Gamma \vdash_G T : B(t_1, \dots, t_r)$ and $\mathbf{y} : B(x_1, \dots, x_r) \vdash_G U[\mathbf{y}] : C(u_1, \dots, u_p)$, then $\Gamma \vdash_G U[T] : C(u_1, \dots, u_p)\sigma$, where σ is the substitution that sends x_i to t_i for all $i \in [1, r]$.*

Lemma 6. *Let T be a derivation tree of type B with yield $(w_1, \dots, w_r)$. Suppose that T' is a subtree of T such that $\mathrm{type}(T') = C$ and $\mathrm{yield}(T') = (v_1, \dots, v_p)$. Then there is a derivation tree context $U[\mathbf{y}]$ and some $t_1, \dots, t_r$ such that:*

$$\begin{gathered} T = U[T'], \\ \mathbf{y} : C(y_1, \dots, y_p) \vdash_G U[\mathbf{y}] : B(t_1, \dots, t_r), \\ (w_1, \dots, w_r) = (t_1, \dots, t_r)\sigma, \end{gathered}$$

where σ is the substitution that maps y_i to v_i for $i \in [1, p]$.

We call a derivation tree context $U[\mathbf{y}]$ a *k-pump* if $U[\mathbf{y}] \neq \mathbf{y}$ and there exist $B \in N^{(k)}$ and $t_1, \dots, t_k \in (\Sigma \cup \{x_1, \dots, x_k\})^*$ such that

$$\mathbf{y} : B(x_1, \dots, x_k) \vdash_G U[\mathbf{y}] : B(t_1, \dots, t_k).$$

We say that a derivation tree T *contains* a k-pump $U[\mathbf{y}]$ if $T = U'[U[T']]$ for some derivation tree T' and derivation tree context $U'[\mathbf{y}]$.

A k-pump $U[\mathbf{y}]$ is *even* if there are $v_1, \dots, v_{2k} \in \Sigma^*$ such that

$$\mathbf{y} : B(x_1, \dots, x_k) \vdash_G U[\mathbf{y}] : B(v_1 x_1 v_2, \dots, v_{2k-1} x_k v_{2k}),$$

and it is a *proper* even k-pump if it moreover holds that

$$v_1 \dots v_{2k} \neq \epsilon.$$

Lemma 7. *Let G be an MCFG and T be a complete derivation tree of G with yield z. If T contains a proper even k-pump, then z is $2k$-pumpable in $L(G)$.*

Example 4 (continued). Let π_1, π_2, π_3 name the three rules of the grammar G in Example 4. We have $\mathcal{D}_G^S = \{\, T_i \mid i \geq 0 \,\}$, where

$$T_i = \pi_1 \, \underbrace{(\pi_2 \dots (\pi_2}_{i \text{ times}} \, \pi_3 \underbrace{) \dots)}_{i \text{ times}},$$

and

$$\text{yield}(T_i) = \begin{cases} \epsilon & \text{if } i = 0, \\ \mathtt{a}^{i-1}\mathtt{abc}(\mathtt{bdc})^{i-1}\mathtt{d} & \text{if } i \geq 1. \end{cases}$$

Note that the derivation tree T_1 contains an (uneven) 2-pump, but yield$(T_1) =$ $\mathtt{abcd}$ is not 4-pumpable. For $i \geq 2$, yield$(T_i) = \mathtt{a}^{i-1}\mathtt{abc}(\mathtt{bdc})^{i-1}\mathtt{d}$ is 2-pumpable, but no matter which 2-pump one picks in T_i, the occurrences of symbols that come from the 2-pump do not form contiguous substrings of yield(T_i) that can be repeated.

Lemma 8. *Let $G = (N, \Sigma, P, S)$ be an MCFG. Then the set*

$$\{\, T \in \mathcal{D}_G^S \mid T \text{ contains an even } k\text{-pump} \,\}$$

is recognizable.

Let $G = (N, \Sigma, P, S)$ and $G' = (N', \Sigma, P', S')$ be m-MCFGs. A mapping $h \colon N' \to N$ is a *homomorphism* from G' to G if the following conditions hold:

- $h(S') = S$.
- For every $B' \in N'^{(r)}$, $h(B') \in N^{(r)}$.
- For every $\pi' \in P'$ of the form

  $$B'(t_1, \dots, t_r) :\!- B'_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B'_n(x_{n,1}, \dots, x_{n,r_n}),$$

  if $B = h(B')$ and $B_i = h(B'_i)$ for $i \in [1, n]$, then

  $$B(t_1, \dots, t_r) :\!- B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n})$$

  is a rule in P. (We refer to this rule as $h(\pi')$.)

Lemma 9. *Let G and G' be MCFGs such that there is a homomorphism from G' to G. If G is well-nested, then so is G'.*

If h is a homomorphism from G' to G and $T' \in \mathcal{D}_{G'}^{B'}$, then we write $h(T')$ for the result of replacing occurrences of each rule π' in T' by $h(\pi')$. Note that $h(T')$ must be a derivation tree in $\mathcal{D}_G^{h(B')}$ that has the same yield as T'.

Lemma 10. *Let $G = (N, \Sigma, P, S)$ be an m-MCFG. If K is a non-empty recognizable subset of $\mathcal{D}_G^S$, then there are an m-MCFG $G' = (N', \Sigma, P', S')$ and a homomorphism h from G' to G such that*

1. $K = \{ h(T') \mid T \in \mathcal{D}_{G'}^{S'} \}$.
2. *If G is reduced, then G' is reduced.*

5 Unfolding

Let $G = (N, \Sigma, P, S)$ be an m-MCFG. A rule

$$B(t_1, \dots, t_r) :- B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n})$$

over N, Σ is a *derivable rule* of G if it holds that

$$B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n}) \vdash_G B(t_1, \dots, t_r).$$

We call an MCFG $G' = (N', \Sigma, P', S')$ *conservative over* $G = (N, \Sigma, P, S)$ if $N' \subseteq N$, $S' = S$, and every rule in P' is a derivable rule of G. Clearly, "is conservative over" is a transitive relation.

Lemma 11. *Let G and G' be MCFGs such that G' is conservative over G.*

1. *If G is well-nested, then so is G'.*
2. *If G' has an even m-pump, so does G.*

Let π, π' denote the following two rules:

$$\begin{aligned} &B(t_1, \dots, t_r) :- B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n}) \\ &C(u_1, \dots, u_s) :- C_1(y_{1,1}, \dots, y_{1,s_1}), \dots, C_p(y_{p,1}, \dots, y_{p,s_p}), \end{aligned}$$

where $B_i = C$ (which implies $r_i = s$). Then we denote by $\pi \circ_i \pi'$ the rule

$$\begin{aligned} B(t_1, \dots, t_r)\sigma :- {} & B_1(x_{1,1}, \dots, x_{1,r_1}), \dots, B_{i-1}(x_{i-1,1}, \dots, x_{i-1,r_{i-1}}), \\ & C_1(y_{1,1}, \dots, y_{1,s_1}), \dots, C_p(y_{p,1}, \dots, y_{p,s_p}), \\ & B_{i+1}(x_{i+1,1}, \dots, x_{i+1,r_{i+1}}), \dots, B_n(x_{n,1}, \dots, x_{n,r_n}), \end{aligned}$$

where σ is the substitution that sends $x_{i,j}$ to u_j. Note that if π and π' are rules of an MCFG G, then $\pi \circ_i \pi'$ is a derivable rule of G.

Let $G = (N, \Sigma, P, S)$ be an MCFG. Let $\pi \in P$ be as above, and let $\pi_1, \dots, \pi_k$ be all the rules in P that have B_i in their head. The result of *unfolding* π in G (at the i-th subgoal) is defined to be $G' = (N, \Sigma, P', S)$, where $P' = (P - \{\pi\}) \cup \{ \pi \circ_i \pi_j \mid j \in [1, k] \}$. Clearly, G' is conservative over G. The following is familiar from logic programming (Tamaki and Sato 1984):

Lemma 12. *Let $G = (N, \Sigma, P, S)$ be an MCFG and π be a rule in P. If G' is the result of unfolding π in G at some subgoal, then $L(G) = L(G')$.*

6 Proof of the Main Theorem

We call a rule $B(t_1, \ldots, t_m) :- B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n})$ of an m-MCFG *m-proper* if there exists an $i \in [1, n]$ such that $r_i = m$ and for all $j \in [1, m]$,

$$t_j \in (\Sigma \cup X)^* x_{i,j} (\Sigma \cup X)^*.$$

In this case we call this rule *m-proper on the i-th subgoal.*

Lemma 13. *Let G be an m-MCFG that has no even m-pump. Then there is an m-MCFG G' that satisfies the following conditions:*

- *G' is conservative over G,*
- *G' has no m-proper rules, and*
- *$L(G) = L(G')$.*

Proof. The desired grammar G' may be obtained from G by repeatedly unfolding m-proper rules. We omit the details. □

Lemma 14. *Let $m \geq 2$ and let G be a well-nested m-MCFG without m-proper rules. Then there is a well-nested $(m-1)$-MCFG G' such that $L(G) = L(G')$.*

Proof. Define the *m-degree* of a rule to be the number of subgoals whose nonterminal is of rank m if the nonterminal in the head is of rank m, and 0 otherwise. We repeatedly apply the following transformation to eliminate from G rules that have m-degree > 0. Pick a rule π with m-degree > 0, if there is one. Modulo the order of the subgoals, π is of the following form:

$$B(t_1, \ldots, t_m) :- C(y_1, \ldots, y_m), B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n}).$$

Let $f \colon \{1, \ldots, m\} \to \{1, \ldots, m\}$ be the function such that y_i occurs in $t_{f(i)}$ for all $i \in [1, m]$. Since π is not m-proper, at least one of the following possibilities must obtain:

Case 1. Either $1 < f(1)$ or $f(m) < m$. Suppose $t_{f(1)} = u y_1 v$ and $t_{f(m)} = u' y_m v'$. Since π is well-nested,

$$B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n})$$

can be partitioned into Γ_1, Γ_2 so that $x_{i,j}$ occurs in

$$t_1 \ldots t_{f(1)-1} u v' t_{f(m)+1} \ldots t_m$$

if and only if $B_i(x_{i,1}, \ldots, x_{i,r_i})$ is in Γ_1. Let $p = f(m) - f(1) + 1$, and let D be a new nonterminal of rank p. Note $p < m$.

Case 1a. $f(1) < f(m)$. We replace π by the following two well-nested rules:

$$B(t_1, \ldots, t_{f(1)-1}, u z_1, z_2, \ldots, z_{p-1}, z_p v', t_{f(m)+1}, \ldots, t_m) :- \Gamma_1, D(z_1, \ldots, z_p).$$
$$D(y_1 v, t_{f(1)+1}, \ldots, t_{f(m)-1}, u' y_m) :- C(y_1, \ldots, y_m), \Gamma_2.$$

Case 1b. $f(1) = f(m)$. Then $t_{f(1)} = t_{f(m)} = uy_1wy_mv'$ for some w. We replace π by the following two well-nested rules:

$$B(t_1, \ldots, t_{f(1)-1}, uz_1v', t_{f(m)+1}, \ldots, t_m) :\!-\, \mathit{\Gamma}_1, D(z_1).$$
$$D(y_1wy_m) :\!-\, C(y_1, \ldots, y_m), \mathit{\Gamma}_2.$$

Case 2. There is a $k \in [1, m-1]$ such that $f(k+1) - f(k) > 1$. Suppose $t_{f(k)} = uy_kv$ and $t_{f(k+1)} = u'y_{k+1}v'$. Since π is well-nested,

$$B_1(x_{1,1}, \ldots, x_{1,r_1}), \ldots, B_n(x_{n,1}, \ldots, x_{n,r_n})$$

can be partitioned into $\mathit{\Gamma}_1, \mathit{\Gamma}_2$ so that $x_{i,j}$ occurs in

$$vt_{f(k)+1} \ldots t_{f(k+1)-1}u'$$

if and only if $B_i(x_{i,1}, \ldots, x_{i,r_i})$ is in $\mathit{\Gamma}_1$. Let $p = f(k) + (m - f(k+1) + 1)$, and let D be a new nonterminal of rank p. Note $p < m$. We replace π by the following two well-nested rules:

$$B(z_1, \ldots, z_{f(k)-1}, z_{f(k)}v, t_{f(k)+1}, \ldots, t_{f(k+1)-1}, u'z_{f(k)+1}, z_{f(k)+2}, \ldots, z_p)$$
$$:\!-\, D(z_1, \ldots, z_p), \mathit{\Gamma}_1.$$
$$D(t_1, \ldots, t_{f(k)-1}, uy_k, y_{k+1}v', t_{f(k+1)+1}, \ldots, t_m) :\!-\, C(y_1, \ldots, y_m), \mathit{\Gamma}_2.$$

In all cases, π is replaced by two new rules, and the m-degree of the first rule is less than that of π and the m-degree of the second rule is 0 (since the rank of D is $p < m$), so the transformation reduces the sum of the m-degrees of the rules. It is also clear that the first rule is not m-proper, so the grammar continues to be without m-proper rules. The generated language remains the same because the original grammar can be obtained from the new grammar by unfolding (Lemma 12).

The process of repeatedly applying this transformation must terminate after a finite number of steps, and every rule in the final grammar has rank m nonterminals only in the head or only in the body (if any).

We can now eliminate all occurrences of rank m nonterminals in rule bodies by unfolding. If a rule π has a rank m nonterminal C in the i-th subgoal, we unfold π at that subgoal. Since any rule π' that has C in the head has no rank m nonterminal in the body, $\pi \circ_i \pi'$ has one fewer rank m nonterminals in the body than π does. Thus, we can repeatedly apply this transformation, which will terminate in a finite number of steps.

After this procedure, rank m nonterminals become useless, and we can simply delete rules with rank m nonterminals in the head to obtain an $(m-1)$-MCFG. □

Example 4 (continued). Applying the procedure of Lemma 14 to the grammar G in Example 4 gives the following 1-MCFG:

$$S(\epsilon). \qquad S(\mathtt{a}z\mathtt{cd}) :\!-\, D(z). \qquad D(\mathtt{b}). \qquad D(\mathtt{a}z\mathtt{cbd}) :\!-\, D(z).$$

Theorem 15. *Let $m \geq 1$. For every well-nested m-MCFG G, all but finitely many strings $z \in L(G)$ are $2m$-pumpable in $L(G)$.*

Proof. Induction on m. The case $m = 1$ is just the pumping lemma for context-free languages. Let $m \geq 2$ and assume that the theorem holds for $m - 1$.

Let $G = (N, \Sigma, P, S)$ be a well-nested m-MCFG. Without loss of generality, we can assume that G is reduced. Let

$$K = \{\, T \in \mathcal{D}_G^S \mid T \text{ does not contain an even } m\text{-pump} \,\}.$$

By Lemma 8, K is a recognizable subset of $\mathcal{D}_G^S$. By Lemma 10, there is a reduced well-nested m-MCFG $G' = (N', \Sigma, P', S')$ such that $L(G') = \{\, w \in \Sigma^* \mid\ \vdash_G T : S(w)$ for some $T \in K \,\}$ and no derivation tree in $\mathcal{D}_{G'}^{S'}$ contains an even m-pump. Since G' is reduced, it follows that G' has no even m-pump. By Lemmas 13 and 14, there is a well-nested $(m - 1)$-MCFG G'' such that $L(G') = L(G'')$. By induction hypothesis, there is a number p such that all strings z in $L(G'')$ with $|z| \geq p$ are $2(m - 1)$-pumpable.

Now assume $z \in L(G)$ and $|z| \geq p$. We show that z is $2m$-pumpable. If $z \in L(G'')$, then z is $2(m - 1)$-pumpable, so a fortiori z is $2m$-pumpable. Now suppose $z \notin L(G'')$ and consider a smallest complete derivation tree T of G with yield z. Since $z \notin L(G'')$, T contains an even m-pump $U[\mathbf{y}]$:

$$T = U'[U[T']].$$

Because of the minimality of T, the even m-pump $U[\mathbf{y}]$ must be proper (otherwise $U'[T']$ is a smaller complete derivation tree for z). By Lemma 7, we conclude that z is $2m$-pumpable. □

Example 16. Let $D^*_{\{a,\bar{a}\}}$ be the Dyck language over $\{a, \bar{a}\}$ generated by the following context-free grammar:

$$S \to \epsilon \mid TS \qquad T \to aS\bar{a}$$

Define Shuffle$_3(L_1, L_2, L_3)$ to be

$$\{\, u_1 v_1 w_1 \dots u_n v_n w_n \mid n \geq 1, u_1 \dots u_n \in L_1, v_1 \dots v_n \in L_2, w_1 \dots w_n \in L_3 \,\},$$

and consider the language $L = \text{Shuffle}_3(D^*_{\{\mathtt{a},\bar{\mathtt{a}}\}}, D^*_{\{\mathtt{b},\bar{\mathtt{b}}\}}, D^*_{\{\mathtt{c},\bar{\mathtt{c}}\}})$. Note that L is semiliniear and satisfies Seki et al.'s pumping condition for 3-MCFLs. We do not know whether L is a 3-MCFL, but we can use Theorem 15 to show that L is not a *well-nested* 3-MCFL. Suppose that L is a well-nested 3-MCFL. Let

$$\begin{aligned} K &= L \cap \mathtt{a}^*(\bar{\mathtt{a}}\mathtt{b})^*(\bar{\mathtt{b}}\mathtt{c})^*(\bar{\mathtt{c}}\mathtt{a})^*(\bar{\mathtt{a}}\mathtt{b})^*(\bar{\mathtt{b}}\mathtt{c})^*\bar{\mathtt{c}}^* \\ &= \{\, \mathtt{a}^i(\bar{\mathtt{a}}\mathtt{b})^j(\bar{\mathtt{b}}\mathtt{c})^k(\bar{\mathtt{c}}\mathtt{a})^l(\bar{\mathtt{a}}\mathtt{b})^m(\bar{\mathtt{b}}\mathtt{c})^n\bar{\mathtt{c}}^q \mid \\ &\qquad i \geq j \geq k \geq l \leq m \leq n \leq q = i \text{ and } i + l = j + m = k + n \,\}. \end{aligned}$$

From known facts about equivalent formalisms (Fischer 1968, Kepser and Mönnich 2006, Seki and Kato 2008), the class of well-nested m-MCFLs is closed under intersection with regular sets, so K must be a well-nested 3-MCFL. Note that K still satisfies Seki et al.'s pumping condition for 3-MCFLs, and is also semilinear. By Theorem 15, there is a number p such that all strings in K of length $\geq p$ are 6-pumpable. Take

$$w = \mathtt{a}^p(\bar{\mathtt{a}}\mathtt{b})^p(\bar{\mathtt{b}}\mathtt{c})^p(\bar{\mathtt{c}}\mathtt{a})^p(\bar{\mathtt{a}}\mathtt{b})^p(\bar{\mathtt{b}}\mathtt{c})^p\bar{\mathtt{c}}^p \in K,$$

which must have six substrings that can be pumped up and down. It is not hard to see that each of the six substrings must lie entirely inside one of the seven intervals $[pi + 1, p(i + 1)]$ consisting of the $(pi + 1)$-th through the $p(i + 1)$-th symbols of w $(i = 0, \dots, 6)$, and yet each of the seven intervals must contain one of the six substrings, a contradiction.

7 Conclusion

We have proved a pumping lemma for *well-nested* m-MCFGs, which, unlike Seki et al.'s (1991) pumping lemma for general MCFGs, has the usual universal form. The special case of this for $m = 2$ is already known (Palis and Shende 1995), but the result is new for $m \geq 3$. The only place in our proof where well-nestedness is used is Lemma 14. While it is an open question whether this lemma holds of m-MCFGs in general, it is easy to see that it holds of (not necessarily well-nested) 2-MCFGs. Thus we have

Theorem 17. *For every* 2*-MCFG* G*, all but finitely many strings* $z \in L(G)$ *are* 4*-pumpable in* $L(G)$*.*

References

Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., Tommasi, M.: Tree Automata Techniques and Applications, October 12 (2007), `http://tata.gforge.inria.fr/`

Engelfriet, J., Heyker, L.: The string generating power of context-free hypergraph grammars. Journa of Computer and System Sciences 43, 328–360 (1991)

Fisher, M.J.: Grammars with Macro-Like Productions. Ph.D. thesis, Harvard University (1968)

Gécseg, F., Steinby, M.: Tree languages. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, Beyond Words, vol. 3, pp. 1–68. Springer, Berlin (1997)

Groenink, A.: Surface without Structure. Ph.D. thesis, Utrecht University (1997)

Hotz, G., Pitsch, G.: On parsing coupled-context-free languages. Thoretical Computer Science 161, 205–253 (1996)

Joshi, A.K., Schabes, Y.: Tree-adjoining grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, Beyond Words, vol. 3, pp. 69–123. Springer, Berlin (1997)

Kanazawa, M., Salvati, S.: Generating control languages with abstract categorial grammars. In: The preliminary proceedings of FG-2007: The 12th Conference on Formal Grammar (2007)

Kepser, S., Mönnich, U.: Closure properties of linear context-free tree languages with an application to optimality theory. Theoretical Computer Science 354(1), 82–97 (2006)

Kracht, M.: The Mathematics of Language. Mouton de Gruyter, Berlin (2003)

Kuhlman, M., Möhl, M.: Mildly context-sensitive dependency languages. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, pp. 160–167 (2007a)

Kuhlman, M., Möhl, M.: The string-generative capacity of regular dependency languages. In: FG-2007 (2007b)

Michaelis, J.: An additional observation on strict derivational minimalism. In: Rogers, J. (ed.) Proceedings of FG-MoL 2005: The 10th conference on Formal Grammar and the 9th Meeting on Mathematics of Language, pp. 101–111. CSLI Publications, Stanford (2009)

Michaelis, J., Kracht, M.: Semilinearity as a syntactic invariant. In: Retoré, C. (ed.) Logical Aspects of Computational Linguistics, pp. 329–345. Springer, Berlin (1997)

Palis, M.A., Shende, S.M.: Pumping lemmas for the control language hierarchy. Mathematical Systems Theory 28(3), 199–213 (1995)

Radzinski, D.: Chinese number-names, tree adjoining languages, and mild context-sensitivity. Computational Linguistics 17(3), 277–299 (1991)

Rambow, O., Satta, G.: Independent parallelism in finite copying parallel rewriting systems. Theoretical Computer Science 223, 87–120 (1999)

Seki, H., Kato, Y.: On the generative power of multiple context-free grammars and macro grammars. IEICE Transactions on Information and Systems E91–D(2), 209–221 (2008)

Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science 88(2), 191–229 (1991)

Smullyan, R.M.: Theory of Formal Systems. Princeton University Press, Princeton (1961)

Staudacher, P.: New frontiers beyond context-freeness: DI-grammars and DI-automata. In: 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL 1993), pp. 358–367 (1993)

Tamaki, H., Sato, T.: Unfold/fold transformation of logic programs. In: Proceedings of the Second International Conference on Logic Programming, pp. 127–138 (1984)

Vijay-Shanker, K., Weir, D.J., Joshi, A.K.: Characterizing structural descriptions produced by various grammatical formalisms. In: 25th Annual Meeting of the Association for Computational Linguistics, pp. 104–111 (1987)

Villemonte de la Clergerie, É.: Parsing MCS languages with thread automata. In: Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6), pp. 101–108 (2002a)

Villemonte de la Clergerie, É.: Parsing mildly context-sensitive languages with thread automata. In: Proceedings of the 19th International Conference on Computational Linguistics, pp. 1–7 (2002b)

Weir, D.: Linear context-free rewriting systems and deterministic tree-walking transducers. In: Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics, pp. 136–143 (1992)

Weir, D.J.: Characterizing Mildly Context-Sensitive Grammar Formalisms. Ph.D. thesis, University of Pennsylvania (1988)

The Support of a Recognizable Series over a Zero-Sum Free, Commutative Semiring Is Recognizable

Daniel Kirsten

University Leipzig, Institute for Computer Science,
Postfach 10 09 20, 04009 Leipzig, Germany
www.informatik.uni-leipzig.de/~kirsten/

Abstract. We show that the support of a recognizable series over a zero-sum free, commutative semiring is a recognizable language. We also give a sufficient and necessary condition for the existence of an effective transformation of a weighted automaton recognizing a series S over a zero-sum free, commutative semiring into an automaton recognizing the support of S.

1 Introduction

One stream in the rich theory of formal power series deals with connections to formal languages. To each formal power series, one associates a certain language, called the support, which consists of all words which are not mapped to zero.

It is well-known that the support of some recognizable series is not always a recognizable language. However, for large classes of semirings, it is known that the support of a recognizable series is always recognizable, see [3, 8] for recent overviews. These classes include all positive semirings (semirings which are both zero-divisor free and zero-sum free), all finite, and more generally, all locally finite semirings.

Wang introduced the notion of a quasi-positive semiring (that is, for every $k \in \mathbb{K} \setminus \{0\}$, $\ell \in \mathbb{K}$, $n \in \mathbb{N}$, it holds $k^n + \ell \neq 0$), and showed that the support of some recognizable series over a commutative, quasi-positive semiring is always a recognizable language [10]. Every quasi-positive semiring is zero-sum free by definition.

In the present paper, we generalize Wang's result to all zero-sum free, commutative semirings. The proof relies on Dickson's lemma.

Further, we examine under which assumptions we can effectively transform a weighted automaton recognizing some series S over a zero-sum free, commutative semiring into an automaton recognizing the support of S. For this, we introduce the zero generation problem (see Section 3) and show that the decidability of the zero generation problem is a sufficient and necessary condition for the existence of such a transformation. Surprisingly, the computability of the semiring operations is not related to the effectivity of the transformation.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 326–333, 2009.

The paper is organized as follows: In Section 2, we deal with some preliminaries. In Section 3, we present known results and the contribution of the paper. To keep Section 3 as a succinct survey, the main proofs are shifted to Section 4.

2 Preliminaries

2.1 Notations

Let $\mathbb{N} = \{0, 1, \ldots\}$.

Let $n \in \mathbb{N}$. Given some tuple $\bar{x} \in \mathbb{N}^n$, we denote by x_i the i-th component of $\bar{x}$ for $i \in \{1, \ldots, n\}$. Given two tuples $\bar{x}, \bar{y} \in \mathbb{N}^n$, we denote $\bar{x} \leq \bar{y}$ if $x_i \leq y_i$ for every $i \in \{1, \ldots, n\}$. If $\bar{x} \leq \bar{y}$ and $x_i < y_i$ for some $i \in \{1, \ldots, n\}$, then we denote $\bar{x} < \bar{y}$.

Given some subset $M \subseteq \mathbb{N}^n$, we denote by $\mathsf{Min}(M)$ the set of all minimal tuples of M, that is, $\mathsf{Min}(M) = \{\bar{x} \in M \mid \text{ for every } \bar{y} \in M, \bar{y} \leq \bar{x} \text{ implies } \bar{x} = \bar{y}\}$.

The following lemma is well-known in combinatorics, order theory, and commutative algebra, see e.g. [5],

Lemma 1 (Dickson's lemma). *For every $M \subseteq \mathbb{N}^n$, the set $\mathsf{Min}(M)$ is finite.*

Given some $\bar{x} \in \mathbb{N}^n$ and some $z \in \mathbb{N}$, we denote by $\lfloor \bar{x} \rfloor_z$ the tuple defined by $\left(\lfloor \bar{x} \rfloor_z\right)_i = \mathsf{min}\{x_i, z\}$ for every $i \in \{1, \ldots, n\}$.

Let Σ be some finite alphabet. We denote the empty word by ε. We denote by $|w|$ the length of some word $w \in \Sigma^*$. For every $w \in \Sigma^*$, $a \in \Sigma$, let $|w|_a$ be the number of occurrences of the letter a in w.

A monoid $(\mathbb{M}, \cdot, 1)$ consists of a set $\mathbb{M}$ together with a binary associative operation $\cdot$ and an identity 1.

We call some monoid $(\mathbb{M}, \cdot, 1)$ *commutative* if $k\ell = \ell k$ for every $k, \ell \in \mathbb{M}$.

We call some $0 \in \mathbb{M}$ a *zero*, if $k0 = 0k = 0$ for every $k \in \mathbb{M}$.

Given some monoid $\mathbb{M}$, some $m \in \mathbb{N}$, and $s_1, \ldots, s_m \in \mathbb{M}$, we denote by $\langle s_1, \ldots, s_m \rangle$ the submonoid of $\mathbb{M}$ generated by $s_1, \ldots, s_m$, that is, the smallest monoid $\mathbb{M}' \subseteq \mathbb{M}$ satisfying $s_1, \ldots, s_m \in \mathbb{M}'$.

A *semiring* $(\mathbb{K}, +, \cdot, 0, 1)$ consists of a set $\mathbb{K}$ together with two binary operations $+$ and $\cdot$ such that $(\mathbb{K}, +, 0)$ is a commutative monoid, $(\mathbb{K}, \cdot, 1)$ is a monoid with zero 0, and $(\mathbb{K}, \cdot, 1)$ distributes over $(\mathbb{K}, +, 0)$.

We call some semiring $(\mathbb{K}, +, \cdot, 0, 1)$ *commutative* if $(\mathbb{K}, \cdot, 1)$ is a commutative monoid.

We call $\mathbb{K}$ *zero-divisor free* if for every $k, \ell \in \mathbb{K} \setminus \{0\}$, we have $k\ell \neq 0$. We call $\mathbb{K}$ *zero-sum free* if for every $k, \ell \in \mathbb{K} \setminus \{0\}$, we have $k + \ell \neq 0$. Semirings which are zero-divisor free and zero-sum free are called *positive* semirings.

2.2 Weighted Finite Automata

We recall some notions on (weighted) automata and recommend [1, 2, 6, 4, 7, 9] for overviews.

Let $(\mathbb{K}, +, \cdot, 0, 1)$ be a semiring. Mappings from Σ^* to $\mathbb{K}$ are often called *series*. We denote the class of all series from Σ^* to $\mathbb{K}$ by $\mathbb{K}\langle\langle \Sigma^* \rangle\rangle$.

A *weighted finite automaton* (for short *WFA*) over $\mathbb{K}$ is a tuple $[Q, E, \lambda, \varrho]$, where

- Q is a non-empty, finite set of *states*,
- E is a finite subset of $Q \times \Sigma \times \mathbb{K} \times Q$, and
- $\lambda, \varrho : Q \to \mathbb{K}$.

We call the tuples in E *transitions.* For every $q \in Q$, we call $\lambda(q)$ resp. $\varrho(q)$ the *initial weight* resp. *accepting weight* of q. We call states $q \in Q$ which satisfy $\lambda(q) \neq 0$ (resp. $\varrho(q) \neq 0$) *initial* (resp. *accepting*) states.

Let $\mathcal{A} = [Q, E, \lambda, \varrho]$ be a WFA. Let $n \geq 1$. A path π of length n is a sequence

$$(q_0, a_1, s_1, q_1)\,(q_1, a_2, s_2, q_2)\ \ldots\ (q_{n-1}, a_n, s_n, q_n)$$

of transitions in E. We call the word $a_1 \ldots a_n$ the *label* of π. We define $\sigma(\pi) = \lambda(q_0) \cdot s_1 \cdot s_2 \cdot \cdots \cdot s_n \cdot \varrho(q_n)$, the *weight* of π. For every state $q \in Q$, we assume some path from q to q which is labeled with ε and weighted with 1.

For every $p, q \in Q$ and every $w \in \Sigma^*$, we denote by $p \overset{w}{\rightsquigarrow} q$ the set of all paths with label w which start at p and end at q. Then, $\mathcal{A}$ defines a series $|\mathcal{A}| : \Sigma^* \to \mathbb{K}$ by

$$|\mathcal{A}|(w) \;=\; \sum_{p,q \in Q,\ \pi \in p \overset{w}{\rightsquigarrow} q} \sigma(\pi)$$

for every $w \in \Sigma^*$.

We call some series $S : \Sigma^* \to \mathbb{K}$ *recognizable* if $S = |\mathcal{A}|$ for some WFA $\mathcal{A}$.

We define the *support* of some series $S : \Sigma^* \to \mathbb{K}$ as

$$\mathsf{supp}(S) = \{w \in \Sigma^* \mid S(w) \neq 0\}.$$

An *(unweighted) automaton* is a tuple $\mathcal{A} = [Q, E, I, F]$, where Q is a finite set, $E \subseteq Q \times \Sigma \times Q$, $I \subseteq Q$, and $F \subseteq Q$.

Let $\mathcal{A} = [Q, E, \lambda, \varrho]$ be an automaton. Let $n \geq 1$. A path π of length n is a sequence

$$(q_0, a_1, q_1)\,(q_1, a_2, q_2)\ \ldots\ (q_{n-1}, a_n, q_n)$$

of transitions in E. As above, we call $a_1 \ldots a_n$ the *label* of π. We call π successful, if $q_0 \in I$ and $q_n \in F$. We denote by $L(\mathcal{A})$ the language of $\mathcal{A}$, that is, the language consisting of all labels of successful paths.

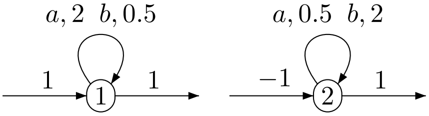
3 Overview and Main Results

The supports of recognizable series are well-studied objects, see [3, 8] for recent overviews.

It is well known that there are recognizable series S such that $\mathsf{supp}(S)$ is not a recognizable language. A folklore example is the series S over the semiring of the rational numbers $(\mathbb{Q}, +, \cdot, 0, 1)$ defined by $S(w) = 2^{|w|_a} 0.5^{|w|_b} - 0.5^{|w|_a} 2^{|w|_b}$. For every $w \in \Sigma^*$, we have $S(w) = 0$ iff $|w|_a = |w|_b$. Hence,

$$\mathsf{supp}(S) \;=\; \big\{ w \in \Sigma^* \;\big|\; |w|_a \neq |w|_b \big\}$$

which is not a recognizable language. Nevertheless, S is a recognizable series: just consider the WFA, below.



However, for a large class of semirings, the support of a recognizable series is always a recognizable language. It is well known that this class includes all positive semirings and all (locally) finite semirings [3, 8].

Moreover, Wang [10] defined the notion of a quasi-positive semiring: some semiring $\mathbb{K}$ is called *quasi-positive* if for every $k \in \mathbb{K} \setminus \{0\}$, $\ell \in \mathbb{K}$, $n \in \mathbb{N}$, it holds $k^n + \ell \neq 0$. Every positive semiring is quasi-positive, and every quasi-positive semiring is zero-sum free. However, there are zero-sum free semirings which are not quasi-positive, e.g., matrices over $(\mathbb{Q}_+, +, \cdot, 0, 1)$, and there are quasi-positive semirings which are not positive, e.g., direct products of positive semirings [10].

Wang showed that for every recognizable series S over some commutative, quasi-positive semiring, $\mathsf{supp}(S)$ is recognizable [10].

In the present paper, we generalize Wang's result to all commutative, zero-sum free semirings.

Further, we examine under which assumptions we can effectively construct an automaton recognizing $\mathsf{supp}(S)$ from a WFA recognizing S. Surprisingly, the computability of $+$ or $\cdot$ is not related to the effectivity of the construction. To achieve an effective construction, we introduce the *zero generation problem* (for short *ZGP*):

Let $\mathbb{M}$ be some monoid with a zero. An instance of the ZGP consists of two integers $m, m' \in \mathbb{N}$ and $s_1, \ldots, s_m, s'_1, \ldots, s'_{m'} \in \mathbb{M}$. The ZGP means to decide whether $0 \in s_1 \cdots s_m \cdot \langle s'_1, \ldots, s'_{m'} \rangle$, i.e., whether there exists some $s \in \langle s'_1, \ldots, s'_{m'} \rangle$ such that the product $s_1 \cdots s_m \cdot s$ yields zero. The presentation of the ZGP seems to be circumstantial, but we want to avoid using the computability of the product in $\mathbb{M}$.

We can show that the decidability of the ZGP of the monoid $(\mathbb{K}, \cdot, 1)$ is a sufficient and necessary condition for the effectivity of the construction of the automaton recognizing the support of some recognizable series.

To sum up:

Theorem 1. *Let Σ be an alphabet and $(\mathbb{K}, +, \cdot, 0, 1)$ be a zero-sum free, commutative semiring.*

1. *For every recognizable series $S \in \mathbb{K}\langle\langle \Sigma^* \rangle\rangle$, $\mathsf{supp}(S)$ is a recognizable language.*
2. *Assume $|\Sigma| \geq 2$. Given some WFA $\mathcal{A}$ over $\mathbb{K}$, we can effectively construct an automaton which recognizes $\mathsf{supp}(|\mathcal{A}|)$ iff $(\mathbb{K}, \cdot, 1)$ has a decidable ZGP.*

Clearly, the construction in (2) is also effective for $|\Sigma| = 1$. But if $|\Sigma| = 1$ we cannot show that the decidability of the ZGP is a necessary condition.

4 The Main Proof

4.1 Dickson's Lemma and Computability

Throughout this section, let $(\mathbb{M}, \cdot, 1)$ be some commutative monoid with a zero 0 and let $C = (c_1, \ldots, c_n) \in \mathbb{M}^n$. We denote by $[\![\,]\!] : (\mathbb{N}^n, +, (0, \ldots, 0)) \to (\mathbb{M}, \cdot, 1)$ the homomorphism defined by $[\![\bar{x}]\!] = c_1^{x_1} \cdots c_n^{x_n}$ for every tuple $\bar{x} = (x_1, \ldots, x_n) \in \mathbb{N}^n$.

We are interested in the set of all $\bar{x} \in \mathbb{N}^n$ satisfying $[\![\bar{x}]\!] = 0$, i.e., we are interested in the set $[\![0]\!]^{-1}$.

Given some $\bar{x} \in [\![0]\!]^{-1}$ and some $\bar{y} \in \mathbb{N}^n$ satisfying $\bar{x} \leq \bar{y}$, we have $\bar{y} \in [\![0]\!]^{-1}$.

By Lemma 1, the set $\mathsf{Min}([\![0]\!]^{-1})$ is finite. We denote by $\mathsf{dg}(C)$ the least non-negative integer such that $\mathsf{Min}([\![0]\!]^{-1})$ is a subset of $\{0, \ldots, \mathsf{dg}(C)\}^n$.

Lemma 2. *For every $\bar{x} \in \mathbb{N}^n$, we have $[\![\bar{x}]\!] = 0$ iff $[\![\lfloor \bar{x} \rfloor_{\mathsf{dg}(C)}]\!] = 0$.*

Proof. We have "$\Leftarrow$", since $\lfloor \bar{x} \rfloor_{\mathsf{dg}(C)} \leq \bar{x}$.

We show "$\Rightarrow$". Since $\bar{x} \in [\![0]\!]^{-1}$, there is some $\bar{y} \in \mathsf{Min}([\![0]\!]^{-1})$ satisfying $\bar{y} \leq \bar{x}$. Let $i \in \{1, \ldots, n\}$. If $x_i \leq \mathsf{dg}(C)$, then $y_i \leq x_i = (\lfloor \bar{x} \rfloor_{\mathsf{dg}(C)})_i$. If $x_i > \mathsf{dg}(C)$, then $y_i \leq \mathsf{dg}(C) = (\lfloor \bar{x} \rfloor_{\mathsf{dg}(C)})_i$ by the definitions of $\mathsf{dg}(C)$ and $\lfloor \bar{x} \rfloor_{\mathsf{dg}(C)}$. Consequently, $\bar{y} \leq \lfloor \bar{x} \rfloor_{\mathsf{dg}(C)}$, and hence, $\lfloor \bar{x} \rfloor_{\mathsf{dg}(C)} \in [\![0]\!]^{-1}$.

For the effectivity of our construction of the support automaton, it is very important to compute $\mathsf{dg}(C)$ from a given tuple C. Assume that the ZGP is decidable.

Given some $m \geq 1$ and $s_1, \ldots, s_m \in \mathbb{M}$, we can decide whether $s_1 \cdots s_m = 0$ by setting $m' = 1$, $s'_1 = 1$ and applying the algorithm for the ZGP.

Given some $m' \in \mathbb{N}$ and $s'_1, \ldots, s'_m \in \mathbb{M}$, we can decide whether $0 \in \langle s'_1, \ldots, s'_{m'} \rangle$ by setting $m = 0$ and applying the algorithm for the ZGP.

Lemma 3. *If the ZGP is decidable in $\mathbb{M}$, then we can effectively compute $\mathsf{dg}(C)$ from C.*

Proof. It suffices to show that for given $n \in \mathbb{N}$, $C = (c_1, \ldots, c_n) \in \mathbb{M}^n$, and $z \in \mathbb{N}$, we can decide whether $z < \mathsf{dg}(C)$. The algorithm can then check for increasing $z \in \{0, 1, 2, \ldots\}$ whether $z < \mathsf{dg}(C)$, and put out the least z which does not satisfy $z < \mathsf{dg}(C)$.

So assume n, C, z as above. We want to show that $z < \mathsf{dg}(C)$ iff there exists a tuple $\bar{x} \in \{0, \ldots, z\}^n$ which satisfies the following properties:

1. We have $x_i = z$ for some $i \in \{1, \ldots, n\}$.
2. We have $[\![\bar{x}]\!] \neq 0$. Given C and $\bar{x}$, it is decidable whether $[\![\bar{x}]\!] \neq 0$ by the decidability of the ZGP.
3. There is some $\bar{y} \in \mathbb{N}^n$ such that $\bar{x} = \lfloor \bar{y} \rfloor_z$ and $[\![\bar{y}]\!] = 0$.
   Given C and $\bar{x}$, this condition is decidable as follows: Let $m = \sum_{i=1}^n x_i$. Let $s_1, \ldots, s_m$ be the list over $\mathbb{M}$ constructed by putting x_1 times c_1, x_2 times c_2, ..., and x_n times c_n. We have $s_1 \cdots s_m = [\![\bar{x}]\!]$.
   Let $m' \geq 1$ and $s'_1, \ldots, s'_{m'} \in \mathbb{M}$ be a list of the c_i's for the $i \in \{1, \ldots, n\}$ satisfying $x_i = z$.

Clearly, there exists some $\bar{y} \in \mathbb{N}^n$ such that $\bar{y} = \lfloor \bar{x} \rfloor_z$ and $[\![\bar{y}]\!] = 0$ iff $0 \in s_1 \cdots s_m \cdot \langle s'_1, \ldots, s'_{m'} \rangle$. The latter condition is decidable.

Assume $z < \mathsf{dg}(C)$. Choose some $\bar{y} \in \mathsf{Min}\big([\![0]\!]^{-1}\big)$ such that at least one entry of $\bar{y}$ equals $\mathsf{dg}(C)$. Let $\bar{x} = \lfloor \bar{y} \rfloor_z$. It is easy to verify that $\bar{x}$ satisfies (1) and (3), above. Moreover, $\bar{x}$ satisfies (2), since $\bar{y}$ is chosen from $\mathsf{Min}\big([\![0]\!]^{-1}\big)$ and $\bar{x} < \bar{y}$.

Assume $z \geq \mathsf{dg}(C)$. Let $\bar{x}, \bar{y} \in \mathbb{N}^n$ such that (1) and (3) are satisfied. From Lemma 2, it follows $[\![\lfloor \bar{y} \rfloor_{\mathsf{dg}(C)}]\!] = 0$. Since $\mathsf{dg}(C) \leq z$, we have $\lfloor \bar{y} \rfloor_{\mathsf{dg}(C)} \leq \lfloor \bar{y} \rfloor_z = \bar{x}$, and hence, $[\![\bar{x}]\!] = 0$, i.e., $\bar{x}$ does not satisfy (2).

An algorithm to decide whether $z < \mathsf{dg}(C)$ can check by brute force whether there is some $\bar{x} \in \{0, \ldots, z\}^n$ which satisfies (1), (2), and (3).

4.2 The Construction of a Support Automaton

Proof (Theorem 1). In the first part of the proof we prove (1) and "$\Leftarrow$" in (2).

Let S be the series computed by some WFA $\mathcal{A} = [Q, E, \lambda, \varrho]$.

Let $n \in \mathbb{N}$ and $C = (c_1, \ldots, c_n) \in \mathbb{K}^n$ such that

- for every $(p, a, s, q) \in E$, there is exactly one $i \in \{1, \ldots, n\}$ satisfying $c_i = s$.
- for every $q \in Q$, there is exactly one $i \in \{1, \ldots, n\}$ satisfying $\lambda(q) = c_i$, and there is exactly one $i \in \{1, \ldots, n\}$ satisfying $\varrho(q) = c_i$.

We further assume that for every $i \in \{1, \ldots, n\}$, c_i occurs in $\mathcal{A}$ as a weight of some transition or as some initial or accepting weight.

If the ZGP is decidable, we can effectively compute $\mathsf{dg}(C)$ by Lemma 3.

We construct an (unweighted) automaton $\mathcal{A}_s$. The states of $\mathcal{A}_s$ are tuples $Q_s = \{0, \ldots, \mathsf{dg}(C)\}^n \times Q$.

Some state $(\bar{x}, q) \in Q_s$ is an initial state iff there exists some $i \in \{1, \ldots, n\}$ such that

- $x_i = 1, \ \lambda(q) = c_i$, and
- for every $j \in \{1, \ldots, n\}$, $j \neq i$, we have $x_j = 0$.

Consequently, $[\![\bar{x}]\!] = \lambda(q)$.

The reader should be aware that there might exist states $(\bar{x}, q) \in Q_s$ which are *not* initial states although they satisfy the condition $[\![\bar{x}]\!] = \lambda(q)$. For example, if $x_1 = x_2 = 1$, $x_3 = \cdots = x_n = 0$ and $c_1 c_2 = \lambda(q)$, then $(\bar{x}, q)$ is not an initial state although $[\![\bar{x}]\!] = \lambda(q)$. This restriction is due to the fact that $[\![\bar{x}]\!] = \lambda(q)$ is in general undecidable, even if the ZGP is decidable.

We denote the set of initial states by I_s.

We define a partial mapping $\oplus : \{0, \ldots, \mathsf{dg}(C)\}^n \times \mathbb{K} \dashrightarrow \{0, \ldots, \mathsf{dg}(C)\}^n$. Let $\bar{x} \in \{0, \ldots, \mathsf{dg}(C)\}^n$ and $s \in \mathbb{K}$. We define $\bar{x} \oplus s$ iff s occurs in C. So assume that there is some unique $i \in \{1, \ldots, n\}$ such that $c_i = s$. Let $\bar{y} \in \{0, \ldots, \mathsf{dg}(C)\}^n$ be defined by

$$y_j = \begin{cases} x_j + 1 & \text{if } j = i \\ x_j & \text{if } j \neq i. \end{cases}$$

We define $\bar{x} \oplus s = \lfloor \bar{y} \rfloor_{\mathsf{dg}(C)}$.

The key idea behind $\oplus$ is that given some $m \in \mathbb{N}$, $s_1, \ldots, s_m \in \mathbb{K}$, the operation

$$(\cdots((\bar{x} \oplus s_1) \oplus s_2) \cdots \oplus s_m)$$

counts (up to $\mathsf{dg}(C)$) the number of occurrences of the c_i's in the sequence $s_1, \ldots, s_m$.

Some state $(\bar{x}, q) \in Q_s$ is an accepting state iff $[\![\bar{x} \oplus \varrho(q)]\!] \neq 0$. Using the decidability of the ZGP, we can decide whether $(\bar{x}, q)$ is an accepting state. We denote the set of accepting states by F_s.

Let $(\bar{x}, p), (\bar{y}, q) \in Q_s$ and $a \in \Sigma$. The triple $\big((\bar{x}, p), a, (\bar{y}, q)\big)$ is a transition in E_s iff there exists a transition $(p, a, s, q) \in E$ such that $\bar{x} \oplus s = \bar{y}$. We say that $\big((\bar{x}, p), a, (\bar{y}, q)\big)$ stems from $(p, a, s, q) \in E$.

Let $\mathcal{A}_s = [Q_s, E_s, I_s, F_s]$. We want to show $L(\mathcal{A}_s) = \mathsf{supp}(S)$.

Let $w \in L(\mathcal{A}_s)$. There are $(\bar{x}_0, q_0) \in I_s$, $(\bar{x}_{|w|}, q_{|w|}) \in F_s$, and some path $\pi \in (\bar{x}_0, q_0) \overset{w}{\rightsquigarrow} (\bar{x}_{|w|}, q_{|w|})$ satisfying $[\![\bar{x}_{|w|} \oplus \varrho(q_{|w|})]\!] \neq 0$.

We denote the states of π by $(\bar{x}_0, q_0)$, $(\bar{x}_1, q_1)$, $\ldots$, $(\bar{x}_{|w|}, q_{|w|})$.

For $j \in \{1, \ldots, |w|\}$, let $t_j \in E$ such that the j-th transition of π stems from t_j. Clearly, $t_1 \cdots t_{|w|} \in q_0 \overset{w}{\rightsquigarrow} q_{|w|}$ is a path in $\mathcal{A}$.

For every $j \in \{1, \ldots, |w|\}$, let $s_j \in \mathbb{K}$ be the weight of t_j. For $j \in \{0, \ldots, |w|\}$, let $\bar{y}_j \in \mathbb{N}^n$ be the tuple such that for every $i \in \{1, \ldots, n\}$, $y_{j,i}$ is the number of occurrences of c_i among $\lambda(q_0), s_1, \ldots, s_j$. In particular $\bar{y}_0 = \bar{x}_0$.

Let $\bar{y} \in \mathbb{N}^n$ such that for every $i \in \{1, \ldots, n\}$, y_i is the number of occurrences of c_i among $\lambda(q_0), s_1, \ldots, s_{|w|}, \varrho(q_{|w|})$. Clearly, $[\![\bar{y}]\!]$ is the weight of the path $t_1 \cdots t_{|w|}$.

By a straightforward inductive argument, we can show that for every $j \in \{0, \ldots, |w|\}$, $\bar{x}_j = \lfloor \bar{y}_j \rfloor_{\mathsf{dg}(C)}$, and $\bar{x}_{|w|} \oplus \varrho(q_{|w|}) = \lfloor \bar{y} \rfloor_{\mathsf{dg}(C)}$.

Since $(\bar{x}_{|w|}, q_{|w|}) \in F_s$, we have $[\![\bar{x}_{|w|} \oplus \varrho(q_{|w|})]\!] \neq 0$, and hence, $[\![\lfloor \bar{y} \rfloor_{\mathsf{dg}(C)}]\!] \neq 0$. By Lemma 2, we have $[\![\bar{y}]\!] \neq 0$, i.e., the weight of the path $t_1 \cdots t_{|w|}$ is different from 0. Since $\mathbb{K}$ is zero-sum-free, we have $w \in \mathsf{supp}(|\mathcal{A}|)$.

Thus, we have shown $L(\mathcal{A}_s) \subseteq \mathsf{supp}(|\mathcal{A}|)$. To show $L(\mathcal{A}_s) \supseteq \mathsf{supp}(|\mathcal{A}|)$, we can proceed in the same way. We assume some $w \in \mathsf{supp}(|\mathcal{A}|)$, some accepting path $t_1 \ldots t_{|w|}$ with non-zero weight for w in $\mathcal{A}$, and construct an accepting path for w in $\mathcal{A}_s$. For $j \in \{1, \ldots, |w|\}$, denote $t_j = (q_{j-1}, a_j, s_j, q_j)$. Let $\bar{x}_0 = (0, \ldots, 0) \oplus \lambda(q_0)$. For $j \in \{1, \ldots, |w|\}$, let $\bar{x}_j = \bar{x}_{j-1} \oplus s_j$. We can argue as above to show that the transitions $\big((\bar{x}_{i-1}, q_{i-1}), a_j, (\bar{x}_i, q_i)\big)$ form an accepting path for w in $\mathcal{A}_s$. To sum up, $L(\mathcal{A}_s) = \mathsf{supp}(|\mathcal{A}|)$.

We have shown (1) and "⇐" in (2). It remains to show "⇒" in (2).

Assume two integers $m, m' \in \mathbb{N}$ and $s_1, \ldots, s_m, s'_1, \ldots, s'_{m'} \in \mathbb{M}$.

Let $w_1, \ldots, w_{m'} \in \Sigma^*$ be mutually distinct, non-empty words of equal length.

We sketch the construction of a WFA $\mathcal{A}$. It has just one initial and one accepting state. The initial and accepting weights are 1. Let a be some letter from Σ. There is one path from the initial to the accepting state. This path is labeled with a^m. The transition weights along this path are $s_1, \ldots, s_m$. For every $j \in \{1, \ldots, m'\}$, we add a loop at the accepting state which is labeled with w_j. The first transition of the loop is weighted with s'_j, the remaining transitions of the loop are weighted with 1.

For every n and $i_1, \ldots, i_n \in \{1, \ldots, m'\}$, we have

$$|\mathcal{A}|(a^m w_{i_1} \ldots w_{i_n}) \;=\; s_1 \cdots s_m \cdot s'_{i_1} \cdots s'_{i_n}.$$

Moreover, we have $\mathsf{supp}(|\mathcal{A}|) = a^m\{w_1, \ldots, w_{m'}\}^*$ iff $0 \notin s_1 \cdots s_m \cdot \langle s'_1 \cdots s'_{m'} \rangle$.

Since we show "⇒" in (2), we can assume an effective construction of an automaton $\mathcal{A}_s$ which recognizes $\mathsf{supp}(|\mathcal{A}|)$. Hence, we can decide whether it holds $\mathsf{supp}(|\mathcal{A}|) = a^m\{w_1, \ldots, w_{m'}\}^*$, i.e., we can decide the ZGP.

References

[1] Berstel, J.: Transductions and Context-Free Languages. B. G. Teubner, Stuttgart (1979)

[2] Berstel, J., Reutenauer, C.: Rational Series and Their Languages. EATCS Monographs on Theoretical Computer Science, vol. 12. Springer, New York (1984)

[3] Berstel, J., Reutenauer, C.: Noncommutative rational series with applications (prelimary electronic version) (2009), http://www-igm.univ-mlv.fr/~berstel/

[4] Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. Monographs in Theoretical Computer Science. An EATCS Series. Springer, Heidelberg (2009)

[5] Kreuzer, M., Robbiano, L.: Computational Commutative Algebra 1. Springer, Heidelberg (2008)

[6] Kuich, W.: Semirings and formal power series. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, Word, Language, Grammar, vol. 1, pp. 609–677. Springer, Berlin (1997)

[7] Reutenauer, C.: A survey on noncommutative rational series. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 24, 159–169 (1996)

[8] Sakarovitch, J.: Rational and recognisable power series. In: [4], ch. 4 (2009)

[9] Salomaa, A., Soittola, M.: Automata-Theoretic Aspects of Formal Power Series. Texts and Monographs on Computer Science. Springer, New York (1978)

[10] Wang, H.: On rational series and rational languages. Theoretical Computer Science 205(1-2), 329–336 (1998)

A Game-Theoretic Characterization of Boolean Grammars*

Vassilis Kountouriotis[1], Christos Nomikos[2], and Panos Rondogiannis[1]

[1] Department of Informatics & Telecommunications
University of Athens, Athens, Greece
{bk,prondo}@di.uoa.gr
[2] Department of Computer Science, University of Ioannina,
P.O. Box 1186, 45 110 Ioannina, Greece
cnomikos@cs.uoi.gr

Abstract. We obtain a simple, purely game-theoretic characterization of Boolean grammars [*A. Okhotin, Information and Computation, 194 (2004) 19-48*]. In particular, we propose a two-player infinite game of perfect information for Boolean grammars, which is equivalent to their well-founded semantics. The game is directly applicable to the simpler classes of conjunctive and context-free grammars, and it offers a promising new connection between game theory and formal languages.

1 Introduction

Boolean grammars were recently proposed by A. Okhotin [Okh04] as a generalization of context-free grammars. The main characteristic of Boolean grammars is that they allow conjunction and negation to appear in the right hand sides of rules. The resulting formalism has proven to be a very expressive one (see for example [OJ07]), while retaining to a large extent the efficient parsing properties of context-free grammars.

The theory of Boolean grammars is presently under rapid development. However, there exist many fundamental questions that still remain unanswered (see [Okh06] for an exposition of the basic open problems of the field). The area appears to be a quite intriguing one, since most of the problems remain unanswered even for the negation-free class (namely, for the class of conjunctive grammars [Okh01]).

In this paper we contribute to this area of research by providing a simple, purely game-theoretic characterization of the semantics of this type of grammars. In particular, we propose a two-player infinite game of perfect information for Boolean grammars, which is equivalent to their well-founded semantics [KNR06].

* This work is supported by the 03$E\Delta$ 330 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" ($\Pi ENE\Delta$) and co-financed by National and Community Funds (75% from E.U.-European Social Fund and 25% from the Greek Ministry of Development-General Secretariat of Research and Technology and from the private sector).

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 334–347, 2009.

The game we present has been inspired from a recent game-theoretic characterization of logic programs with negation [GRW08]. Actually, the present game is more complicated than the one in [GRW08] since it involves string manipulation by the two-players. A further contribution of our work is that it gives an alternative proof of correctness than the one derived in [GRW08]. More specifically, the proof in [GRW08] proceeds in two steps: it first establishes the determinacy of the logic programming game by using certain deep results from infinite game theory (namely, the theory of *Borel sets* [Mosch80] and Martin's *Borel Determinacy Theorem* [Mar75]); subsequently, based on the determinacy result, it establishes the equivalence of the game-semantics to the well-founded semantics of logic programs. Our present proof establishes *at the same time* both the determinacy of the game *and* its equivalence to the the well-founded semantics (avoiding completely the use of Borel sets and Martin's theorem).

The game characterization we propose has the advantage of being very simple to understand and present, due to its anthropomorphic flavor. In this respect, it appears to be easier to use than the corresponding well-founded approach of [KNR06]. We believe that the new approach will offer more benefits when used in order to prove the correctness of transformations on Boolean grammars, while the well-founded approach will be more appropriate for computing (inductively) the meaning of specific grammars.

The rest of the paper is organized as follows: Section 2 presents preliminary material. Section 3 gives an informal explanation of the game and motivates it by examples. Section 4 presents the basic notions regarding infinite games and gives a precise formalization of the game. Section 5 proves the equivalence of the game to the well-founded semantics of Boolean grammars. Section 6 contains pointers to future work.

2 Preliminaries

In [Okh01] and [Okh04] A. Okhotin introduced the classes of conjunctive and Boolean grammars respectively. Formally:

Definition 1. *A* Boolean grammar *is a quadruple $G = (\Sigma, N, P, S)$, where Σ and N are disjoint finite nonempty sets of terminal and nonterminal symbols respectively, P is a finite set of rules, each of the form*

$$C \to \alpha_1 \& \cdots \& \alpha_m \& \neg\beta_1 \& \cdots \& \neg\beta_n \qquad (m+n \geq 1, C \in N, \alpha_i, \beta_i \in (\Sigma \cup N)^*)$$

and $S \in N$ is the start symbol of the grammar. We will call the α_i's positive literals *and the $\neg\beta_i$'s* negative. *A Boolean grammar is called* conjunctive *if all its rules contain only positive literals.*

The semantics of Boolean grammars is not straightforward due to the possible existence in the rules of circularities through negation. To circumvent this problem, it has been proposed [KNR06] that the correct mathematical formulation of the meaning of Boolean grammars should be based on *three-valued formal languages*. Intuitively, given a three-valued language L and a string w over the alphabet of L, there are three-cases: either $w \in L$ (i.e., $L(w) = 1$), or $w \notin L$ (i.e., $L(w) = 0$), or finally, the membership of w in L is unclear (i.e., $L(w) = \frac{1}{2}$). Given this extended notion of language, it is now possible to interpret Boolean grammars with negative circularities. For example, the meaning of the grammar $S \to \neg S$ is the language which assigns to every string the value $\frac{1}{2}$. These ideas are formalized in the rest of this section (our presentation follows [NR08]).

Definition 2. *Let Σ be a finite non-empty set of symbols. Then, a (three-valued) language over Σ is a function from Σ^* to the set $\{0, \frac{1}{2}, 1\}$.*

The following definition, which generalizes the familiar notion of concatenation of languages, is also needed:

Definition 3. *Let Σ be a finite set of symbols and let $L_1, \ldots, L_n$ be (three-valued) languages over Σ. We define the* three-valued concatenation *of the languages $L_1, \ldots, L_n$ to be the language L such that:*

$$L(w) = \max_{\substack{(w_1,\ldots,w_n): \\ w = w_1 \cdots w_n}} \left(\min_{1 \le i \le n} L_i(w_i) \right)$$

The concatenation of $L_1, \ldots, L_n$ will be denoted by $L_1 \circ \cdots \circ L_n$.

We can now define the notion of *interpretation* of a given Boolean grammar:

Definition 4. *An* interpretation *I of a Boolean grammar $G = (\Sigma, N, P, S)$ is a function $I : N \to \left(\Sigma^* \to \{0, \frac{1}{2}, 1\}\right)$.*

An interpretation I can be recursively extended to apply to expressions that appear as the right-hand sides of Boolean grammar rules:

Definition 5. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I be an interpretation of G. Then I can be extended to apply to expressions that appear as the right-hand sides of Boolean grammar rules as follows:*

- *For the empty sequence ϵ and for all $w \in \Sigma^*$, it is $I(\epsilon)(w) = 1$ if $w = \epsilon$ and 0 otherwise.*
- *Let $a \in \Sigma$ be a terminal symbol. Then, for every $w \in \Sigma^*$, $I(a)(w) = 1$ if $w = a$ and 0 otherwise.*
- *Let $\alpha = \alpha_1 \cdots \alpha_n$, $n \geq 1$, be a sequence in $(\Sigma \cup N)^*$. Then, for every $w \in \Sigma^*$, it is $I(\alpha)(w) = (I(\alpha_1) \circ \cdots \circ I(\alpha_n))(w)$.*
- *Let $\alpha \in (\Sigma \cup N)^*$. Then, for every $w \in \Sigma^*$, $I(\neg\alpha)(w) = 1 - I(\alpha)(w)$.*
- *Let $l_1, \ldots, l_n$ be literals. Then, for every string $w \in \Sigma^*$, $I(l_1 \& \cdots \& l_n)(w) = \min\{I(l_1)(w), \ldots, I(l_n)(w)\}$.*

We are particularly interested in interpretations that *satisfy* all the rules of a given grammar:

Definition 6. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I an interpretation of G. Then, I is a* model *of G if for every rule $A \to l_1 \& \cdots \& l_n$ in P and for every $w \in \Sigma^*$, it is $I(A)(w) \geq I(l_1 \& \cdots \& l_n)(w)$.*

In the definition of the well-founded model, two orderings on interpretations play a crucial role. Given two interpretations, the first ordering (usually called the *standard ordering*) compares their *degree of truth*:

Definition 7. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I, J be two interpretations of G. Then, we say that $I \preceq J$ if for all $A \in N$ and for all $w \in \Sigma^*$, $I(A)(w) \leq J(A)(w)$.*

Among the interpretations of a given Boolean grammar, there is one which is the least with respect to the $\preceq$ ordering, denoted by $\perp$. It holds that for all A and all w, $\perp(A)(w) = 0$.

The second ordering (usually called the *Fitting ordering*) compares the *degree of information* of two interpretations:

Definition 8. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and I, J be two interpretations of G. Then, we say that $I \preceq_F J$ if for all $A \in N$ and for all $w \in \Sigma^*$, if $I(A)(w) = 0$ then $J(A)(w) = 0$ and if $I(A)(w) = 1$ then $J(A)(w) = 1$.*

Among the interpretations of a given Boolean grammar, there is one which is the least with respect to the $\preceq_F$ ordering, denoted by $\perp_F$. It holds that for all A and all w, $\perp_F (A)(w) = \frac{1}{2}$.

Given a set U of interpretations, we will write $lub_{\preceq} U$ (respectively $lub_{\preceq_F} U$) for the least upper bound of the members of U under the standard ordering (respectively, the Fitting ordering).

Consider a Boolean grammar G. Then, for any interpretation J of G we define the operator $\Theta_J : \mathcal{I} \rightarrow \mathcal{I}$ on the set $\mathcal{I}$ of all 3-valued interpretations of G.

Definition 9. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, let $\mathcal{I}$ be the set of all three-valued interpretations of G and let $J \in \mathcal{I}$. The operator $\Theta_J : \mathcal{I} \rightarrow \mathcal{I}$ is defined as follows. For every $I \in \mathcal{I}$, for all $A \in N$ and for all $w \in \Sigma^*$:*

1. *$\Theta_J(I)(A)(w) = 1$ if there is a rule $A \rightarrow l_1 \& \cdots \& l_n$ in P such that, for all positive l_i it is $I(l_i)(w) = 1$ and for all negative l_i it is $J(l_i)(w) = 1$;*
2. *$\Theta_J(I)(A)(w) = 0$ if for every rule $A \rightarrow l_1 \& \cdots \& l_n$ in P, either there exists a positive l_i such that $I(l_i)(w) = 0$ or there exists a negative l_i such that $J(l_i)(w) = 0$;*
3. *$\Theta_J(I)(A)(w) = \frac{1}{2}$, otherwise.*

An important fact regarding the operator Θ_J is that it is monotonic with respect to the $\preceq$ ordering of interpretations:

Theorem 1. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and let J be an interpretation of G. Then, the operator Θ_J is monotonic with respect to the $\preceq$ ordering of interpretations. Moreover, Θ_J has a unique least (with respect to $\preceq$) fixed point $\Theta_J^{\uparrow\omega}$ which is defined as follows:*

$$\begin{aligned} \Theta_J^{\uparrow 0} &= \perp \\ \Theta_J^{\uparrow n+1} &= \Theta_J(\Theta_J^{\uparrow n}) \\ \Theta_J^{\uparrow\omega} &= lub_{\preceq}\{\Theta_J^{\uparrow n} \mid n < \omega\} \end{aligned}$$

We will denote by $\Omega(J)$ the least fixed point $\Theta_J^{\uparrow\omega}$ of Θ_J. Given a grammar G, we can use the Ω operator to construct a sequence of interpretations whose ω-limit M_G is a distinguished model of G:

Theorem 2. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. Then, the operator $\Omega(J) = \Theta_J^{\uparrow\omega}$ is monotonic with respect to the $\preceq_F$ ordering of interpretations. Moreover, Ω has a unique least (with respect to $\preceq_F$) fixed point M_G which is defined as follows:*

$$\begin{aligned} M_0 &= \perp_F \\ M_{n+1} &= \Omega(M_n) \\ M_G &= lub_{\preceq_F}\{M_n \mid n < \omega\} \end{aligned}$$

Theorem 3. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar. Then, M_G is a model of G (which will be called the* well-founded model *of G).*

3 The Game for Boolean Grammars

Consider a Boolean grammar $G = (\Sigma, N, P, S)$ and let $A \in N$ and $w \in \Sigma^*$. We describe at an intuitive level a two-person game $\Gamma_{G_A^w}$ which has the property that $w \in L(G)$ if and only if Player I has a winning strategy in $\Gamma_{G_A^w}$; similarly, $w \notin L(G)$ if and only if Player II has a winning strategy in $\Gamma_{G_A^w}$

In the game $\Gamma_{G_A^w}$, Player I, who initially plays the role of the *Believer*, believes that the string w can be produced by the nonterminal A of the Boolean grammar G. For this reason, he initiates the game by initially playing a pair $(A \rightarrow l_1\& \cdots \& l_m, w)$, where $A \rightarrow l_1\& \cdots \& l_m$ is a rule in G. The intuitive explanation behind this move is "*I believe that w can be produced by A and I can prove this by using this specific rule of the grammar*". Player II, who initially plays the role of the *Doubter*, does not agree that w can be produced by nonterminal A. So, her reply to the move of Player I is a pair of the form (l_i, w), where l_i is one of the literals in the body of the rule that Player I has just played. The intuitive explanation in this case is "*I doubt that w can be produced from the rule you have just played and I am going to convince you about this by showing that w can not be produced from l_i*". It remains to specify the reaction of a player to a move of the form (l, u), for some literal l and $u \in \Sigma^*$. We have to distinguish the following subcases:

Case 1: $l = B$, where B is a nonterminal. Then, the player whose turn it is to play, plays a pair $(B \rightarrow l_1\& \cdots \& l_m, u)$, where $B \rightarrow l_1\& \cdots \& l_m$ is a rule in G. The intuitive explanation behind this move is identical to that specified in the beginning of the previous paragraph for the first move in the game.

Case 2: $l = \neg\alpha$, where $\alpha \in (\Sigma \cup N)^*$. Then, the player whose turn it is to play, must play the pair (α, u) as the next move. The intuitive reading here is : "*I doubt that u can be produced from α (and therefore I was right in my previous belief that u can be produced by the rule that contains $\neg\alpha$ in its body)*". Therefore, the significance of this rule is that when negation is encountered, the players swap roles: the Believer now becomes a Doubter and vice versa.

Case 3: $l = \alpha$, where α contains at least one nonterminal and $|\alpha| \geq 2$. Then, the player whose turn it is to play, partitions u into $|\alpha|$ parts (not necessarily of the same size), and plays (α, π) where π is the partition just mentioned. The intuition behind this rule is as follows: "*I believe that u can be produced from α and I can demonstrate this to you by partitioning u into $|\alpha|$ substrings such that each symbol from α can produce the corresponding substring from u*". The other player will then have to choose one of the symbols of the sequence α, say $\alpha(i)$, together with the corresponding string from the partition π, say $\pi(i)$, and play the move $(\alpha(i), \pi(i))$. The intuition now is: "*I doubt that u can be produced from α and I can demonstrate this to you by choosing a symbol $\alpha(i)$ from α and the corresponding part $\pi(i)$ from π such that $\alpha(i)$ does not produce $\pi(i)$*".

Case 4: $l = u$, i.e., the last move was of the form (u, u). Then, the player whose turn it is to play, plays the move $(\#, \#)$ to which there is no legal response. The intuition here is "*You have just doubted that u can be produced from u, which is completely unreasonable and I have to end the game*".

Before we specify the rules for winning, losing and getting a tie, we motivate the game with a few simple examples. Notice that a play of the game can be formed by following the above rules. It is possible that at some point of the play, one of the players has no legal move to play. It is also possible that the game goes on forever with both players playing legal moves. These cases are illustrated below:

Example 1. Consider the following grammar:

$$\begin{aligned} &S \rightarrow bbS \ \& \ \neg bSb \\ &S \rightarrow a \end{aligned}$$

Moreover, consider the string $w = bba$. The following is a possible play of the game $\Gamma_{G_S^{bba}}$:

Player I	*Player II*
$(S \rightarrow bbS \ \& \ \neg bSb,\ bba)$	$(\neg bSb,\ bba)$
$(bSb,\ bba)$	$(bSb,\ \langle b, b, a\rangle)$
(b, a)	

Obviously, Player II can not find any legal move to continue the game. One can easily see that Player I has a strategy that always "corners" Player II. As we are going to see shortly, this means that the string *bba* belongs to the language of the grammar.

Consider on the other hand the string *abb* and the corresponding game $\Gamma_{G_S^{abb}}$. The following is a possible play:

Player I	*Player II*
$(S \rightarrow bbS \ \& \ \neg bSb,\ abb)$	$(bbS,\ abb)$
$(bbS,\ \langle a, \epsilon, bb\rangle)$	$(b,\ a)$

In this case, Player I can not find any legal move to continue the game. It is easy to see that Player II has a strategy that always "corners" Player I. As we are going to see shortly, this means that the string *abb* does not belong to the language of the grammar. □

Example 2. Consider the context-free grammar $S \rightarrow aS$ (which does not produce any string). The following is a possible play of the game $\Gamma_{G_S^{aa}}$:

Player I	*Player II*
$(S \rightarrow aS,\ \ aa)$	$(aS,\ \ aa)$
$(aS,\ \ \langle\epsilon, aa\rangle)$	$(S,\ \ aa)$
$(S \rightarrow aS,\ \ aa)$	$(aS,\ \ aa)$
...	...

In this case the game goes on for ever. The winner of the play is Player II: if one of the players manages to remain a doubter for ever, then this player wins (see the rules just after the examples). □

Example 3. Consider the Boolean grammar $S \rightarrow \neg S$. The following is a possible play of the game $\Gamma_{G_S^{aa}}$:

Player I	*Player II*
$(S \rightarrow \neg S,\ \ aa)$	$(\neg S,\ \ aa)$
$(S,\ \ aa)$	$(S \rightarrow \neg S,\ \ aa)$
$(\neg S,\ \ aa)$	$(S,\ \ aa)$
...	...

In this case the game also goes on for ever. However, in this play the two players swap roles (the Believer becomes a Doubter and vice versa) infinitely often. The result of this play will be a tie. □

We can now state more generally the criteria for winning, losing and obtaining a tie. First of all, *any player who has no legal move loses immediately.* Furthermore, *if the game play is infinite and after a certain point one of the players remains a doubter, this player wins.* Finally, *if during a play the two players swap roles infinitely often, the result is a tie*: intuitively, none of the players has managed to get an advantage during this play, since they both engaged in circularities through negation.

4 A Formalization of the Game

In this section we formalize the game we have just described. In particular, we present some basic background on infinite games of perfect information, which we then use in order to present the proposed game for Boolean grammars in a formal way.

4.1 Infinite Games of Perfect Information

Infinite games of perfect information [GS53] are games that take place between two players that we will call *Player I* and *Player II*. In such games there does not exist any “hidden information”: both players know all the moves that have been played so far, and there are no simultaneous moves. The games are infinite in the sense that they do not terminate at a finite stage and therefore in order to derive the outcome of a play it may be necessary to examine its entire history.

Before defining perfect information games in a formal way, we need to introduce some notation. Sequences (finite or infinite in length) will usually be denoted by s or x. A finite sequence of length k will be denoted by $\langle s_0, s_2, \ldots, s_{k-1}\rangle$ and the empty sequence by $\langle\rangle$. Given a non-empty set X, the set of all infinite sequences of elements of X is denoted by X^ω. A *tree* on X is a set R of finite sequences of members of X such that if $u \in R$ and v is a prefix of u, then $v \in R$.

During a perfect information game, the two players exchange moves from a non-empty set X, called the *set of moves*. Initially, Player I chooses some $x_0 \in X$, then Player II chooses $x_1 \in X$, and so on. There also exists a set R of *rules* that imposes restrictions on the moves of the two players. The rules are usually (see for example [Mosch80]) modeled by a tree R on X: the game must proceed along some branch of R, otherwise the first player who gets outside R looses. The rules of the game will usually be defined by putting down restrictions on the choice of x_n that depend on the preceding moves $x_0, \ldots, x_{n-1}$. The tree R is then obtained as:

$$\langle x_0, \ldots, x_{n-1}\rangle \text{ is a path in } R \Leftrightarrow \text{for each } i < n,\ x_i \text{ is allowed by the restrictions}$$

Additionally, we assume the existence of a set D, called the *set of rewards*, which models the potential profit that a player will have after winning the game. Finally, we consider a function Φ, called the payoff function, which calculates the reward that the winner of a play of the game will get. Usually, the definition of Φ depends on the set of rules R in the sense that if during a play one of the players first breaks the rules, then this should be reflected by the value that Φ returns for that play. The above notions are formalized as follows:

Definition 10. *An infinite game of perfect information is a quadruple $\Gamma = (X, R, D, \Phi)$, where:*

- *X is a nonempty set, called the set of moves for Players I and II.*
- *R is a tree on X (usually implicitly specified by a set of rules) which imposes restrictions on the moves of the two players.*
- *D is a linearly ordered set called the set of* rewards*, with the property that for all $S \subseteq D$, lub(S) and glb(S) belong to D.*
- *$\Phi : X^\omega \to D$ is the* payoff function *of the game.*

Based on the set of moves X of a game, we define two sets $\text{Strat}^{I}(\Gamma)$ and $\text{Strat}^{II}(\Gamma)$ which correspond to the set of strategies for Player I and Player II respectively. A strategy $\sigma \in \text{Strat}^{I}(\Gamma)$ assigns a move to each even length partial play of the game; similarly for $\tau \in \text{Strat}^{II}(\Gamma)$ and odd length partial plays.

Definition 11. *Let $\Gamma = (X, R, D, \Phi)$ be a game. Define the set of strategies for Player I and Player II respectively to be the sets:*

- $\mathit{Strat}^I(\Gamma) = (\bigcup_{n<\omega} X^{2n}) \to X$
- $\mathit{Strat}^{II}(\Gamma) = (\bigcup_{n<\omega} X^{2n+1}) \to X$

Definition 12. *Let Γ be a game and let $\sigma \in \mathit{Strat}^I(\Gamma)$ and $\tau \in \mathit{Strat}^{II}(\Gamma)$. We define the following sequence:*

$$\begin{aligned} s_0 &= \sigma(\langle\rangle) \\ s_{2i} &= \sigma(\langle s_0, s_1, \ldots, s_{2i-1}\rangle) \\ s_{2i+1} &= \tau(\langle s_0, s_1, \ldots, s_{2i}\rangle) \end{aligned}$$

A play of the game determined by the strategies σ and τ is the infinite sequence $\langle s_0, s_1, s_2, \ldots\rangle$*. The s_i's will be called the* moves of the play*.*

Given two strategies $\sigma \in \mathrm{Strat}^I(\Gamma)$ and $\tau \in \mathrm{Strat}^{II}(\Gamma)$, we will often write $\sigma \star \tau$ for the play determined by these two strategies. Given a play s, we will say that a player *first breaks the rules in s* if the first move in s that does not conform to the rules of the game is played by that particular player. A play s will be called *legal* if all its moves conform to the rules of the game. We should emphasize here that since we are dealing with infinite games, we will insist that always a play of the game is infinite. In other words, a play continues even after one of the two players has broken the rules; however, the moves played after this point are completely irrelevant with respect to the outcome of the play. Therefore, if one of the players first breaks the rules during a play, then this particular player will lose (no matter what moves are played afterwards).

Until now we have focused on particular plays of a game. We would like to have a notion that gives us the outcome of the whole game provided that Player I tries his best to maximize the result while Player II tries her best to minimize it. Moreover, we would like that during this process, each player can decide for his (her) best strategy, independently of the corresponding choice of the other player. This idea is captured by *determinacy*:

Definition 13 (Determinacy). *Let $\Gamma = (X, R, D, \Phi)$ be a game. Then Γ is* determined *with* value v *if:*

$$\mathit{glb}_{b\in B}\mathit{lub}_{a\in A}\Phi(a\star b) = \mathit{lub}_{a\in A}\mathit{glb}_{b\in B}\Phi(a\star b) = v$$

Determinacy is a very important notion which is in general not straightforward to establish. In fact, one can define infinite games that are not determined. For the game we are considering here, we demonstrate its determinacy in Section 5.

Actually, for our game, one can intuitively understand what determinacy means and be convinced that it indeed holds. Given a string $w \in L(G)$, Player I can design a strategy for proving this fact which succeeds against any strategy of Player II (i.e., Player I doesn't have to worry about how Player II is going to try to reject the membership of w in $L(G)$). Symmetrically, given a string $w \notin L(G)$, Player II can design a strategy for proving this fact which succeeds against any strategy of Player I.

4.2 A Formal Definition of the Game

Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, let $A \in N$ and let $w \in \Sigma^*$. We will define the perfect information game $\Gamma_{G_A^w} = (X, R, D, \Phi)$. The following preliminary definition will be used:

Definition 14. *Let $u \in \Sigma^*$. Then, a* partition π *of u of length n, is a sequence $\langle u_1, \ldots, u_n \rangle$ such that $u_i \in \Sigma^*, 1 \leq i \leq n$, and $u_1 \cdots u_n = u$.*

We will refer to the i-th element of a partition π as $\pi(i)$. Similarly, given $\alpha \in (\Sigma \cup N)^+$, we will write $\alpha(i)$ for the i'th symbol of α.

The game $\Gamma_{G_A^w}$ can now be formally defined as follows:

R1. The first move of Player I is $(A \rightarrow l_1 \& \cdots \& l_m, w)$, where $A \rightarrow l_1 \& \cdots \& l_m$ is a rule of G.

R2. If the previous move is $(B \rightarrow l_1 \& \cdots \& l_m, u)$, then the next move is (l_i, u), where $1 \leq i \leq m$.

R3. If the previous move is (B, u), the next move is $(B \rightarrow l_1 \& \cdots \& l_m, u)$, where $B \rightarrow l_1 \& \cdots \& l_m$ is a rule of G.

R4. If the previous move is $(\neg\alpha, u)$, then the next move is (α, u). A move of this form will be called a *role-switch*.

R5. If the previous move is (α, u), where α contains at least one nonterminal and $|\alpha| \geq 2$, then the next move is of the form (α, π), where π is a partition of u of length $|\alpha|$.

R6. If the previous move is (α, π), then the next move is $(\alpha(i), \pi(i))$, where $1 \leq i \leq |\alpha|$.

R7. If the previous move is (u, u), then the next move is $(\#, \#)$.

We should repeat at this point that since we are dealing with *infinite games*, a play continues even after one of the two players has broken the above rules. However, the moves beyond this point will be irrelevant to the outcome of the play. Notice also that there does not exist any legal response to the move $(\#, \#)$; after this move, the player whose turn it is to play, can only perform an illegal move. Similarly, one can check that there does not exist any legal response to a move of the form (u_1, u_2) where $u_1, u_2 \in \Sigma^*$ and $u_1 \neq u_2$.

The rules given above, implicitly specify the tree R of the game. The strategies of the two players are as defined in Definition 11. Consider now the set of rewards. We define $D = \{0, \frac{1}{2}, 1\}$. In other words, a play of the game can be assigned the value 0 (this means that Player II has won the play), the value 1 (Player I has won), or the value $\frac{1}{2}$ (the result is a tie). It remains to formally define the payoff function. The following definitions are needed:

Definition 15 (True-play, False-play). *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar, $w \in \Sigma^*$ and $A \in N$, and let s be a play of the corresponding game $\Gamma_{G_A^w}$. Then, s is called a* true-play *if either Player II first breaks the rules in s or s is a legal play that contains an odd number of role-switches. Similarly, s is called a* false-play *if either Player I first breaks the rules in s or s is a legal play that contains an even number of role-switches.*

The payoff function is defined as follows:

$$\Phi(s) = \begin{cases} 1, & \text{if } s \text{ is a true-play} \\ 0, & \text{if } s \text{ is a false-play} \\ \frac{1}{2}, & \text{otherwise} \end{cases}$$

This completes the formal presentation of the game. It should be noted at this point that since conjunctive and context-free grammars are subcases of Boolean grammars, the game (actually in a simpler form) is also applicable to them.

5 Equivalence to the Well-Founded Semantics

There still remains a crucial issue that needs to be clarified in order for the game for Boolean grammars to be "well-defined". We still have not argued regarding the *determinacy* of the game nor have we investigated the relationship of the game to the well-founded semantics of Boolean grammars [KNR06]. For infinite games that are win-lose (i.e., no ties), there exists a well-known result, namely *Martin's theorem* [Mar75], which can be used to establish determinacy in most practical cases. In [GRW08], based on Martin's theorem, a criterion is defined that ensures that certain three-valued games are determined. This criterion presupposes the use of the theory of Borel sets (see [GRW08] for details).

In the following, we circumvent the use of Martin's theorem by demonstrating *at the same time* both the determinacy of the game *and* its equivalence to the well-founded model. Our new proof can also be adapted to work for the case of logic programs.

Theorem 4. *Let $G = (\Sigma, N, P, S)$ be a Boolean grammar and M_G be its well-founded model. For every $A \in N$ and $w \in \Sigma^*$, the game $\Gamma_{G_A^w}$ is determined with value $M_G(A)(w)$.*

Proof. (Outline) We start by defining a refinement $\Gamma'_{G_A^w}$ of the standard game $\Gamma_{G_A^w}$ for Boolean grammars. The idea is that since the well-founded model is defined in stages [KNR06], we would like the refined game to possess a notion of level as-well; this will help us establish the equivalence of the two semantic approaches by using an induction on the level numbers.

In the refined game $\Gamma'_{G_A^w}$, the payoff of a given play will depend on the number of role-switches that have taken place. Intuitively, given two true-plays (respectively, false-plays) that have a different number of role-switches, we would like the one that has the less number of such switches to result to a better payoff for Player I (respectively, Player II). For this purpose, the new set D' of rewards has an infinite number of members in the form of pairs, that are ordered as follows:

$$\underbrace{(0,0) < (0,1) < \cdots < (0,i) < \cdots}_{\text{False Plays}} < \underbrace{\frac{1}{2}}_{\text{Tie}} < \underbrace{\cdots < (1,i) < \cdots < (1,1) < (1,0)}_{\text{True Plays}}$$

Consider now a play of the game. If the play is a true-play (respectively false-play), the payoff is going to be $(1,i)$ (respectively $(0,i)$), where i is the number of role switches of the form $(\neg\alpha, u)$ that have taken place in the maximum legal initial part of the play, such that in the third move that follows the role-switch, which is of the form $(\alpha(i), \pi(i))$, $\alpha(i)$ in not a terminal symbol. Notice that a role switch which is excluded by the previous statement, can only appear at the final legal moves of a play.

Subsequently, we define a refinement M'_G of M_G as follows:

$$M'_G(A)(w) = \begin{cases} (1,i), & \text{if } M_i(A)(w) = \frac{1}{2} \text{ and } M_{i+1}(A)(w) = 1 \\ (0,i), & \text{if } M_i(A)(w) = \frac{1}{2} \text{ and } M_{i+1}(A)(w) = 0 \\ \frac{1}{2}, & \text{if } M_G(A)(w) = \frac{1}{2} \end{cases}$$

Similarly to three-valued interpretations used in the definition of well-founded semantics [KNR06], M'_G can be extended to apply to literals (i.e., sequences of terminal and non-terminal symbols and their negations) and conjunctions of literals:

- For every $\alpha \in \Sigma \cup \{\epsilon\}$ and for every $w \in \Sigma^*$, $M'_G(\alpha)(w) = (1,0)$ if $w = \alpha$ and $M'_G(\alpha)(w) = (0,0)$ otherwise.
- For every $\alpha = \alpha_1 \cdots \alpha_n \in (\Sigma \cup N)^*$ with $n \geq 2$, and for every $w \in \Sigma^*$, $M'_G(\alpha)(w) = \max_{\substack{(w_1,\ldots,w_n):\\ w=w_1\cdots w_n}} (\min_{1\leq j\leq n} M'_G(\alpha_j)(w_j))$.
- For every $l = \neg\alpha$, with $\alpha \in (\Sigma \cup N)^*$, and for every $w \in \Sigma^*$, if $M_0(\alpha)(w) = u \neq \frac{1}{2}$, then $M'_G(l)(w) = (1-u, 0)$; otherwise, if $M'_G(\alpha)(w) = (v,i)$, $v \in \{0,1\}$, then $M'_G(l)(w) = (1-v, i+1)$; otherwise $M'_G(l)(w) = \frac{1}{2}$.
- For every conjunction of literals of the form $l_1 \& \ldots \& l_n$ and for every $w \in \Sigma^*$, $M'_G(l_1 \& \cdots \& l_n)(w) = \min\{M'_G(l_1)(w), \ldots, M'_G(l_n)(w)\}$.

In the above cases, min and max are defined with respect to the ordering we have imposed on the domain of refined rewards D'. In order to prove the theorem it suffices to demonstrate that for every $A \in N$ and for every $w \in \Sigma^*$, the refined game $\Gamma'_{G_A^w}$ is determined with value $M'_G(A)(w)$. For this purpose, we need to define a strategy $\hat{\sigma}$ for Player I and a strategy $\hat{\tau}$ for Player II, which will help us establish the determinacy of the game $\Gamma'_{G_A^w}$ (as it will become clear later on, these two strategies are the "best possible" ones for the two players).

We start by defining a strategy $\hat{\sigma}$ for Player I. We specify the move of Player I in all cases in which he may have more than one legal choices:
Case 1: The previous move is (B, u), with $B \in N$ and $u \in \Sigma^*$. Then he plays a move (R, u), where $R = B \to \alpha_1 \& \cdots \& \alpha_m \& \neg\beta_1 \& \cdots \& \neg\beta_n$ is a rule in P such that $M'_G(\alpha_1 \& \cdots \& \alpha_m \& \neg\beta_1 \& \cdots \& \neg\beta_n)(u) = M'_G(B)(u)$ selected as follows:

- if $M'_G(B)(u) = (1,i)$ and $r > 0$ is the least integer such that $\Theta_{M_i}^{\uparrow r}(B)(u) = 1$, then R should have the property that for every j, $1 \leq j \leq m$, $\Theta_{M_i}^{\uparrow r-1}(a_j)(u) = 1$ and for every k, $1 \leq k \leq n$, $M_i(b_k)(u) = 0$.
- if $M'_G(B)(u) = (0,0)$, then R is any rule with head B.
- if $M'_G(B)(u) = (0,i)$, with $i > 0$, and $r > 0$ is the least integer such that $\Theta_{M_{i-1}}^{\uparrow r}(B)(u) = \frac{1}{2}$, then R should have the property that for every j, $1 \leq j \leq m$, $\Theta_{M_{i-1}}^{\uparrow r-1}(a_j)(u) \geq \frac{1}{2}$ and for every k, $1 \leq k \leq n$, $M_{i-1}(b_k)(u) \leq \frac{1}{2}$.

- if $M'_G(B)(u) = \frac{1}{2}$, and $r > 0$ is the least integer such that $\Theta^{\uparrow r}_{M_G}(B)(u) = \frac{1}{2}$, then R should have the property that for every j, $1 \le j \le m$, $\Theta^{\uparrow r-1}_{M_G}(a_j)(u) \ge \frac{1}{2}$ and for every k, $1 \le k \le n$, $M_G(b_k)(u) \le \frac{1}{2}$.

Case 2: The previous move is (α, u), where α contains at least one element in N and $|\alpha| \ge 2$. In this case, Player I plays (α, π), where π is a partition of u, such that $M'_G(\alpha(i))(\pi(i)) \ge M'_G(\alpha)(u)$ for all i, which is selected by considering analogous subcases as in Case 1 above.
Case 3: The previous move is $(B \rightarrow l_1\& \cdots \&l_m, w)$. If there exists a positive l_i such that $M'_G(l_1\& \cdots \&l_m)(u) = M'_G(l_i)(u)$, he plays (l_i, u). Otherwise, there exists a negative literal l_j such that $M'_G(l_1\& \cdots \&l_m)(u) = M'_G(l_j)(u)$ and he plays (l_j, w).
Case 4: The previous move is (α, π). In this case he selects an i such that $M'_G(\alpha(i))(\pi(i))$ is minimum and plays $(\alpha(i), \pi(i))$.

Using exactly the same selections for each case, we can define a strategy $\hat{\tau}$ for Player II.

Now, we can demonstrate by an induction on i, that if $M'_G(A)(w) = (v, i)$, $v \in \{0, 1\}$, then it holds that:

1. For every strategy τ of Player II, $\Phi(\hat{\sigma} \star \tau) \ge (v, i)$.
2. For every strategy σ of Player I, $\Phi(\sigma \star \hat{\tau}) \le (v, i)$.

Notice that, although the two players according to $\hat{\sigma}$ and $\hat{\tau}$ react in exactly the same way in every move, the inequalities have opposite directions due to the fact that the roles of the two players are initially different. Using the above inequalities we obtain:

$$\begin{aligned}(v, i) &\le glb_{\tau \in B}\ \Phi(\hat{\sigma} \star \tau)\\ &\le lub_{\sigma \in A}\ glb_{\tau \in B}\ \Phi(\sigma \star \tau)\\ &\le glb_{\tau \in B}\ lub_{\sigma \in A}\ \Phi(\sigma \star \tau)\\ &\le lub_{\sigma \in A}\ \Phi(\sigma \star \hat{\tau})\\ &\le (v, i)\end{aligned}$$

which implies that the refined game $\Gamma'_{G^w_A}$ is determined with value (v, i).

Using the fact that the determinacy result holds for every value (v, i), with $v \in \{0, 1\}$, we can show that if $M'_G(A)(w) = \frac{1}{2}$, then:

1. For every strategy τ of Player II, $\Phi(\hat{\sigma} \star \tau) \ge \frac{1}{2}$.
2. For every strategy σ of Player I, $\Phi(\sigma \star \hat{\tau}) \le \frac{1}{2}$.

which implies in an analogous way as before that the refined game $\Gamma'_{G^w_A}$ is determined with value $\frac{1}{2}$. □

6 Conclusions

We have presented an infinite game semantics for Boolean grammars and have demonstrated that it is equivalent to the well-founded semantics of this type of grammars. The simplicity of the new semantics stems mainly from its anthropomorphic flavor. In this respect, it differs from the well-founded semantics whose construction requires a more heavy mathematical machinery. We believe that these two semantical approaches can be used in a complementary way in the study of Boolean grammars. In our opinion, the game-theoretic approach will prove useful in establishing the correctness of meaning-preserving transformations for Boolean grammars. The reasoning in such a case can proceed as follows. Consider a Boolean grammar G and its transformed version G'. We can verify that the meaning of a nonterminal A in G coincides with the meaning of A in G' if for every string w, Player i has a winning strategy in the game $\Gamma_{G^w_A}$ iff Player i has a winning strategy in game $\Gamma_{G'^w_A}$. On the other hand, the well-founded semantics appears to be more useful in computing the meaning of *specific* grammars. This is due to the iterative-inductive flavor of the well-founded approach (see [NR08] for an example of an iterative computation of the meaning of a Boolean grammar using a procedure that was inspired by the well-founded construction).

References

[GRW08] Galanaki, C., Rondogiannis, P., Wadge, W.: An Infinite-Game Semantics for Well-Founded Negation in Logic Programming. Annals of Pure and Applied Logic 151(2–3), 70–88 (2008)

[GS53] Gale, D., Stewart, F.M.: Infinite Games with Perfect Information. Annals of Mathematical Studies 28, 245–266 (1953)

[KNR06] Kountouriotis, V., Nomikos, C., Rondogiannis, P.: Well-founded semantics for boolean grammars. In: H. Ibarra, O., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 203–214. Springer, Heidelberg (2006)

[Mar75] Martin, D.A.: Borel Determinacy. Annals of Mathematics 102, 363–371 (1975)

[Mosch80] Moschovakis, Y.N.: Descriptive Set Theory. North-Holland, Amsterdam (1980)

[NR08] Nomikos, C., Rondogiannis, P.: Locally Stratified Boolean Grammars. Information and Computation 206(9-10), 1219–1233 (2008)

[Okh01] Okhotin, A.: Conjunctive Grammars. Journal of Automata, Languages and Combinatorics 6(4), 519–535 (2001)

[Okh04] Okhotin, A.: Boolean Grammars. Information and Computation 194(1), 19–48 (2004)

[Okh06] Okhotin, A.: Nine open problems on conjunctive and Boolean grammars. TUCS Technical Report No 794, Turku Centre for Computer Science, Turku, Finland (November 2006)

[OJ07] Jeż, A., Okhotin, A.: Conjunctive Grammars over a Unary Alphabet: Undecidability and Unbounded Growth. In: Diekert, V., Volkov, M.V., Voronkov, A. (eds.) CSR 2007. LNCS, vol. 4649, pp. 168–181. Springer, Heidelberg (2007)

[PP90] Przymusinska, H., Przymusinski, T.: Semantic Issues in Deductive Databases and Logic Programs. In: Banerji, R. (ed.) Formal Techniques in Artificial Intelligence: a Source-Book, pp. 321–367. North-Holland, Amsterdam (1990)

Word Equations with One Unknown

Markku Laine[1,⋆] and Wojciech Plandowski[2,⋆⋆]

[1] Department of Mathematics, University of Turku and
Turku Centre for Computer Science (TUCS), 20014 Turku, Finland
[2] Institute of Informatics, University of Warsaw, Poland

Abstract. We consider properties of the solution set of a word equation with one unknown. We prove that the solution set of a word equation possessing infinite number of solutions is of the form $(pq)^*p$ where pq is primitive. Next, we prove that a word equation with at most four occurrences of the unknown possesses either infinitely many solutions or at most two solutions. We show that there are equations with at most four occurrences of the unknown possessing exactly two solutions. Finally, we prove that a word equation with at most $2k$ occurrences of the unknown possesses either infinitely many solutions or at most $8 \log k + O(1)$ solutions. Hence, if we consider a class $\mathcal{E}_k$ of equations with at most $2k$ occurrences of the unknown, then each equation in this class possesses either infinitely many solutions or a constant number of solutions. Our considerations allow to construct the first truly linear time algorithm for computing the solution set of an equation in a nontrivial class of equations.

Keywords: combinatorics on words, word equations.

1 Introduction

The theory of word equations is a part of combinatorics on words [5,6] and unification theory [1]. It has applications in graph theory [9], constraint logic programming [8], artificial intelligence [4], and formal languages [2].

As in number theory the problems in this theory are simple to explain to a nonexpert and difficult to solve. As an evidence of the difficulty of the problems in the theory of word equations is that not everything is known on word equations with one unknown. Our paper deals with such equations. The satisfiability problem for such equations can be solved in $O(n \log n)$ time where n is the size of the equation [7]. For finite alphabets of fixed size it can be solved in $O(n + \#_X \log n)$ time where $\#_X$ is the number of occurrences of the unknown X in the equation [3]. If $\#_X = \Theta(n)$, then the algorithm is not linear. It is still not known whether there is a linear time algorithm for the problem.

Our main results deal with the structure of the solution set of an equation with one unknown. The set is either finite or infinite. In [7] it was implicitely

[⋆] E-mail: `majlain@utu.fi`.

[⋆⋆] E-mail: `w.plandowski@mimuw.edu.pl`. The research supported by Polish Ministry of Science and Higher Education under grant nr N 206 004 32/0806.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 348–359, 2009.

proved that in the former case it is of size at most $2\log n$ and in the latter case it is a union of a finite set of size at most $2\log n$ and a set of the form $(pq)^+p$ with pq-primitive and q nonempty. We improve the latter case by proving that for such equations the solution set is of the form $(pq)^*p$ with pq-primitive and q nonempty. As an immediate effect of our considerations we obtain a linear time algorithm independent on the size of the alphabet, for finding the solution set of an equation which have infinitely many solutions.

It is quite elementary to find out that equations with at most three occurrences of the unknown can have only zero, one or infinitely many solutions, see Example 1. We show an example of an equation with exactly four occurrences of the unknown which has exactly two solutions. Next we prove that this number is tight. Namely we prove that equations with at most four occurrences of the unknown have either at most two solutions or infinitely many solutions.

At the end we consider equations with at most $2k$ occurrences of the unknown and prove that they have either at most $8\log k+O(1)$ solutions or infinitely many solutions. If we fix k, then this gives a constant which bounds the number of solutions of an equation containing at most $2k$ occurrences of the unknown in case it has finitely many solutions. Our considerations allow one to construct the first linear time algorithm independent of the size of the alphabet, for the satisfiability problem as well as to find the solution set of an equation for a nontrivial class of equations.

2 Basic Notions

For two numbers p and q, $gcd(p,q)$ denotes the greatest common divisor of p and q. Let Σ be any set. The set Σ is called *alphabet*. The elements of Σ are called *letters*. Sequences of letters are called *words*. The empty sequence is called *the empty word* and is denoted by 1.

Let w be a word. The length of w is denoted by $|w|$. A *prefix* of w is a word p such that $w = ps$, for some word s. If $s \neq 1$, then p is called a *proper prefix*. A *suffix* of w is a word s such that $w = ps$, for some word p. If $p \neq 1$, then s is called a *proper suffix*. A *subword* of w is a word u such that $w = pus$, for some words p and s. Denote by $w[i..j]$, for $1 \leq i \leq j \leq |w|$ the subword of w which starts at letter i in w and ends at letter j of w. The i-th letter of w is denoted by $w[i]$. The prefix of w of length k is denoted by $\mathrm{pref}_k\,(w)$. The suffix of w of length k is denoted by $\mathrm{suff}_k\,(w)$.

For a prefix p of w, we denote by $p^{-1}w$ the word s such that $w = ps$. Similarly, for a suffix s of w, we denote by ws^{-1} the word p such that $w = ps$. For two words, u, w, by $u\cdots = w\cdots$ we mean that there are words u', w' such that $uu' = ww'$. Since it means that either u is a prefix of w or vice versa, we say then that the words u and w are *prefix comparable.* Similarly, by $\cdots u = \cdots w$ we mean that there are two words u' and w' such that $u'u = w'w$. Since it means that either u is a suffix of w or vice versa, we say then that the words u and w are *suffix comparable.*

For a word u and a positive integer k, by u^k we mean a word which is a repetition of the word u, k times. If $k = 0$, we define $u^0 = 1$. We say that a word

w is *primitive*, if it is not of the form u^k, for any word u and $k \geq 2$. A primitive word u such that $w = u^k$, for some $k \geq 1$, is called *the primitive root* of w and denoted by $root(w)$. The function *root* is well-defined for nonempty words, since such a word is unique.

By u^ω we mean a right infinite word which consists of the word u repeated infinitely many times. Similarly, by ${}^\omega u$ we mean a left infinite word which consists of the word u repeated infinitely many times. A *period* of a word w is a prefix u of w such that w is a prefix of u^ω. We call a period also the number $|u|$. We say that a period u of a prefix of w *breaks* at position p of w if u is a period of $w[1..p-1]$ and u is not a period of $w[1..p]$. The proof of the next well known proposition can be found in [5,6].

Proposition 1 (Lemma by Fine & Wilf). *Let u and v be periods of a word w satisfying $|u| + |v| \leq |w|$. Then the prefix p of w of length $gcd(|u|, |v|)$ is also a period of w.*

We say that two words u and v *commute* if $uv = vu$. We say that words x and y are *conjugates*, if there are words p, q such that $x = pq$ and $y = qp$.

A *reverse* of a word $w = w[1] \cdots w[k]$ is $w[k] \cdots w[1]$.

A *word equation* e with one unknown X is an equation of the form

$$A_0XA_1X \cdots A_k = XB_0XB_1X \cdots B_l$$

where A_i, $B_i \in \Sigma^*$ and either A_k or B_l is the empty word. A *solution* of a word equation is a word x such that $A_0xA_1x \cdots A_k = xB_0xB_1x \cdots B_l$. Observe that always x is a prefix of A_0^ω.

Example 1. A word equation with one occurrence of X is of the form $X = A_0$. It has exactly one solution $x = A_0$.

A word equation with two occurrences of X is either of the form $XB_0X = A_0$ or of the form $A_0X = XB_0$.

The length argument shows that the only possible length of a solution of the first equation is $\frac{1}{2}(|A_0| - |B_0|)$. Hence, it can have one or zero solutions. The only possible solution is a prefix of appropriate length of A_0.

The second equation is well known in combinatorics on words. It has either zero or infinitely many solutions. In the latter case there are words p, q with pq-primitive and q nonempty such that $A_0 = (pq)^k$, $B_0 = (qp)^k$. Then the solution set is $(pq)^*p$. For any pair of words A_0 and B_0, there is at most one such pair (p, q). Consequently, if pq is primitive and q nonempty, then the solution set of the equation $(pq)^kX = X(qp)^k$ is $(pq)^*p$.

The length argument shows that, if the number of occurrences of X on the left hand side of the equation is different from the number of occurrences of X on the right hand side, then there is at most one solution of such equation. In particular, each equation with odd number of occurrences of X has either zero or one solution. Hence, the only equations we consider next contain the same number of occurrences of X on both sides of the equation.

3 The Family of Equations $u_1Xu_2X = Xv_1Xv_2$

In this section we are interested in equations with finitely many solutions. As we concluded in Section 2 the simplest interesting case of equations with at most four occurrences of X are equations of the form $u_1Xu_2X = Xv_1Xv_2$ or $u_1Xu_2Xu_3 = Xv_1X$. We will prove that such equations have at most two solutions and this boundary is sharp. Since all solutions are prefixes of the same word, then the shortest solution is a prefix of all the other solutions. Let this solution be x. Consider the equation $u_1xXu_2xX = xXv_1xXv_2$ (resp. $u_1xXu_2xXu_3 = xXv_1xX$). It has the same number of solutions as the original equation and, additionally $X = 1$ is a solution of it. This means that we may restrict our considerations to equations such that $X = 1$ is a solution of them. Then, $u_1u_2 = v_1v_2$ (resp. $u_1u_2u_3 = v_1$). If $u_1w = v_1$, then $u_2 = wv_2$, and the equation is of the form $(u_1X)w(v_2X) = (Xu_1)wX(v_2)$ (resp. $(u_1X)u_2(Xu_3) = (Xu_1)u_2(u_3X)$). These split into simpler equations with two occurrences of X and either one or infinitely many solutions.

If $u_1 = v_1w$, then $wu_2 = v_2$ and the equation is of the form $v_1wXu_2X = Xv_1Xwu_2$. Renaming the constant words we obtain the equations of the form $uvXwX = XuXvw$. This is the family of equations we consider in this section.

Example 2. Consider the equation $XbXcbbccbc = bcbbccbXcX$. Since all solutions are prefixes of $(bcbbccb)^\omega$ it is easy to check that all solutions shorter than 6 are 1 and bc. All solutions of length at least 6 start with $bcbbcc$. Hence, the original equation can be split so that the solutions satisfy $Xbbcbbcc = bcbbccbX$. Hence, such solutions are of the form $x = (bcbbccb)^ibcbbcc$. However, if we look at the original equation we see that bc should be a suffix of x. Hence, there are no such solutions.

Similarly we prove that the solution set of the equation $XbXcbc = bcbXcX$ is $\{1, bc\}$.

Lemma 1. *If pq and $p'q'$ are primitive with q, q' nonempty, and $(pq)^ip = (p'q')^jp'$, and $(pq)^np = (p'q')^mp'$, where either i, $j \geq 2$, or $n \neq i$ or $m \neq j$, then $p = p'$, $q = q'$, $i = j$ and $n = m$.*

Proof. Follows from Proposition 1.

In the following, the equation

$$XuXvw = uvXwX \tag{1}$$

is assumed to have a finite solution set $S = \{s_0 = 1, \cdots, s_{k-1}\}$, where s_{i-1} is a proper prefix and a proper suffix of s_i, for $i = 1, \cdots, k-1$. The solutions imply the equalities, see Fig. 1

$$s_iuv_i = uvs_i \qquad \text{and} \qquad s_ivw = v_iws_i, \tag{2}$$

where $v_i = \mathrm{pref}_{|v|}(s_iv) = \mathrm{suff}_{|v|}(vs_i)$. On the other hand, a common solution s of equations

$$Xuv_i = uvX \qquad \text{and} \qquad Xvw = v_iwX \tag{3}$$

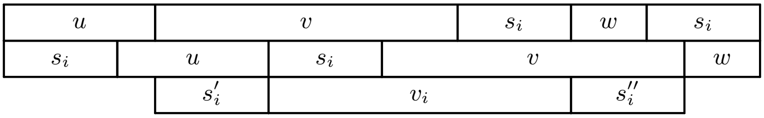


Fig. 1. The dependences between words s_i, s_i', s_i'', v_i and other words. The letters in the same columns are the same. The relative position of boundaries between words in first two rows may be different.

is also a solution of (1), since $susvw = suv_iws = uvsws$. Moreover, we also get the equalities

$$s_iu = us_i' \qquad \text{and} \qquad s_i''w = ws_i, \tag{4}$$

where $s_i' = \text{suff}_{|s_i|}(s_iu) = \text{pref}_{|s_i|}(vs_i)$ and $s_i'' = \text{pref}_{|s_i|}(ws_i) = \text{suff}_{|s_i|}(s_iv)$, and finally

$$s_i'v_i = vs_i \qquad \text{and} \qquad s_iv = v_is_i''. \tag{5}$$

Observe, that all the words v_i have the same length as v and $|s_i| = |s_i'| = |s_i''|$, for all i. By symmetry, we may assume, that $|u| \geq |w|$.

Lemma 2. *The length of any solution s_i of equation (1) is less than $2|uv|$.*

Proof (sketch). Assume the contrary. By Lemma 1, the solution set then contains an infinite set $(pq)^*p$.

Lemma 3. *Word s_i, is the only solution for the equation pair (3).*

Proof. Consequence of Lemma 1.

Corollary 1. *If $i \neq j$, then $v_i \neq v_j$.*

Corollary 2. *Equation (1) has at most one solution of length at least $|v|/2$.*

Proof. Assume, on the contrary, that $|v|/2 \leq |s_i|$, for $i < k-1$. But then

$$\begin{aligned} v_i &= \text{pref}_{\lfloor |v|/2 \rfloor}(s_iv)\,\text{suff}_{\lceil |v|/2 \rceil}(vs_i) \\ &= \text{pref}_{\lfloor |v|/2 \rfloor}(s_{i+1}v)\,\text{suff}_{\lceil |v|/2 \rceil}(vs_{i+1}) = v_{i+1}. \end{aligned}$$

Lemma 4. *For all $i > 0$, $s_i' \neq s_i \neq s_i''$.*

Proof. We may assume, on the contrary, that $s_i' = s_i$, since the case $s_i'' = s_i$ is symmetric. We get from equalities (5), that also $s_i'' = s_i$. Thus, $s_i, s_i', s_i'', u, w \in r^+$, for some primitive r. Hence, $s_ius_ivw = uvs_iws_i$ and also $v \in r^+$. Hence, all the words in r^+ are solutions of (1), a contradiction.

Lemma 5. *If $k \geq 3$, then $|s_2| > |u| + |s_1|$.*

Proof (sketch). Leaving from the counter assumption it is not hard to deduce the contradiction that $s_1' = s_1$.

We use the following well known fact in combinatorics on words.

Observation 1. $x^n z = zy^n$, for some $n \geq 1 \iff xz = zy$

Our next lemma gives the necessary and sufficient condition for an equation to have infinite number of solutions.

Lemma 6. *The solution set S of* (1) *is infinite, iff uv commutes with vw.*

Proof. If $uv, vw \in z^+$, then any word $s \in z^*$ is a solution, since $susvw = suvws = uvsws$. The proof for the other direction is more intriguing, but we do not actually need this direction and omit the proof to save space.

Lemma 7. *Let x and x' be conjugates such that $x \neq x'$. Then the word xx' is primitive.*

Proof. This follows from Lyndon-Schützenberger lemma, see Observation 2.

A period of a word is called *primitive* if it is a primitive word.

Lemma 8. *There is at most one non-empty solution s_i such that $|s_i| \leq |v|/2$.*

Proof. We have $s_i' s_i v = v s_i s_i''$, see Fig. 1. Then $(s_i s_i')(s_i v s_i) = (s_i v s_i)(s_i'' s_i)$. By Lemma 7, $s_i s_i'$ is primitive, and, consequently, the word $s_i v s_i$ has a primitive period of length $2|s_i|$. Suppose that s and t with $|s| < |t| \leq |v|/2$ are solutions. Then the word svs would have two primitive periods of lengths $2|s|$ and $2|t|$, since svs is a subword of tvt, which has a primitive period of length $2|t|$. Also $|svs| \geq 2|s| + 2|t|$, and by Proposition 1, $|s| = |t|$ raising a contradiction.

Summing this together we have at most one non-empty solution such that $|x| \leq |v|/2$. Together with the empty solution and one possible solution of length at least $|v|/2$ we have at most three solutions.

We proceed to show that $|S| \leq 2$, by assuming the contrary and then gradually closing the window of possible lengths for s_2. We already know, that $|s_1| < |v|/2 < |s_2| < 2|uv|$. Thus, we have $v = s_1' s_1 t_1 = t_1 s_1 s_1''$, $v_1 = s_1 t_1 s_1$, see Fig. 2. Since s_1 is a prefix and a suffix of s_2 and $|s_1| \leq |v| = |v_2|$, then, by (5), s_1 is a prefix and a suffix of v_2. Hence, $v_2 = s_1 t_2 s_1$, for some non-empty word t_2 satisfying $|t_1| = |t_2|$.

By Lemma 4, Lemma 6 and Corollary 1 we know, that none of the three conditions 1) $s_1' = s_1$ ($s_1'' = s_1$); 2) uv commutes with vw; and 3) $t_1 = t_2$ is possible. The proofs of each one of the following lemmas start from a counter assumption and then one of these three contradictions is deduced. Unfortunately, we have to omit the rather technical proofs to save space.

Lemma 9. $|s_2| > 2|s_1|$

Now, we may write $s_2 = s_1 \gamma s_1$. Since $us_2' = s_2 u$ and s_1 is a suffix of s_2, we have that $us_1' = s_1 u$ is a suffix of s_2', by Lemma 5. Thus, $s_2' = s_1' \gamma' s_1'$, for some γ'. Symmetrically, $s_2'' = s_1'' \gamma'' s_1''$. Observe also, that since $v = s_1' s_1 t_1$ (resp. $v = t_1 s_1 s_1''$) and v and s_2' are prefix comparable (resp. v and s_2'' are suffix comparable), then s_1 is a prefix of $\gamma' s_1'$ (resp. a suffix of $s_1'' \gamma''$).

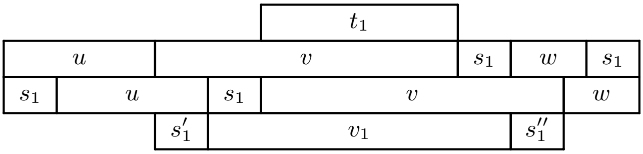


Fig. 2. Illustration for dependencies between word t_1 and words defined earlier

Lemma 10. *$|\gamma| > |t_1|$. Hence, $|s_2| > |v|$.*

Lemma 11. *$|\gamma| > |t_1| + |s_1|$. Hence, $|s_2| > |v| + |s_1|$.*

Lemma 12. *$|s_2| > |uv|$.*

The next Lemma finally shows, that the assumption of three solutions is not possible. It also improves Lemma 2.

Lemma 13. *The length of any solution s_i of equation (1) is at most $|uv|$.*

As an immediate consequence of our considerations we obtain.

Theorem 1. *Each equation with at most four occurrences of X possesses either at most two or infinitely many solutions. There are equations with at most four occurrences of X which possesses exactly two solutions.*

4 Infinite Solution Set

We will use several well known facts in combinatorics on words. The last one of them is the famous Lyndon-Schützenberger lemma, see the proof in [5].

Observation 2. *1. Let u and v with $|u| > |v|$ be two periods of a word w. Then $|u| - |v|$ is a period of $v^{-1}w$.*
2. If p is a period of w and p is primitive and p is a suffix of w, then $w = p^l$, for some $l \geq 1$.
3. Let x be a prefix of yx' where x' is a prefix of x. Then y is a period of yx'.
4. Suppose $x^i y \cdots = y^j x \cdots$ with $i, j \geq 1$. Then x and y are powers of the same word.
5. Suppose x and y satisfy a nontrivial identity. Then x and y are powers of the same word. In particular, if x and y commute, then they are powers of the same word.
6. If $x^n y^m = z^k$ where n, m, $k \geq 2$, then x, y and z are powers of the same word.

Let e be a nontrivial equation to solve. We assume that equations in this section contain infinitely many solutions. Then e is of the form $A_0XA_1 \cdots A_kX = XB_0XB_1X \cdots B_k$ with $k \geq 0$ and $A_0 \neq 1$ or of the form $A_0XA_1 \cdots XA_k =$

$XB_0X \cdots B_{k-2}X$ with $k \geq 2$ and $A_0 \neq 1$. An equation in the first form is called *an equation of type (1)*. An equation in the second form is called *an equation of type (2)*.

An equation of type (1) can have a solution only if $\sum_{i=0}^{k} |A_i| = \sum_{i=0}^{k} |B_i|$. An equation of type (2) can have a solution only if $\sum_{i=0}^{k} |A_i| = \sum_{i=0}^{k-2} |B_i|$. The equation e can be split into two equations which are equivalent to e if for some $0 \leq l < k$: $\sum_{i=0}^{l} |A_i| \leq \sum_{i=0}^{l} |B_i|$. Let l be the minimal one. Then the equation of type (1) splits into two ones. The first one is $A_0XA_1 \cdots XA_lX = XB_0XB_1 \cdots XB_{l-1}XB'$ where B' is a prefix of B_l of length

$$\sum_{i=0}^{l} |A_i| - \sum_{i=0}^{l-1} |B_i|.$$

The second equation is $A_{l+1}XA_{l+2}X \cdots A_kX = B''XB_{l+1} \cdots XB_k$ where $B_l = B'B''$. The equation of type (2) splits into two ones. The first one is

$$A_0XA_1 \cdots XA_lX = XB_0XB_1 \cdots XB_{l-1}XB'$$

where B' is a prefix of B_l of length $\sum_{i=0}^{l} |A_i| - \sum_{i=0}^{l-1} |B_i|$. The second equation is $A_{l+1}XA_{l+2}X \cdots XA_k = B''XB_{l+1} \cdots B_{k-1}X$ where $B_l = B'B''$.

The equation of type (2) can always be splitted. Additionally, for both types of equations, the first equation of two cannot be splitted. Using this splitting technique we can replace the equation e by an equivalent system of equations. Each of these equations is of the form $A_0XA_1 \cdots A_kX = XB_0XB_1X \cdots B_k$ with $k \geq 0$ and satisfies additionally $\sum_{i=0}^{l} |A_i| > \sum_{i=0}^{l} |B_i|$, for $0 \leq l < k$. Such an equation is called *irreducible*.

We want to prove that, if e has infinitely many solutions, then the solution set is of the form r^*, for some primitive r, or of the form $(pq)^*p$, for some primitive pq and nonempty p and q. Since each solution of e is a solution of all equations in the system, each irreducible equation in the system has infinitely many solutions. If we prove the theorem for irreducible equations, we prove the theorem for all equations. Indeed, if $|(p_1q_1)^*p_1 \cap (p_2q_2)^*p_2| = \infty$, for some p_i, q_i, where q_i is nonempty and p_iq_i is primitive, then $(p_1q_1)^*p_1 = (p_2q_2)^*p_2$.

If $k = 0$, then there is nothing to prove. Assume then that $k \geq 1$.

We define now *elementary* equations. An equation e is *elementary* if

$$\sum_{i=0}^{l} |A_i| > \sum_{i=0}^{l-1} |B_i|, \text{ for each } 0 \leq l \leq k.$$

Observe here that each irreducible equation is elementary but not vice versa. Indeed, the equation $aXbX = XaXb$ is elementary and not irreducible. It can be split into two equations $aX = Xa$ and $bX = Xb$.

Lemma 14. *Assume that an elementary equation has infinitely many solutions of the form r^i where r is primitive. Then all words of the form r^i, for $i \geq 0$, are solutions of the equation. Moreover $A_0 = r^m$, for some $m \geq 1$ and $B_k = r^t$, for some $t \geq 1$. Additionally, r is a prefix of B_ir, for $0 \leq i \leq k$, and, for $1 \leq i \leq k$, $B_{i-1}r^j = r^sA_i$, for some j, $s > 0$.*

Proof (sketch). Consider the words $w_{i,l} = A_0 r^i A_1 r^i \cdots A_l r^i$, for $i \geq 0$ and $0 \leq l \leq k$ and $u_{i,l} = r^i B_0 r^i \cdots r^i B_{l-1} r^i$, for $i \geq 0$ and $0 \leq l \leq k$. We have

$$|w_{i,l}| - |u_{i,l}| = \sum_{s=0}^{l} |A_s| - \sum_{s=0}^{l-1} |B_s|.$$

Observe here that this difference does not depend on i. Since the equation is elementary $w_{i,l}$ is longer than $u_{i,l}$, for all l.

It can be proved by induction on l that, for each i, $u_{i,l}$ is a prefix of $w_{i,l}$ and $u_{i,l}^{-1} w_{i,l}$ is of the form r^s, for $s \geq 1$. As we showed s does not depend on i. The result then follows easily.

Lemma 15. *Assume that an elementary equation has infinitely many solutions of the form $(pq)^i p$ for nonempty p, q and primitive pq. Then all words of the form $(pq)^i p$, for $i \geq 0$, are solutions of the equation. Moreover $A_0 = (pq)^m$, for some $m \geq 1$ and $B_k = (qp)^t$, for some $t \geq 1$. Additionally, for each $0 \leq i \leq k$, qp is a prefix of $B_i pq$, and pq is a suffix of qpA_i.*

Proof (sketch). Let e be the equation. Make a substitution which replaces the variable X by the word pX. New equation e' is also elementary. The result follows, when we apply Lemma 14 to e'.

Let e be an irreducible equation. Then $|B_0| < |A_0|$. Let B' be a prefix of A_0 of length $|A_0| - |B_0|$. We distinguish two types of equations: equations where $A_0 = B_0 B'$ and equations where $A_0 \neq B_0 B'$. The equations of the first type are called of type r and the equations of the second type are called of type (p, q) where r, p, q are nonempty words which will be determined later by A_0 and $B_0 B'$.

We divide solutions of the equation e into two groups: long solutions and short solutions. Long solutions are those which are not shorter than $|B'|$. Short solutions are those which are shorter than $|B'|$. Since all solutions are prefixes of A_0^ω, then long solutions start by B'. Hence they all satisfy $A_0 X = X B_0 B'$ where B' comes from the prefix of the X which stays in equation just to the right of B_0. If our equation has a long solution then A_0 and $B_0 B'$ are conjugates. Hence, either $B_0 B' = A_0 = r^m$, for some primitive r and $m \geq 1$, or $A_0 = (pq)^m$, $B_0 B' = (qp)^m$, for $m \geq 1$ and some nonempty p, q such that pq is primitive. In the former case the equation is of type r. In the latter case the equation is of type (p, q).

As a result of our previous considerations we have

Lemma 16. *Assume that the equation e has infinitely many solutions. If the equation is of type r, then all words in r^* are solutions for it. If the equation is of type (p, q), then all words in $(pq)^* p$ are solutions for it. In both cases there are no other long solutions.*

By Lemma 16 it is enough to prove that there are no short solutions except the ones mentioned in the lemma.

In the following sub sections the meaning of B', m, t, A_i, B_i are the same as in this section.

4.1 Equations of Type r

Theorem 2. *Let e be an equation of type r with infinitely many solutions. Then the solution set of e is r^*.*

Proof (sketch). Observe first that, if x is a short solution, then xB_0 is a proper prefix of A_0. Since the equation is of type r, we have $r^m = A_0 = B_0B'$ where B' is a prefix of r^m. The proof gets divided into three cases depending on the lengths of B_0 and B'. First we show that if either $|B_0| \geq |r|$, or $B_0 \neq 1$ and $|B'| \geq |r|$, then the solution set is r^*.

In the second case $0 < |B_0| < |r|$ and $|B'| < |r|$. Then either we get that $x = 1$ is the only short solution or we get the contradiction that r is not primitive. In the last case we have $B_0 = 1$. This time either the short solution is in r^+, or it turns out that x is a period of r, which implies the same contradiction that r is not primitive.

4.2 Equations of Type (p, q)

We start by proving a corollary of Theorem 2.

Corollary 3. *Let e be an equation of type (p, q) with infinitely many solutions. Then all solutions of e which are not shorter than p form the set $(pq)^*p$.*

Proof. Let e' be an equation which can be obtained from e after replacing the unknown X by the word Xp. Then e' is splitted into a set of equations of type pq with infinitely many solutions. Hence, the solution set of e' is $(pq)^*$. Since each solution of e which is not shorter than p is of the form xp, such solutions form the set $(pq)^*p$.

By Corollary 3 it is enough to prove that there are no short solutions which are shorter than p. To prove this we need a technical lemma. To save space we omit its rather long proof.

Lemma 17. *Let e be an equation of type (p, q) with infinitely many solutions. Then the empty word is not a solution for e.*

We use Lemma 17 to prove our main result of this section.

Theorem 3. *Let e be an equation of type (p, q) with infinitely many solutions. Then all solutions form the set $(pq)^*p$.*

Proof (sketch). By Corollary 3 it is enough to prove that the equation e has no solutions shorter than p. Suppose on the contrary, that a prefix of p, x with $|x| < |p|$, is a solution of e. Replace in e each occurrence of X by xX obtaining an equation e'. The equation e' has infinitely many solutions and the empty word is a solution of it. Now apply splitting to the equation e'. We obtain an irreducible equation e'' and another equation. It turns out that e'' does not have 1 as its solution, a contradiction.

5 A Family of Equations with Constant Number of Occurrences of X

The following proposition was proved in [7].

Proposition 2. *Let pq be primitive and $p \neq 1$. Let e be an equation with one unknown. Let $Sol(e)$ be the solution set of the equation e. Let $T = Sol(e) \cap (pq)^*p$. Then either $T = (pq)^*p =$ or $T = (pq)^+p$ or $T \subseteq \{p, (pq)^ip\}$ for some $i \geq 1$. Moreover, the set T can be found in linear time.*

In [7], the following proposition was implicitely proved.

Proposition 3. *Let $e : sX... = XtX...$ be an equation with one unknown X and let m be an integer. Then there are at most two solutions $X = x$ such that $m/2 < |tx| \leq m$.*

In [3], the following proposition was proved.

Proposition 4. *$O(\log n)$ candidates for short solutions of an equation with one unknown can be found in linear time independently of the size of the alphabet. The candidates satisfy the constraint in Proposition 3.*

Let $\mathcal{E}_k$ be the family of equations containing at most $2k$ occurrences of the unknown X and possessing finitely many solutions. In this section we prove the following theorem.

Theorem 4. *If $e \in \mathcal{E}_k$, then it possesses at most $8 \log k + O(1)$ solutions. A solution set of an equation in this class can be found in linear time.*

Consider $e \in \mathcal{E}_k$. If it is not irreducible, then it splits into irreducible equations, each of which contains at most $2k$ occurrences of X. Hence, it is enough to consider irreducible equations of the form

$$A_0XA_1 \cdots A_{k-1}X = XB_0XB_1 \cdots XB_{k-1}.$$

Assume additionally that $|B_{k-1}| \geq |A_0|$, the other case being symmetric.

Denote by $B(x)$ the word $xB_0xB_1x \cdots xB_{k-2}x$. Similarly, denote by $A(x)$ the word $xA_1xA_2 \cdots xA_{k-1}x$. We divide all solutions of e into two sets: the solutions x such that $|B(x)| \leq |A_0|$ and the solutions x such that $|B(x)| > |A_0|$.

We omit the detailed proofs. The idea in both of these cases is first, using Proposition 2, to find a length bound β, such that there is only one solution with the length of at least β . Or alternatively β is a (partial) period of some B_l. It could be a period of B_l or it might break somewhere inside B_l. In either case it limits the number of solutions with at least this size to one. Last, another bound $\alpha = \frac{\beta}{k^\gamma}$ is found, and using Propositions 3 and 4 we can deduce, that there are only $2\gamma \log k$ solutions with length in range $[\alpha, \beta]$ and these can be checked in linear time. We prove that there is no solution of length smaller than α. In the second case we actually need another such pair (α', β') of bounds, before we are done.

Our proof is constructive in the sense that we are able to find a solution set of the equation in linear time.

References

1. Baader, F., Snyder, W.: Unification theory. In: Robinson, J.A., Voronkov, A. (eds.) Handbook of Automated Reasoning, vol. 1, pp. 447–533. Elsevier, Amsterdam (2001)
2. Bala, S.: Decision Problems on Regular Expressions. PhD thesis, Universytet Wroclawski, Wroclaw, Poland (2005)
3. Dabrowski, R., Plandowski, W.: On word equations in one variable. In: Diks, K., Rytter, W. (eds.) MFCS 2002. LNCS, vol. 2420, pp. 212–221. Springer, Heidelberg (2002)
4. Gleibman, A.: Knowledge representation via verbal description generalization: alternative programming in sampletalk language. In: Proceedings of Workshop on Inference for Textual Question Answering, Pittsburgh, Pennsylvania. AAAI 2005 - the Twentieth National Conference on Artificial Inteligence, pp. 59–68 (2005), http://www.sampletalk.com
5. Lothaire, M.: Combinatorics on Words. Encyclopedia of Mathematics and its Applications, vol. 17. Addison-Wesley, Reading (1983); Reprinted in the Cambridge Mathematical Library, Cambridge University Press, Cambridge (1997)
6. Lothaire, M.: Algebraic Combinatorics on Words. Encyclopedia of Mathematics and its Applications, vol. 90. Cambridge University Press, Cambridge (2002)
7. Eyono Obono, S., Goralcik, P., Maksimenko, M.N.: Efficient solving of the word equations in one variable. In: Privara, I., Ružička, P., Rovan, B. (eds.) MFCS 1994. LNCS, vol. 841, pp. 336–341. Springer, Heidelberg (1994)
8. Rajasekar, A.: Applications in constraint logic programming with strings. In: Borning, A. (ed.) PPCP 1994. LNCS, vol. 874, pp. 109–122. Springer, Heidelberg (1994)
9. Schaefer, M., Sedgwick, E., Stefankovic, D.: Recognizing string graphs is in NP. Journal of Computer and System Sciences 67(2), 365–380 (2003)

On Equations over Sets of Numbers and Their Limitations[*]

Tommi Lehtinen[1,2] and Alexander Okhotin[1,3]

[1] Department of Mathematics, University of Turku, Finland
{tommi.lehtinen,alexander.okhotin}@utu.fi
[2] Turku Centre for Computer Science
[3] Academy of Finland

Abstract. Systems of equations of the form $X = Y + Z$ and $X = C$, in which the unknowns are sets of natural numbers, "+" denotes element-wise sum of sets $S + T = \{m + n \mid m \in S,\ n \in T\}$, and C is an ultimately periodic constant, have recently been proved to be computationally universal (Jeż, Okhotin, "Equations over sets of natural numbers with addition only", STACS 2009). This paper establishes some limitations of such systems. A class of sets of numbers that cannot be represented by unique, least or greatest solutions of systems of this form is defined, and a particular set in this class is constructed.

1 Introduction

Equations with languages as unknowns, or *language equations*, occur in formal language theory quite often. In particular, finite automata and context-free grammars can be regarded as systems of language equations with states or nonterminal symbols as unknowns. A survey of various results on language equations has recently been given by Kunc [6].

The first connection between language equations and Turing computability was found by Charatonik [1], who established undecidability of testing whether a given system of equations using concatenation and any Boolean operations has no solutions. A decade later Okhotin [10] showed that this problem is actually complete for recursively enumerable (r.e.) sets, while the families of languages representable by unique, least and greatest solutions of such language equations are exactly the recursive, the r.e and the co-r.e. sets, respectively.

Over time it was discovered that progressively simpler types of language equations possess computational universality. First it was shown [11] that systems of equations $\varphi(X) = \psi(X)$ remain computationally complete if the operations in φ, ψ are restricted to union and concatenation. Kunc [5] has constructed an ultimately small computationally universal language equation $LX = XL$, where L is a finite constant, for which the greatest solution is non-recursive. Jeż and Okhotin [3] extended computational completeness of language equations $\varphi(X) = \psi(X)$ (in which the operations are union and concatenation) to the

[*] Supported by the Academy of Finland under grant 118540.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 360–371, 2009.

case of a unary alphabet $\Sigma = \{a\}$. Recently, Jeż and Okhotin [4] proved that these equations remain computationally universal if the only allowed operation is concatenation, while the alphabet remains unary.

As formal languages over a unary alphabet can be regarded as sets of natural numbers, these language equations become *equations over sets of numbers*. The operation of concatenation accordingly represents addition of sets $S + T = \{m + n \mid m \in S,\ n \in T\}$. The properties of sets of natural numbers under this operation is a common subject of studies ranging from additive combinatorics [15] to computational complexity investigated by Stockmeyer and Meyer [14] and McKenzie and Wagner [8].

The recent results of Jeż and Okhotin [3,4] show that equations over sets of numbers with addition only can represent a certain *encoding* $\sigma(S)$ of every recursive (r.e., co-r.e.) set S by a unique (least, greatest, respectively) solution. This encoding contains a number $16n + 13$ if and only if $n \in S$, while the rest of the numbers are defined by a simple pattern. A question remains, whether an arbitrary recursive, r.e. or co-r.e. set of numbers can be specified by equations using only addition without any encoding? This paper settles this question by constructing a computationally easy set of numbers that is not representable.

Only two negative results of this kind are known for any nontrivial families of equations over sets of numbers. Okhotin and Yakimova [13] constructed a set of numbers not representable by systems of equations of the form $X_i = \varphi_i(X_1, \ldots, X_n)$ $(1 \leqslant i \leqslant n)$, where φ_i contain addition and complementation. Okhotin and Rondogiannis [12] established the limitations of one-variable equations $X = \varphi(X)$ using addition, union and intersection, which correspond to conjunctive grammars [9,2] with a single nonterminal symbol.

This paper starts with showing that the approach of Okhotin and Rondogiannis [12] based upon the density of sets is not applicable to unresolved equations of the form $\varphi = \psi$: in Section 3 it is shown that these equations can represent both dense and sparse sets. Then a new class of non-representable sets is proposed in Section 4: these are sets that are both *prime* (in the sense of having no nontrivial representations as sums) and *fragile* (meaning that every nontrivial sum with another set is co-finite), and it is shown that no such set is representable by a unique, least or greatest solution of any system using addition and ultimately periodic constants. Finally, a particular fragile and prime set is constructed in Section 5, which constitutes a non-representable example.

2 Systems of Equations

Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ be the set of natural numbers including 0, and let $S, T \subseteq \mathbb{N}$ be its subsets. The *sum* of these sets is the set $S + T = \{m + n \mid m \in S,\ n \in T\}$.

The subject of this paper are *systems of equations* of the form

$$\begin{cases} \varphi_1(X_1, \ldots, X_n) = \psi_1(X_1, \ldots, X_n) \\ \vdots \\ \varphi_m(X_1, \ldots, X_n) = \psi_m(X_1, \ldots, X_n) \end{cases} \tag{*}$$

where the unknowns X_i are subsets of $\mathbb{N}$ and every expression φ_j, ψ_j is a sum of variables and ultimately periodic constants. A *solution* of such a system is a vector $(S_1, \ldots, S_n)$ with $S_i \subseteq \mathbb{N}$, such that the substitution $X_i = S_i$ turns each equation into an equality.

Define a partial order of componentwise inclusion on such vectors by $(S_1, \ldots, S_n) \sqsubseteq (S'_1, \ldots, S'_n)$ if $S_i \subseteq S'_i$ for all i. For systems with multiple solutions, there is sometimes the *least* or the *greatest* solution with respect to this order.

If equations over sets of numbers may use, besides addition of sets, the operation of set-theoretic union, then the following result on representation of sets by such equations is known:

Theorem 1 (Jeż, Okhotin [3]). *For every recursive (r.e., co-r.e.) set $S \subseteq \mathbb{N}$ there exists a system (*) with φ_j, ψ_j using singleton constants and the operations of union and addition, which has a unique (least, greatest, respectively) solution with $X_1 = S$.*

As a matching upper bound, unique (least, greatest) solutions of equations over sets of numbers with any Boolean operations and addition are known to be recursive (r.e., co-r.e., respectively), so this result precisely characterizes the families of sets representable by solutions of such equations.

For systems of equations over sets of numbers with addition as the only operation, a computational universality result was recently established by Jeż and Okhotin [3]. The idea was to take any recursive (r.e., co-r.e.) set S and consider its *encoding*: another set S' with $16n + 13 \in S'$ if and only if $n \in S$. Then it was proved that any system as in Theorem 1 (that is, with the operations of union and addition) can be simulated by another system using addition only, which manipulates such encodings of sets instead of the sets in their original form. This encoding of sets is defined as follows.

Definition 1. *For each $S \subseteq \mathbb{N}$ and $i \in \{0, 1, \ldots, 15\}$,*

$$\tau_i(S) = \{16n + i \mid n \in S\}.$$

For every set $S \subseteq \mathbb{N}$, its encoding *is the set*

$$\sigma(S) = \{0\} \cup \tau_6(\mathbb{N}) \cup \tau_8(\mathbb{N}) \cup \tau_9(\mathbb{N}) \cup \tau_{12}(\mathbb{N}) \cup \tau_{13}(S).$$

Note that the role of each number in this encoding depends upon its value modulo 16: all numbers equal to 6, 8, 9 or 12 modulo 16 are always in $\sigma(S)$, the numbers equal to 13 modulo 16 represent the encoded set S, and $0 \in \sigma(S)$ is the only extra number. Let us introduce some related terminology. For any set $S' \subseteq \mathbb{N}$, the subset $S' \cap \tau_i(\mathbb{N})$ is called the i^{th} *track of* S'. Track i is said to be *empty (full)* if $S' \cap \tau_i(\mathbb{N}) = \varnothing$ ($\tau_i(\mathbb{N}) \subseteq S'$, respectively).

The first step of the construction used to establish computational completeness of these equations is *checking the encoding.* A special equation is constructed, which has the set of solutions equal to the set of all valid encodings.

Lemma 1 ([4]). *A set $X \subseteq \mathbb{N}$ satisfies an equation*

$$X + \{0,4,11\} = \bigcup_{\substack{i \in \{0,4,6,8,9,\\ 10,12,13\}}} \tau_i(\mathbb{N}) \cup \bigcup_{i \in \{1,3,7\}} \tau_i(\mathbb{N}+1) \cup \{11\}$$

if and only if $X = \sigma(S)$ for some $S \subseteq \mathbb{N}$.

Once it is ensured that all variables represent some sets encoded by σ, both the addition and the union of the encoded sets can be represented by addition of encodings as follows:

Lemma 2 ([4]). *For all sets $X, Y, Z \subseteq \mathbb{N}$,*

$$\sigma(Y) + \sigma(Z) + \{0,1\} = \sigma(X) + \sigma(\{0\}) + \{0,1\} \text{ if and only if } Y + Z = X$$
$$\sigma(Y) + \sigma(Z) + \{0,2\} = \sigma(X) + \sigma(X) + \{0,2\} \text{ if and only if } Y \cup Z = X.$$

The given equations lead to the following general result on the expressive power of equations using only addition:

Theorem 2 (Jeż, Okhotin [4]). *For every recursive (r.e., co-r.e.) set $S \subseteq \mathbb{N}$ there exists a system of equations*

$$\begin{cases} \varphi_1(X_1, \ldots, X_n) = \psi_1(X_1, \ldots, X_n) \\ \vdots \\ \varphi_m(X_1, \ldots, X_n) = \psi_m(X_1, \ldots, X_n) \end{cases}$$

with φ_j, ψ_j using the operation of addition and ultimately periodic constants, which has a unique (least, greatest, respectively) solution with $X_i = \sigma(S_i)$ for some $S_i \subseteq \mathbb{N}$, of which $S_1 = S$.

In plain words, every set that can theoretically be represented by equations over sets of numbers, can be represented modulo a simple encoding. This leaves open the question of whether all such sets can be represented as they are, without using any encoding. A negative answer will be given as the main result of this paper.

In the following, it is convenient to assume that a system has all equations of the form $X = Y + Z$ or $X = C$, where X, Y, Z are variables and $C \subseteq \mathbb{N}$ is an ultimately periodic constant. Any system can be decomposed to this form by adding auxiliary variables representing subexpressions.

3 Representation of Dense and Sparse Sets

In a recent paper on one-variable equations $X = \varphi(X)$ using union, intersection and addition [12], two classes of sets non-representable by such equations were identified. These were *sparse sets* growing faster than exponential, such as $\{n! \mid n \geqslant 1\}$, as well as non-periodic *dense sets* whose complements grow faster than linear. However, in the case of our equations it turns out that sets of both forms can be represented, which will be demonstrated in this section.

Definition 2. *A set $S \subseteq \mathbb{N}$ is* sparse *(*dense*) if the limit* $\lim_{n\to\infty} \frac{|S\cap\{0,\ldots,n-1\}|}{n}$ *exists and equals 0 (1, respectively).*

One can note that a set is dense if its complement is sparse.

The existence of representable non-periodic dense sets easily follows from the calculations of Jeż and Okhotin [4].

Lemma 3. *If S is a recursive (r.e., co-r.e.) set, then $S' = \sigma(S) + \sigma(S) + \{0, 2\}$ is a set representable by a unique (least, greatest, respectively) solution. If S is dense, then S' is also dense.*

Proof. The set $\sigma(S)$ is representable according to Theorem 2, and so is the sum $\sigma(S) + \sigma(S) + \{0, 2\}$. The form of such a sum has already been calculated [4, Tbl. 3]:

$$S' = \tau_{13}(S) \cup \bigcup_{i\in\{0,2,6,8,9,10,11,12,14,15\}} \tau_i(\mathbb{N}) \cup \bigcup_{i\in\{1,3,4,5,7\}} \tau_i(\mathbb{N} + 1).$$

The only numbers greater than 16 in the complement of S' are of the form $16n + 13$, where $n \notin S$. If S is dense, it follows immediately that S' is dense as well. □

For instance, according to this lemma, such a set as $\mathbb{N} \setminus (\{16n^2 + 13 \mid n \geqslant 0\} \cup \{1, 3, 4, 5, 7\})$ is representable by a unique solution of a system.

Representing sparse sets requires a more elaborate construction on top of the constructions of Theorem 2. It will now be proved that all recursive, r.e. and co-r.e. sets of numbers can be represented by equations using a new encoding $\pi : 2^{\mathbb{N}} \to 2^{\mathbb{N}}$, which is much simpler than the encoding σ used by Jeż and Okhotin [4], and which in particular does not use any full tracks. The construction is based upon the known equations for $\sigma(S)$ [4].

Lemma 4. *Assume $0 \in S$. If S is a recursive (r.e., co-r.e.) set, then $\pi(S) = \{0\} \cup \tau_{15}(S)$ is a set representable by a unique (least, greatest, respectively) solution. Furthermore, if S is sparse, then $\pi(S)$ is sparse as well.*

Proof. The question is how to represent $\pi(S)$ for a given S. By Theorem 2, the set $\sigma(S)$ is representable by a certain system of equations with a variable $X = \sigma(S)$. Let Y be a new variable. The goal is to construct two equations: one equation with the set of solutions $Y = \pi(Y_0)$, for all $Y_0 \subseteq \mathbb{N}$ with $0 \in Y_0$, will check that Y represents a valid encoding of some set, while the other equation should ensure that the sets $X = \sigma(S)$ and $Y = \pi(S)$ are encodings of the same set S. The resulting system of equations will have $Y = \pi(S)$ in its unique, least or greatest solution.

The condition of Y being a valid encoding $\pi(S)$ of some set S with $0 \in S$ is represented by the following equation:

$$Y + (\tau_0(\mathbb{N}) \cup \{1\}) = \tau_0(\mathbb{N}) \cup \{1\} \cup \tau_{15}(\mathbb{N})$$

Let Y satisfy the equation. Then $0 \in Y$, since $0 \in \tau_0(\mathbb{N}) \cup \{1\} \cup \tau_{15}(\mathbb{N})$. For all $n \in Y \setminus \{0\}$ it holds that $n, n+1 \in \tau_0(\mathbb{N}) \cup \{1\} \cup \tau_{15}(\mathbb{N})$. This can be the case only if $n \equiv 15 \pmod{16}$. Furthermore $15 \in Y$, because $15 \in \tau_0(\mathbb{N}) \cup \{1\} \cup \tau_{15}(\mathbb{N})$. It follows that $Y = \pi(S)$ for some S with $0 \in S$.

Conversely, let $Y = \pi(S)$ and substitute this value into the equation as follows:

$$\begin{aligned}\pi(S) + (\tau_0(\mathbb{N}) \cup \{1\}) &= (\{0\} \cup \tau_{15}(S)) + (\tau_0(\mathbb{N}) \cup \{1\}) \\ &= \tau_0(\mathbb{N}) \cup \{1\} \cup \underbrace{(\tau_{15}(S) + \tau_0(\mathbb{N}))}_{=\tau_{15}(\mathbb{N})} \cup \underbrace{\tau_0(S+1)}_{\subseteq \tau_0(\mathbb{N})}.\end{aligned}$$

The third component of the last expression equals $\tau_{15}(\mathbb{N})$ because $0 \in S$, while the fourth component is subsumed by $\tau_0(\mathbb{N})$. Accordingly, the expression equals $\tau_0(\mathbb{N}) \cup \{1\} \cup \tau_{15}(\mathbb{N})$, and hence the equation holds for Y.

The correspondence between the encodings σ and π is enforced by the following equation:

$$X + \Big(\bigcup_{\substack{i \in \{0,1,4,5,8,10,\\ 11,12,13,14\}}} \tau_i(\mathbb{N}) \cup \{2\} \Big) = Y + \Big(\{0\} \cup \bigcup_{i \in \{1,\ldots,14\} \setminus \{3\}} \tau_i(\mathbb{N}) \Big)$$

It is claimed that if X is a σ-encoding of some set and Y is a π-encoding of some set, then they must be encodings of the same set. Formally, $X = \sigma(\widehat{X})$, $Y = \pi(\widehat{Y})$ with $0 \in \widehat{Y}$ is a solution of this equation if and only if $\widehat{X} = \widehat{Y}$.

Consider the left-hand side of the equation. Since $0 \in X$, the tracks 0, 1, 4, 5, 8, 10, 11, 12, 13 and 14 in the sum are full. Track 2 is full for the reason that X contains full track 6 and the constant set in the left-hand side contains full track 12, which sum up to $\tau_6(\mathbb{N}) + \tau_{12}(\mathbb{N}) = \tau_2(\mathbb{N}+1)$, while the number 2 also belongs to the sum. Similarly, track 3 in the sum contains $\tau_8(\mathbb{N}) + \tau_{11}(\mathbb{N}) = \tau_3(\mathbb{N}+1)$, though the number 3 is not included in the sum, Tracks 6 and 9 are full in X, so they are full in the sum also. Track 7 in the sum contains $\tau_6(\mathbb{N}) + \tau_1(\mathbb{N}) = \tau_7(\mathbb{N})$. All tracks in the sum except 15 have been considered. The track 15 contains the set $\widehat{X}$, since $\tau_{13}(\widehat{X}) + \{2\} = \tau_{15}(\widehat{X})$. Other tracks in X do not contribute anything to track 15, since $15 - \{0, 6, 8, 9, 12\} = \{3, 6, 7, 9, 15\}$ and all these tracks are empty in the other summand. These considerations yield:

$$X + \Big(\bigcup_{\substack{i \in \{0,1,4,5,8,10,\\ 11,12,13,14\}}} \tau_i(\mathbb{N}) \cup \{2\} \Big) = \Big(\bigcup_{i \in \{0,1,\ldots,14\} \setminus \{3\}} \tau_i(\mathbb{N}) \Big) \cup \tau_3(\mathbb{N}+1) \cup \tau_{15}(\widehat{X}).$$

Then consider the right-hand side. Since $0 \in Y$, the tracks in $\{1, \ldots, 14\} \setminus \{3\}$ are full in the sum. Track 0 is full, since $\{15\} + \tau_1(\mathbb{N}) = \tau_0(\mathbb{N}+1)$ and the number 0 is included. Track 3 is given by $\{15\} + \tau_4(\mathbb{N}) = \tau_3(\mathbb{N}+1)$. Finally, $\tau_{15}(\widehat{Y}) + \{0\} = \tau_{15}(\widehat{Y})$ specifies the content of track 15. Combining these gives:

$$Y + \Big(\{0\} \cup \bigcup_{i \in \{1,\ldots,14\} \setminus \{3\}} \tau_i(\mathbb{N}) \Big) = \Big(\bigcup_{i \in \{0,1,\ldots,14\} \setminus \{3\}} \tau_i(\mathbb{N}) \Big) \cup \tau_3(\mathbb{N}+1) \cup \tau_{15}(\widehat{Y}).$$

Since the sets in both sides are equal, the sets $\widehat{X}$ and $\widehat{Y}$ must be the same, and so the claim about the correspondence of these two encodings σ and π has been proved. Since $\sigma(S)$ is representable for $S \subseteq \mathbb{N}$ with $0 \in S$, so is $\pi(S)$.

To complete the proof of the lemma, let S be sparse. Clearly, $\pi(S) = \{0\} \cup \{16n + 15 \mid n \in S\}$ is sparse as well. □

For example, though this lemma does not prove that the exact set $\{n! \mid n \geqslant 1\}$ is representable, it asserts representability of a very similar sparse set $\pi(\{n! \mid n \geqslant 1\} \cup \{0\}) = \{0, 15\} \cup \{16n! + 15 \mid n \geqslant 1\}$.

More generally, one can note that for every recursive, r.e. or co-r.e. set S, the set $\pi(S)$ grows asymptotically as fast as S, so, like in the case of equations with union, intersection and addition with multiple variables [2, Thm. 6], sets with any theoretically possible growth rate can be represented.

4 Prime and Fragile Sets

In this section a class of non-representable sets is introduced, which is defined by the conditions of being *prime* and of being *fragile*. Prime sets of numbers are defined in a usual sense, with respect to the monoid of sets with addition as product.

Definition 3. *A set $S \subseteq \mathbb{N}$ is prime if $S = S_1 + S_2$ implies $S_1 = \{0\}$ or $S_2 = \{0\}$.*

Fragility of a set means that the sum of this set with any set containing at least two elements is co-finite.

Definition 4. *A set $S \subseteq \mathbb{N}$ is fragile if $S + \{n_1, n_2\}$ is co-finite for all $n_1, n_2 \in \mathbb{N}$ with $n_1 \neq n_2$.*

In other words, for every $k \geqslant 1$ there are only finitely many $n \in \mathbb{N}$ such that $n, n+k \notin S$. Assuming that S is not co-finite, this is equivalent to saying that the difference between two consecutive numbers not in S tends to infinity. Clearly, every fragile set is dense, but not vice versa: the set $\mathbb{N} \setminus \{2^n, 2^n + 1 \mid n \geqslant 0\}$ is dense but not fragile.

There are no particular problems with representing prime sets or fragile sets by these equations. A fragile set can be represented as in Lemma 3: note that if the set S is fragile, then $S' = \sigma(S) + \sigma(S) + \{0, 2\}$ is fragile as well. Representable prime sets are also known: in fact, all sets represented in Theorem 2 and in Lemma 4 are prime:

Lemma 5. *For every set $S \subseteq \mathbb{N}$, the sets $\sigma(S)$ and $\pi(S)$ (with $0 \in S$ in the latter case) are prime.*

Proof. Let $0 \in S$ and suppose $\pi(S) = X + Y$ for some $X, Y \subseteq \mathbb{N}$. Then $0 \in X, Y$. Since $15 \in \pi(S)$, it has to be in one of X and Y, say X. If Y contains any number $n > 0$, then $n \in \pi(S)$, and thus $n \equiv 15 \pmod{16}$. But in this case $15 + n \equiv 14 \pmod{16}$, and cannot be in $\pi(S)$. So $Y = \{0\}$ and $\pi(S)$ is prime.

Proof for the primality of $\sigma(S)$ is also straightforward, and is omitted. □

Note that the representable prime set $\sigma(S)$ is not fragile, since it contains empty tracks, while the representable fragile set $\sigma(S) + \sigma(S) + \{0, 2\}$ is not prime by construction. It is indeed the case that if a set is both prime and fragile, then it is not representable.

Theorem 3. *Let S be prime and fragile. Then it is not representable by a least or a greatest solution of any system.*

The idea of the proof is that for every solution with a prime and fragile component $X = S$ there exists a smaller solution, in which some number is excluded from X, as well as a greater solution with one extra number in X. Exclusion of a number from a prime and fragile component of a solution is done in the following lemma.

Lemma 6. *If a system has a solution with a component $X = S$, where S is a prime and fragile set, then the system has a smaller solution with $X = S \setminus \{n_0\}$, for some $n_0 \in S$.*

Proof. Assume without loss of generality that all equations in the system are of the form $X = Y + Z$ or $X = C$. Consider a solution, in which one of the components is $X_0 = S$, and possibly some other components have values of the form $S_i = S + \{i\}$: let them be denoted $X_1 = S_1, \ldots, X_{\widehat{m}} = S_{\widehat{m}}$ for some $\widehat{m} \geqslant 0$, and let $Y_1 = T_1, \ldots, Y_{\widehat{n}} = T_{\widehat{n}}$ be the rest of the variables, whose values are not of the form $S + \{i\}$ for any i. It can be assumed that all components of the solution are nonempty.

The smaller solution is defined as follows: Let k be the maximum of the differences of the two smallest numbers in any V, with $|V| \geqslant 2$, that appears in the solution. Since S is fragile, there is a number ℓ, such that every pair of missing numbers $m, n \notin S$ with $n > m \geqslant \ell$ satisfies $n - m > k$. Let $\ell' \geqslant \ell$ be such that $\{\ell', \ell' + 1, \ldots, \ell' + 2k\} \subseteq S$. Such ℓ' exists by the fragility of S.

The new solution is $X_i = S'_i = (S \setminus \{\ell' + k\}) + \{i\}$ for $0 \leqslant i \leqslant \widehat{m}$ and $Y_j = T_j$ for $1 \leqslant j \leqslant \widehat{n}$. As compared to S_i, one more number is missing from S'_i, but it is different by at least $k + 1$ from any other missing number. Thus the above property is inherited by S'_i: every pair $m, n \notin S'_i$ with $n > m \geqslant \ell + i$ satisfies $n - m > k$.

It has to be shown that the new solution satisfies the equations. Consider every equation in the system. If none of the variables X_i are used in the equation, then the new solution clearly satisfies it, since the variables Y_j did not change their values. So consider each equation containing an instance of some X_i; only three cases are possible:

Claim 6.1. *The equations that contain some X_i are of the form*

1. *$X_i = X_{i_1} + Y_j$, with $T_j = \{i_2\}$ and $i = i_1 + i_2$,*
2. *$Y_j = X_i + Y_{j_1}$, with $|T_{j_1}| \geqslant 2$ or*
3. *$Y_j = X_i + X_{i_1}$.*

Proof. Let X_i be on the left-hand side of the equation. The right-hand side of this equation cannot be a constant, since S is not ultimately periodic. So the equation is of the form $X_i = U + V$.

Then $S + \{i\}$ is factorized as $R_1 + R_2$ for some sets $R_1, R_2 \subseteq \mathbb{N}$. Let i_1 and i_2 be the smallest elements of R_1 and R_2, respectively. Then $i_1 + i_2$ is the smallest element of $S + \{i\}$, which must be i, as $0 \in S$ by the primality of S. Therefore, S is factorized as $(R_1 - i_1) + (R_2 - i_2)$, and since S is prime, $R_2 - i_2 = \{0\}$, that is, $R_2 = \{i_2\}$ and $R_1 = S + \{i_1\} = S_{i_1}$. Then the equation is of the form $X_i = X_{i_1} + Y_j$, where $T_j = \{i_2\}$.

If the equation is $Y_j = X_i + V$, and if $V = Y_{j_1}$, then it must be the case that $|T_{j_1}| \geqslant 2$: otherwise, $T_{j_1} = \{i_1\}$ for some $i_1 \in \mathbb{N}$ and $T_j = S_i + \{i_1\} = S_{i+i_1}$, which contradicts the assumption that $T_j \notin \{S_i \mid 0 \leqslant i \leqslant \widehat{m}\}$.

The only case left is $Y_j = X_i + X_{i_1}$. □

So there are three cases to consider:

1. Let X_i be on the left-hand side of the equation, so that it is of the form $X_i = X_{i_1} + Y_j$. Then the solution is $S_i = S_{i_1} + T_j$ with $T_j = \{i_2\}$ and $i = i_1 + i_2$. The equality $S'_i = S'_{i_1} + \{i_2\}$ holds by definition of the sets S'_i.
2. Let the equation be of the form $Y_j = X_i + Y_{j_1}$, with $|T_{j_1}| \geqslant 2$. The goal is to show that $S'_i + T_{j_1} = S_i + T_{j_1}$. Clearly, $S'_i + T_{j_1} \subseteq S_i + T_{j_1}$. Consider any number $m + n$ with $m \in S_i$ and $n \in T_{j_1}$. If $m \neq i + \ell' + k$, then $m \in S'_i$ by the definition of S'_i and thus $m + n \in S'_i + T_{j_1}$. Let $m = i + \ell' + k$ and let $n_1 < n_2$ be the two smallest numbers in T_{j_1}. Then $n_2 - n_1 \leqslant k$, by the definition of k.

   Consider two numbers $m + n - n_1$ and $m + n - n_2$. The former is clearly greater than $\ell + i$, since $n_1 \leqslant n$. For the latter number, note that $n - n_2 \geqslant n - n_1 - k$ and thus $m + n - n_2 \geqslant (i + \ell' + k) + (n - n_1 - k) = i + \ell' + n - n_1 \geqslant \ell + i$. Since both numbers are greater or equal to $\ell + i$ and their difference is at most k, it could not be the case that both of them are missing from S'_i. Therefore, either $(m + n - n_1) + n_1 \in S'_i + T_{j_1}$ or $(m + n - n_2) + n_2 \in S'_i + T_j$, which proves that $m + n$ is in $S'_i + T_{j_1}$.
3. Finally consider the case $Y_j = X_i + X_{i_1}$. The solution is $T_j = S_i + S_{i_1}$. It is to be shown that $T_j = S'_i + S'_{i_1}$. Obviously $S'_i + S'_{i_1} \subseteq S_i + S_{i_1}$. To prove the converse, assume that $m \in S_i$ and $n \in S_{i_1}$. If $m \in S'_i$, then the argument used in the previous case applies, with S'_{i_1} instead of T_{j_1}. The case of $n \in S'_{i_1}$ is handled symmetrically.

   Suppose $m \notin S'_i$ and $n \notin S'_{i_1}$. Then $m = i + \ell' + k$ and $n = i_1 + \ell' + k$. Now $m - 1 \in S'_i$ and $n + 1 \in S'_{i_1}$, so $m + n = (m - 1) + (n + 1) \in S'_i + S'_{i_1}$. Thus $T_j = S'_i + S'_{i_1}$.

It follows that the system of equations is satisfied and the Lemma is proved. □

The next lemma is in some sense symmetric to Lemma 6, as it allows augmenting a prime and fragile component of any solution with an extra number. This, in particular, shows that a greatest solution cannot contain components that are both prime and fragile.

Lemma 7. *If a system has a solution with a fragile and prime component $X = S$, then the system has a greater solution with $X = S \cup \{n_0\}$, for some $n_0 \notin S$.*

Proof (sketch). As in the proof of Lemma 6, let the system of equations have a solution $X_0 = S$, $X_1 = S_1$, …, $X_{\widehat{m}} = S_{\widehat{m}}$, $Y_1 = T_1$, …, $Y_{\widehat{n}} = T_{\widehat{n}}$, where $S_i = S + \{i\}$ and $T_j \notin \{S_i \mid 0 \leqslant i \leqslant \widehat{m}\}$.

For every equation of the form $Y_j = X_i + V$, where V is either a variable X_j or a variable Y_j with $|T_j| \geqslant 2$, the sum $S_i + V$ is co-finite as S_i is fragile. Let k be the least number with $\ell \in S_i + V$ for all $\ell \geqslant k$ and let k_0 be the maximum of all k's for all such equations. Let $n_0 \geqslant k_0$ and $n_0 \notin S$.

In the new solution this number n_0 is added to S: $X_i = S'_i = (S \cup \{n_0\}) + \{i\} = S_i \cup \{n_0 + i\}$ and $Y_j = T_j$ for all applicable i and j. To see that this assignment is a solution, only equations where some X_i occurs have to be considered, since the rest stay as they were. As in the proof of the previous lemma, see Claim 6.1, there are three cases to consider:

1. For an equation $X_i = X_{i_1} + Y_j$ the solution is $S_i = S_{i_1} + T_j$ with $T_j = \{i_2\}$ and $i = i_1 + i_2$. The equation is satisfied for the new solution as well: $(S \cup \{n_0\}) + \{i\} = (S \cup \{n_0\}) + \{i_1 + i_2\} = ((S \cup \{n_0\}) + \{i_1\}) + \{i_2\}$.
2. In the case $Y_j = X_i + Y_{j_1}$ with $|T_{j_1}| \geqslant 2$, note that, since $n_0 \geqslant k_0$, every number greater or equal to n_0 is in T_j by the choice of k_0. Then $S'_i + T_{j_1}$ equals
$$(S_i \cup \{n_0 + i\}) + T_{j_1} = (S_i + T_{j_1}) \cup (\{n_0 + i\} + T_{j_1}) = T_j \cup \underbrace{(\{n_0 + i\} + T_{j_1})}_{\subseteq T_j} = T_j.$$
3. In the final case the equation is $Y_j = X_i + X_{i_1}$. Again $\ell \in T_j$ for every $\ell \geqslant n_0$, and $S'_i + S'_{i_1}$ can be transformed as
$$\begin{aligned} &(S_i \cup \{n + i\}) + (S_{i_1} \cup \{n_0 + i_1\}) = \\ = &\underbrace{(S_i + S_{i_1})}_{=T_j} \cup \underbrace{(S_i + \{n_0 + i_1\})}_{\subseteq T_j} \cup \underbrace{(S_{i_1} + \{n_0 + i_1\})}_{\subseteq T_j} \cup \underbrace{(\{n_0 + i\} + \{n_0 + i_1\})}_{\subseteq T_j}, \end{aligned}$$
which equals T_j. □

It has thus been shown that any set which is *both* prime and fragile is non-representable by systems of equations. It remains to show that any such sets exist. A set having both properties is constructed in the following section.

5 A Fragile Prime Set

In this section a prime and fragile subset of natural numbers is constructed. First some concepts that are used in the construction are defined.

For a set $X \subseteq \mathbb{N}$ the infinite word $w(X) = x_0 x_1 x_2 \cdots \in \{0, 1\}^\omega$, where
$$x_k = \begin{cases} 1, \text{if } k \in X \\ 0, \text{if } k \notin X, \end{cases}$$
is called the *characteristic word* of X.

The relation $\preccurlyeq$ between infinite words over $\{0,1\}$ defined by

$$x_0x_1x_2\ldots \preccurlyeq y_0y_1y_2\ldots, \text{ iff } x_k \leqslant y_k \text{ for all } k \in \mathbb{N}$$

is a partial order. It corresponds to set inclusion over subsets of $\mathbb{N}$ in the sense that $w(X) \preccurlyeq w(Y)$ if and only if $X \subseteq Y$, for $X, Y \subseteq \mathbb{N}$. A similar relation is defined for finite words of matching length as $x_1 \ldots x_n \preccurlyeq y_1 \ldots y_n$ if $x_i \leqslant y_i$ for each i.

A finite word w is a *$\preccurlyeq$-factor* of a word $v \in \{0,1\}^* \cup \{0,1\}^\omega$, if there are words x, w' and y, such that $v = xw'y$, $|w'| = |w|$ and $w \preccurlyeq w'$.

Let $u = 100011110000$ and $v_k = (1^k0)^{2^k}$ for all $k \geqslant 2$, and consider a set S defined by the characteristic word

$$w(S) = s = s_0s_1s_2\cdots = uv_2v_3v_4\cdots = 100011110000\prod_{k=2}^{\infty}(1^k0)^{2^k}.$$

It will now be proved that this has the desired properties.

Lemma 8. *The set S is fragile and prime.*

Proof. Fragility is obvious, since starting from v_k the distance between any two zeroes is at least $k+1$.

To prove the primality of S, suppose that $S = X+Y$, for some $X, Y \subseteq \mathbb{N}$. Let $w(X) = x = x_0x_1x_2\ldots$ and $w(Y) = y = y_0y_1y_2\ldots$ be the characteristic words of X and Y.

Since $S = X+Y$, it holds that $X + \{k\} \subseteq S$ for all $k \in Y$. This is equivalent to $0^kx \preccurlyeq s$. The characteristic word for S has 10001 as a prefix, so 0 and 4 are the two smallest numbers in S. One of the sets, X or Y, must contain them both. Let it be X, so that x begins with 10001.

If the factorization $S = X+Y$ is nontrivial, then there is the smallest nonzero number $m \in Y$. Then $m, m+4 \in S$, and it is easy to see that $m \geqslant 12$: indeed, by the form of u, there is no pair $s_i = s_{i+4} = 1$ for $1 \leqslant i \leqslant 11$. Consequently, u is a prefix of x.

Now $x_i = s_i$ for $i < m$ and $0^mx = 0^mu\cdots \preccurlyeq s$. Since $u = 100011110000$, we have $s_{m+4} = s_{m+5} = s_{m+6} = s_{m+7} = 1$. Then

$$m \geqslant 20 = |uv_2v_3| - 4, \tag{1}$$

because v_4 is the first of the words v_k to have 1111 as a $\preccurlyeq$-factor. Let the first m symbols of x be

$$uv_2\cdots v_nv,$$

where $n \geqslant 2$ and $v_{n+1} = vv'$. It follows that $uv_2\cdots v_n$ is a $\preccurlyeq$-factor of $v'v_{n+2}$, since $|uv_2\cdots v_n| = n\cdot 2^{n+1} + 8 < (n+3)\cdot 2^{n+2} = |v_{n+2}|$. In particular, there is a factor w of $v'v_{n+2}$ with $|w| = |v_n|$ and $v_n \preccurlyeq w$. Since the distance between consecutive occurences of zero in $v'v_{n+2}$ is at most $n+3$, and $|w| = (n+1)\cdot 2^n > 2(n+2)+1$, it follows that w contains at least two occurrences of zero. If w has a zero in some position, then v_n has to have a zero in the same position. Since the distance between consecutive zeroes is $n+1$ in v_n and $n+2$ or $n+3$ in w, a contradiction is obtained. It follows that $Y = \{0\}$ and S is prime. □

A fragile and prime set was constructed, witnessing that the class considered in the previous section is not empty. Therefore, by Theorem 3, this set S is not representable by equations with ultimately periodic constants and the operation of addition. Since S is obviously recursive, and in fact computationally very easy, this shows that these equations are less powerful than the equations equipped with addition and union.

References

1. Charatonik, W.: Set constraints in some equational theories. Information and Computation 142(1), 40–75 (1998)
2. Jeż, A., Okhotin, A.: Conjunctive grammars over a unary alphabet: undecidability and unbounded growth. Theory of Computing Systems (to appear)
3. Jeż, A., Okhotin, A.: On the computational completeness of equations over sets of natural numbers. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 63–74. Springer, Heidelberg (2008)
4. Jeż, A., Okhotin, A.: Equations over sets of natural numbers with addition only. In: STACS 2009, Freiburg, Germany, February 26-28, 2009, pp. 577–588 (2009)
5. Kunc, M.: The power of commuting with finite sets of words. Theory of Computing Systems 40(4), 521–551 (2007)
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. Leiss, E.L.: Unrestricted complementation in language equations over a one-letter alphabet. Theoretical Computer Science 132, 71–93 (1994)
8. McKenzie, P., Wagner, K.: The complexity of membership problems for circuits over sets of natural numbers. Computational Complexity 16(3), 211–244 (2007)
9. Okhotin, A.: Conjunctive grammars. Journal of Automata, Languages and Combinatorics 6(4), 519–535 (2001)
10. Okhotin, A.: Decision problems for language equations with Boolean operations. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719. Springer, Heidelberg (2003)
11. Okhotin, A.: Unresolved systems of language equations: expressive power and decision problems. Theoretical Computer Science 349(3), 283–308 (2005)
12. 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, 2008, vol. 273, pp. 215–227. IFIP (2008)
13. Okhotin, A., Yakimova, O.: On language equations with complementation. In: H. Ibarra, O., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 420–432. Springer, Heidelberg (2006)
14. Stockmeyer, L.J., Meyer, A.R.: Word problems requiring exponential time. In: STOC 1973, pp. 1–9 (1973)
15. Tao, T., Vu, V.: Additive Combinatorics. Cambridge University Press, Cambridge (2006)

Some Remarks on Superposition Based on Watson-Crick-Like Complementarity*

Florin Manea[1], Victor Mitrana[1,2], and Jose M. Sempere[2]

[1] Faculty of Mathematics and Computer Science,
University of Bucharest,
Str. Academiei 14, 70109, Bucharest, Romania
flmanea@gmail.com, mitrana@fmi.unibuc.ro

[2] Department of Information Systems and Computation
Technical University of Valencia,
Camino de Vera s/n. 46022 Valencia, Spain
jsempere@dsic.upv.es

Abstract. In this note we solve a problem left open in [2]: namely, we show that the iterated superposition of a regular language is regular. The proof of this result is based on two facts: (i) the iterated superposition of a language equals the restricted iterated superposition of the same language, (ii) the sequence formed by iteratively applying the restricted superposition can be precisely defined. We then define the restricted superposition distance between a word and a language and prove that this distance can be computed in time $\mathcal{O}(n^2 f(n))$, where the language is accepted in time $\mathcal{O}(f(n))$ in the RAM model. Finally, we briefly discuss the necessity of the n^2 factor for the classes of regular and context-free languages.

1 Introduction

A DNA molecule consists of a double strand, each DNA single strand being composed of nucleotides which differ from each other by their bases: A (adenine), G (guanine), C (cytosine), and T (thymine). The two strands which form the DNA molecule are kept together by the hydrogen bond between the bases: A always bonds with T, while C bonds with G. This paradigm of *Watson-Crick complementarity* is one of the main concepts used in defining the formal operation of superposition investigated in [2].

Two other biological principles used as sources of inspiration in that paper are those of *annealing* and *lengthening DNA by polymerase*, respectively. The first principle refers to fusing two single stranded molecules by complementary base pairing while the second one refers to adding nucleotides to one strand (in a more general setting to both strands) of a double-stranded DNA molecule. The former operation requires a heated solution containing the two strands which

* Work partially supported by the PN II projects 11052(GlobalComp) and 11056(CellSim).

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 372–383, 2009.

is cooled down slowly. The latter one requires two single strands such that one (usually called *primer*) is bonded to a part of the other (usually called *template*) by Watson-Crick complementarity and a *polymerization buffer* with many copies of the four nucleotides that polymerase will concatenate to the primer by complementing the template.

We now informally explain the superposition operation and how it can be related to the aforementioned biological concepts. Let us consider the following hypothetical biological situation: two single stranded DNA molecules x and y are given such that a suffix of x is Watson-Crick complementary to a prefix of y or a prefix of x is Watson-Crick complementary to a suffix of y, or x is Watson-Crick complementary to a subword of y. Then x and y get annealed in a DNA molecule with a double stranded part by complementary base pairing and then a complete double stranded molecule is formed by DNA polymerase. The mathematical expression of this hypothetical situation defines the superposition operation. Assume that we have an alphabet and a complementary relation on its letters. For two words x and y over this alphabet, if a suffix of x is complementary to a prefix of y or a prefix of x is complementary to a suffix of y, or x is complementary to a subword of y, then x and y bond together by complementary letter pairing and then a complete double stranded word is formed by the prolongation of x and y. Now both words, namely the upper one, formed by the prolongation of x, and lower one, formed by the prolongation of y, are considered to be the result of the superposition applied to x and y. Of course, all these phenomena are considered here in an idealized way. For instance, we allow polymerase to extend the shorter strand in either end (3' or 5' in DNA biochemistry) as well as in both, despite that in biology almost all polymerase extend in the direction from 5' to 3'.

As shown in [2], this operation resembles some other operations on words: *sticking* investigated in [5,3,9] (particular polyominoes with sticky ends are combined provided that the sticky ends are Watson-Crick complementary), *PA-matching* considered in [7] which is related to both the *splicing* and the *annealing* operations, and the *superposition* operation introduced in [1] (two words which may contain *transparent* positions are aligned one over the other and the resulting word is obtained by reading the visible positions as well as aligned transparent positions). The reader interested in related bio-inspired operations is referred to [8] and [6].

In this work we propose a slightly different variant of the operation considered in [2]. It turns out that both operations coincide when they are applied to a language. This variant allows us to solve a problem left open in [2]: namely, we show that the iterated superposition of a regular language is regular. The proof of this result is based on two facts: (i) a sort of normal form which implies that the iterated superposition of a language equals the restricted iterated superposition of the same language, (ii) the sequence formed by iteratively applying the restricted superposition can be precisely defined. We then define the restricted superposition distance between a word and a language and prove that this distance can be computed in time $\mathcal{O}(n^2 f(n))$, where the language is accepted in time $\mathcal{O}(f(n))$ in the RAM model. Finally, we briefly discuss the necessity of the n^2 factor for the classes of regular and context-free languages.

2 Preliminaries

We assume the reader to be familiar with the fundamental concepts of formal language theory and automata theory, particularly the notions of grammar and finite automaton [10].

An alphabet is always a finite set of letters. For a finite set A let card(A) denote the cardinality of A. The set of all words over an alphabet V is denoted by V^*. The empty word is written ε; moreover, $V^+ = V^* \setminus \{\varepsilon\}$ or equivalently $V^+ = VV^*$. Given a word w over an alphabet V, let $|w|$ denote the length of w. If $w = xyz$ for some $x, y, z \in V^*$, then x, y, z are called prefix, subword, suffix, respectively, of w.

Let Ω be a "superalphabet", that is an infinite set such that any alphabet considered in this paper is a subset of Ω. In other words, Ω is the *universe* of the languages in this paper, i.e., all words and languages are over alphabets that are subsets of Ω. An *involution* over a set S is a bijective mapping $\sigma : S \longrightarrow S$ such that $\sigma = \sigma^{-1}$. Any involution σ on Ω such that $\sigma(a) \neq a$ for all $a \in \Omega$ is said to be here a *Watson-Crick involution.* Despite that this is nothing more than a fixed point-free involution, we prefer this terminology since the superposition defined later is inspired by the DNA lengthening by polymerase, where the Watson-Crick complementarity plays an important role. Let $\bar{\cdot}$ be a Watson-Crick involution fixed for the rest of the paper. The Watson-Crick involution is extended to a morphism from Ω^* to Ω^* in the usual way. We say that the letters a and $\overline{a}$ are complementary to each other. For an alphabet V, we set $\overline{V} = \{\overline{a} \mid a \in V\}$. Note that V and $\overline{V}$ can intersect and they can be, but need not be, equal. Remember that the DNA alphabet consists of four letters, $V_{DNA} = \{A, C, G, T\}$, which are abbreviations for the four nucleotides and we may set $\overline{A} = T$, $\overline{C} = G$.

2.1 Non-iterated Superposition

Given two words $x, y \in V^+$ we define the following operations:

$$\begin{aligned}
x \blacktriangleright y &= \{uw\overline{v}, \overline{uw}v \mid x = uw, y = \overline{w}v \text{ for some } u, v \in V^*, w \in V^+\}\\
x \blacktriangleleft y &= \{\overline{u}wv, u\overline{wv} \mid x = wv, y = u\overline{w} \text{ for some } u, v \in V^*, w \in V^+\}\\
x \blacktriangle y &= \{\overline{u}x\overline{v}, y \mid y = u\overline{x}v \text{ for some } u, v \in V^*\}\\
x \blacktriangledown y &= \{x, \overline{u}y\overline{v} \mid x = u\overline{y}v \text{ for some } u, v \in V^*\}.
\end{aligned}$$

Clearly, $x \blacktriangleright y = y \blacktriangleleft x$ and $x \blacktriangle y = y \blacktriangledown x$ for any pair of words x, y. Despite this redundancy we prefer to work with these definitions because they allow a simplification of the arguments we are to discuss.

We now define the *superposition* operation applied to the pair of words $x, y \in V^+$ as above, denoted by $\blacklozenge$, as follows:

$$x \blacklozenge y = (x \blacktriangleright y) \cup (x \blacktriangleleft y) \cup (x \blacktriangle y) \cup (x \blacktriangledown y).$$

The result of this operation applied to the two words x and y as above which might be viewed as two single stranded molecules is the pair of words formed by z and its complement $\overline{z}$ which form a double stranded molecule $\frac{z}{\overline{z}}$. By this operation, based on the Watson-Crick complementarity, we can generate a finite set of words, starting from a pair of words, in which the contribution of a word to the result need not be one subword, as happens in classical bio-operations of DNA computing [8]. Note the difference between this operation and the superposition operation defined in [2], where only the upper word is considered to be the result of superposition.

We stress from the very beginning the mathematical character of the definition proposed in [2]: nature cannot distinguish which is the upper or the lower strand in the process of constructing a double stranded molecule from two single strands. This drawback is avoided by the definition proposed here. Further, our model reflects polymerase reactions in both 5'⟶ 3' and 3'⟶ 5' directions. Due to the greater stability of 3' when attaching new nucleotides, DNA polymerase can act continuously only in the 5'⟶ 3' direction. However, polymerase can also act in the opposite direction, but in short “spurts” (Okazaki fragments).

We extend this operation to languages by

$$L_1 \blacklozenge L_2 = \bigcup_{x \in L_1, y \in L_2} x \blacklozenge y.$$

We write $\blacklozenge(L)$ instead of $L \blacklozenge L$. It is plain that the superposition operation proposed in [2] and that proposed here coincide when they are applied to a language.

Note that superposition is not associative. Indeed, take the alphabet $\{a, b, \overline{a}, \overline{b}\}$ and the words $x = ab$, $y = \overline{b}a$, $z = aa$. It is easy to see that $(x \blacklozenge y) \blacklozenge z = \{ab\overline{aa}, \overline{ab}aa, \overline{aab}a, aab\overline{a}\}$ while $x \blacklozenge (y \blacklozenge z) = \emptyset$.

2.2 Iterated Superposition

Given a language L we define the language obtained from L by unrestrictedly iterated application of superposition. This language, called the *unrestricted superposition closure of* L, is denoted by $\blacklozenge_u^*(L)$ and defined by

$$\begin{aligned} \blacklozenge_u^0(L) &= L, \\ \blacklozenge_u^{i+1}(L) &= \blacklozenge_u^i(L) \cup \blacklozenge(\blacklozenge_u^i(L)), i \geq 0, \\ \blacklozenge_u^*(L) &= \bigcup_{i \geq 0} \blacklozenge_u^i(L). \end{aligned}$$

Clearly, $\blacklozenge_u^*(L)$ is the smallest language containing L and closed under superposition. More precisely, it is the smallest language K such that $L \subseteq K$ and $\blacklozenge(K) \subseteq K$. In words, one starts with the words in L and applies superposition iteratively to any pair of words previously produced. Note the lack of any restriction in choosing the pair of words. All the obtained words are collected in the set $\blacklozenge_u^*(L)$.

We say that a family $\mathcal{F}$ of languages is closed under unrestrictedly iterated superposition if $\blacklozenge_u^*(L)$ is in $\mathcal{F}$ for any language $L \in \mathcal{F}$.

We now recall from [2] another superposition closure of a language which may be viewed as a "normal form" of iterated superposition. The *restricted superposition closure of* L denoted by $\blacklozenge_r^*(L)$ is defined in the following way:

$$\begin{aligned}
\blacklozenge_r^0(L) &= L,\\
\blacklozenge_r^{i+1}(L) &= ((\blacklozenge_r^i(L))\blacklozenge L) \cup (\blacklozenge_r^i(L))\\
\blacklozenge_r^*(L) &= \bigcup_{i\geq 0} \blacklozenge_r^i(L).
\end{aligned}$$

Note the main difference between the unrestricted and restricted way of iterating superpositions. In the latter case, superposition takes place between a word produced so far and an initial word only.

3 Iterated Superposition Preserves Regularity

Note that $\blacklozenge_r^*(L) \subseteq \blacklozenge_u^*(L)$ for any language L. Surprisingly enough (remember that $\blacklozenge$ is not associative), we have an equality between the two superposition closures of any language.

Theorem 1. [Normal Form Theorem][2] *$\blacklozenge_r^*(L) = \blacklozenge_u^*(L)$ for any language L.*

This theorem allows us to use the notation $\blacklozenge^*$ when the way of iterating the superposition does not matter. The problem of closure under iterated superposition of the class of regular languages was left open in [2]. We propose here an affirmative answer to this question. To this aim, we need some preliminary results. The redundancy introduced in the definition of the superposition operation turns out to be useful now. For two languages L_1, L_2 we define

$$\begin{aligned}
(i)\quad & \blacktriangleleft (L_1, L_2) = \bigcup_{x\in L_1, y\in L_2} x \blacktriangleleft y,\\
(ii)\quad & \blacktriangleleft^0 (L_1, L_2) = L_1,\\
(iii)\quad & \blacktriangleleft^{i+1} (L_1, L_2) = \blacktriangleleft ((\blacktriangleleft^i (L_1, L_2)), L_2), i \geq 0,\\
(iv)\quad & \blacktriangleleft^* (L_1, L_2) = \bigcup_{i\geq 0} \blacktriangleleft^i (L_1, L_2).
\end{aligned}$$

The language $\blacktriangleright^* (L_1, L_2)$ is defined analogously.

Lemma 1. *For every language L and any integer $k \geq 1$, the following relations hold:*

$$(\blacklozenge_r^k(L)\backslash L) = \bigcup_{0\leq n+m<k} \blacktriangleright^n (\blacktriangleleft^m (\blacklozenge(L), L), L) = \bigcup_{0\leq n+m<k} \blacktriangleleft^n (\blacktriangleright^m (\blacklozenge(L), L), L).$$

Proof. We prove the first relation only; the second one can be easily proved analogously. It is plain that

$$\bigcup_{0\le n+m<k} \blacktriangleright^{n} \left(\blacktriangleleft^{m} \left(\blacklozenge(L), L\right), L\right) \subseteq ((\blacklozenge_r^k(L)) \setminus L).$$

Let $w \in (\blacklozenge_r^k(L) \setminus L)$, we prove that $w \in \blacktriangleright^{n} (\blacktriangleleft^{m} (\blacklozenge(L), L), L)$ for some $0 \le n+m < k$ by induction on k. The assertion is immediately true for $k = 1$. A simple observation makes the assertion true for $k = 2$ as well. Indeed, it is clear that $((x\blacktriangle y) \cup (x\blacktriangledown y)) \subseteq \blacklozenge(L)$ for any $x \in \blacklozenge(L)$ and $y \in L$.

Let $w \in \blacklozenge_r^{k+1}(L)$ for some $k \ge 2$; there exist $x \in \blacklozenge_r^k(L)$ and $y \in L$ such that $w \in x\blacklozenge y$. We distinguish four cases:

1. $w \in x \blacktriangleright y$. By the induction hypothesis, $x \in \blacktriangleright^{n} (\blacktriangleleft^{m} (\blacklozenge(L), L), L)$ for some $0 \le n+m < k$, hence $w \in \blacktriangleright^{n+1} (\blacktriangleleft^{m} (\blacklozenge(L), L), L)$ with $0 \le n+1+m < k+1$ holds.
2. $w \in x \blacktriangleleft y$. If $x \in \blacktriangleright^{0} (\blacktriangleleft^{m} (\blacklozenge(L), L), L)$, then $w \in \blacktriangleright^{0} (\blacktriangleleft^{m+1} (\blacklozenge(L), L), L)$ holds. Assume that $x \in \blacktriangleright^{n} (\blacktriangleleft^{m} (\blacklozenge(L), L), L)$ with $n \ge 1$; it follows that there exist $u \in \blacktriangleright^{n-1} (\blacktriangleleft^{m} (\blacklozenge(L), L), L)$ and $v \in L$ such that $x \in u \blacktriangleright v$. It further follows that $\{u, \overline{u}\} \subseteq \blacklozenge_r^{n+m}$. As $n + m < k$, we infer that $\blacktriangleleft (\{u, \overline{u}\}, y) \subset \blacklozenge_r^k(L)$. Further, $\blacktriangle(\{u, \overline{u}\}, y) \subset \blacklozenge_r^k(L)$ (see also the next item) and $w \in s \blacktriangleright v$ for some $s \in \blacktriangleleft (\{u, \overline{u}\}, y)$. By the induction hypothesis, $s \in \blacktriangleright^{p} (\blacktriangleleft^{q} (\blacklozenge(L), L), L)$ holds for some $0 \le p+q < k$, therefore $w \in \blacktriangleright^{p+1} (\blacktriangleleft^{q} (\blacklozenge(L), L), L)$ with $0 \le p+1+q < k+1$ holds as well.
3. $w \in x\blacktriangle y$. This case immediately leads to $w \in \blacklozenge_r^k(L)$.
4. $w \in x\blacktriangledown y$. This case immediately leads to $w \in \blacklozenge_r^k(L)$ and we are done.

□

A direct consequence of this lemma is the following corollary.

Corollary 1. *For every language L, the following relations hold:*

$$\blacklozenge_r^*(L) = \blacktriangleright^{*} (\blacktriangleleft^{*} (\blacklozenge(L), L), L) \cup L = \blacktriangleleft^{*} (\blacktriangleright^{*} (\blacklozenge(L), L), L) \cup L.$$

We still need one more result. We start with some additional notation. For two words x, y we denote

$$x \blacktriangleright_{\frac{1}{2}} y = \{uw\overline{v} \mid x = uw, y = \overline{w}v \text{ for some } u \in V_1^*, w \in V_1^+, v \in V_2^*\}$$
$$x \blacktriangleleft_{\frac{1}{2}} y = \{\overline{u}wv \mid x = wv, y = u\overline{w} \text{ for some } u \in V_2^*, w \in V_1^+, v \in V_1^*\}.$$

For two languages L_1, L_2 we define

$$(i)\quad \blacktriangleleft_{\frac{1}{2}} (L_1, L_2) = \bigcup_{x\in L_1, y\in L_2} x \blacktriangleleft_{\frac{1}{2}} y,$$
$$(ii)\quad \blacktriangleleft_{\frac{1}{2}}^{0} (L_1, L_2) = L_1,$$
$$(iii)\quad \blacktriangleleft_{\frac{1}{2}}^{i+1} (L_1, L_2) = \blacktriangleleft_{\frac{1}{2}} ((\blacktriangleleft_{\frac{1}{2}}^{i} (L_1, L_2)), L_2), i \ge 0,$$
$$(iv)\quad \blacktriangleleft_{\frac{1}{2}}^{*} (L_1, L_2) = \bigcup_{i\ge 0} \blacktriangleleft_{\frac{1}{2}}^{i} (L_1, L_2).$$

The language $\blacktriangleright_{\frac{1}{2}}^{*} (L_1, L_2)$ is defined analogously.

Lemma 2. *For every language L the following relations hold:*
1. $\blacktriangleleft^*_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L}) = \blacktriangleleft^*(\blacklozenge(L), L)$.
2. $\blacktriangleright^*_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L}) = \blacktriangleright^*(\blacklozenge(L), L)$.

Proof. We prove the first relation only; the second one can be shown analogously. We first consider the inclusion $\blacktriangleleft^*_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L}) \subseteq \blacktriangleleft^*(\blacklozenge(L), L)$.

Let $w \in \blacktriangleleft^k_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L})$; the inclusion immediately holds for $k = 0$. Assume $w \in \blacktriangleleft^{k+1}_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L})$, there exist $x \in \blacktriangleleft^k_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L})$ and $y \in (L \cup \overline{L})$ such that $w \in x \blacktriangleleft_{\frac{1}{2}} y$. If $y \in L$, then $w \in \blacktriangleleft^*(\blacklozenge(L), L)$ by the induction hypothesis. If $y \in \overline{L}$, then $\overline{y} \in L$ and $w \in \overline{x} \blacktriangleleft \overline{y}$ which concludes the proof of this part as soon as we note that $\overline{x} \in \blacktriangleleft^*(\blacklozenge(L), L)$.

Conversely, let $w \in \blacktriangleleft^k(\blacklozenge(L), L)$. The converse inclusion holds for $k = 0$. Assume $w \in \blacktriangleleft^{k+1}(\blacklozenge(L), L)$; there exist $x \in \blacktriangleleft^k(\blacklozenge(L), L)$ and $y \in L$ such that $w \in x \blacktriangleleft y$. If $w = \overline{u}wv$, where $x = wv, y = u\overline{w}$, then $w \in \blacktriangleleft^*_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L})$. If $w = u\overline{wv}$, where $x = wv, y = u\overline{w}$, then $w \in \overline{x} \blacktriangleleft_{\frac{1}{2}} \overline{y}$. Since $\overline{y} \in \overline{L}$ and $\overline{x} \in \blacktriangleleft^k(\blacklozenge(L), L)$, hence $x \in \blacktriangleleft^*_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L})$ by the induction hypothesis. The proof is now complete. □

Corollary 2. *For every language L, the following relations hold:*

$$\blacklozenge^*_r(L) = \blacktriangleright^*_{\frac{1}{2}}(\blacktriangleleft^*_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L}), L \cup \overline{L}) \cup L = \blacktriangleleft^*_{\frac{1}{2}}(\blacktriangleright^*_{\frac{1}{2}}(\blacklozenge(L), L \cup \overline{L}), L \cup \overline{L}) \cup L.$$

We are now ready to prove one of the main results of this note.

Theorem 2. $\blacklozenge^*(L)$ *is always regular for any regular language L. In other words, the class of regular languages is closed under iterated superposition.*

Proof. We start by recalling a result proved in [2], namely $\blacklozenge(L)$ is regular for every regular language L. By the previous corollary, it suffices to prove that $\blacktriangleright^*_{\frac{1}{2}}(E, L \cup \overline{L})$ is regular provided that E is regular.

Let $L \subseteq V^*$ be a regular language; we assume that the deterministic finite automaton $A = (Q, V, \delta, q_0, F)$ accepts $L \cup \overline{L}$. For every state $q \in Q$ we define the regular language $R(q) = (V \cup \overline{V})^*\{\overline{w} \mid w \in V^+, \delta(q_0, w) = q\}$ accepted by the (not necessarily deterministic) finite automaton $A^{(q)} = (Q^{(q)}, V \cup \overline{V}, \delta^{(q)}, s_0^{(q)}, \{s_f^{(q)}\})$. Note that all automata $A^{(q)}$, $q \in Q$, have a single final state.

We now define the following left-linear grammar $G = (N, V \cup \overline{V} \cup \{Z_X \mid X \subseteq (Q^{(q)} \times Q^{(q)}), q \in Q\}, S, P)$, where

$$N = \{S\} \cup (\bigcup_{q \in Q} \{[X, q], [X, q, q'] \mid X \subseteq (Q^{(q)} \times Q^{(q)}), q' \in Q\})$$
$$\cup (\bigcup_{q \in Q} \{\langle X, q\rangle, \langle X, q, q'\rangle \mid X \subseteq (Q^{(q)} \times Q^{(q)}), q' \in Q\}).$$

The set of productions P contains the following rules (each set of rules is accompanied with some explanations):

1. For every $q \in Q$, we add the complement of w as the suffix to the current word provided that $\delta(q, w) \in F$ and the current word lies in $R(q)$. We memorize to check the last condition in the pair $(s_0^{(q)}, s_f^{(q)})$ which must be completed to obtain a non-deterministically guessed correct recognition.

$$\begin{aligned}
&S \to [\{(s_0^{(q)}, s_f^{(q)})\}, q] \text{ for all } q \in Q, X \subseteq (Q^{(q)} \times Q^{(q)})\\
&[X, q] \to [X, q, q']\overline{a}, \text{ if } \delta(q', a) \in F, \text{ for all } q, q' \in Q, X \subseteq (Q^{(q)} \times Q^{(q)})\\
&[X, q, q'] \to [X, q, q'']\overline{a}, \text{ if } q' \in \delta(q'', a), \text{ for all } q, q', q'' \in Q, X \subseteq (Q^{(q)} \times Q^{(q)})\\
&[X, q, q] \to X \text{ for all } q \in Q, X \subseteq (Q^{(q)} \times Q^{(q)}).
\end{aligned}$$

2. For every $q \in Q$, we add the complement of w as the suffix to the current word provided that $\delta(q, w) \in F$ and the current word lies in $R(q)$. Beside memorizing to check the last condition in the pair $(s_0^{(q)}, s_f^{(q)})$ as above we simultaneously continue guessing the appropriate paths in all automata $A^{(q')}$ for $q' \in Q$.

$$\begin{aligned}
&X \to \langle X, q\rangle \text{ for all } q \in Q, X \subseteq (Q^{(q)} \times Q^{(q)})\\
&\langle X, q\rangle \to \langle X', q, q'\rangle\overline{a}, \text{ where}
\end{aligned}$$

- $\delta(q', a) \in F$,
- if $(s^{(r)}, t^{(r)}) \in X$ for some $r \in Q$, then X' contains one pair $(s^{(r)}, p^{(r)})$ such that $t^{(r)} \in \delta^{(r)}(p^{(r)}, \overline{a})$,

$$\langle X, q, q'\rangle \to \langle X', q, q''\rangle\overline{a}, \text{ where}$$

- $\delta(q'', a) = q'$,
- if $(s^{(r)}, t^{(r)}) \in X$ for some $r \in Q$, then X' contains one pair $(s^{(r)}, p^{(r)})$ such that $t^{(r)} \in \delta^{(r)}(p^{(r)}, \overline{a})$,

$$\langle X, q, q\rangle \to X \cup \{(s_0^{(q)}, s_f^{(q)})\}.$$

3. $X \to Z_X$.

We claim $\blacktriangleright_{\frac{1}{2}}^{*}\ (E, L \cup \overline{L}) = s(L(G))$, where s is a substitution $s : (V \cup \overline{V} \cup \{Z_X \mid X \subseteq (Q^{(q)} \times Q^{(q)}), q \in Q\})^* \longrightarrow 2^{(V \cup \overline{V})^*}$ defined by $s(a) = a$ for any $a \in V \cup \overline{V}$ and

$$s(Z_X) = \{w \in (V \cup \overline{V})^* \mid \forall q \in Q, \forall (s^{(q)}, t^{(q)}) \in X, t^{(q)} \in \delta^{(q)}(s^{(q)}, w)\} \cap E.$$

For s is a substitution by regular languages, it follows that the language $\blacktriangleright_{\frac{1}{2}}^{*}$ $(E, L \cup \overline{L})$ is regular. □

4 Restricted Superposition Distance

In this section we discuss the following problem: Given a language L and a word $w \in \blacklozenge_r^*(L)$, compute the minimal value i such that $w \in \blacklozenge_r^i(L)$. To this aim, we first recall a well-known problem whose solution will be useful, namely the so-called *Range Minimum (Maximum) Query Problem*: Given an array A with entries over a totally ordered set can we preprocess A (i.e., produce some additional data structures), such that we can answer efficiently queries like $RmQ_A(i,j) = \text{argmin}_{i \leq t \leq j} A[t]$ or $RMQ_A(i,j) = argmax_{i \leq t \leq j} A[t]$? In both cases, when there is a tie, the leftmost possible value is returned. Harel and Tarjan [4] proposed a solution for this problem such that, if A has n elements, then the preprocessing is performed in time $\mathcal{O}(n)$ (and a data structure of size $\mathcal{O}(n)$, called RmQ or RMQ, respectively, is produced), while the answer to any query can be obtained in time $\mathcal{O}(1)$.

Note that in the following we will use the word "derivation" with the meaning "application of the Watson-Crick superposition".

Theorem 3. *Let L be a language over the alphabet V accepted in time $\mathcal{O}(f(n))$ on the RAM model, and $w \in V^*$. One can compute the minimum value i such that $w \in \blacklozenge_r^i(L)$ in time $\mathcal{O}(|w|^2 f(|w|))$.*

Proof. Let i be the minimum value such that $w \in \blacklozenge_r^i(L)$. It follows that there exists the sequence $w_0, w_1, \ldots, w_i$, such that $w_0 \in L$, $w_i = w$, and $w_t \in (w_{t-1} \circ_t z_t)$, where $\circ_t \in \{\blacktriangleright, \blacktriangleleft, \blacktriangle, \blacktriangledown\}$ and $z_t \in L$, for all $t \in \{1, \ldots, i\}$. It is clear that both w_t and $\overline{w_t}$, for all $t \geq 1$, can be obtained in t steps starting from w_0. Without loss of generality, we may assume that w_0 is a subword of w. By Lemma 1, we may further assume that $\circ_1 \in \{\blacktriangleright, \blacktriangleleft, \blacktriangle, \blacktriangledown\}$ and $\circ_t = \blacktriangleleft$, $2 \leq t \leq p$, $\circ_t = \blacktriangleright$, $p+1 \leq t \leq i$, for some $1 \leq p \leq i$. Note that no operation $\blacktriangleleft$ or $\blacktriangleright$ is applied when $p = 1$ or $p = i$, respectively. Moreover, by the proof of Lemma 1 we may assume that $\circ_t = \blacktriangleright$ for all $2 \leq t \leq i$ provided that $\circ_1 = \blacktriangleright$.

We now focus our discussion on the way the words from L on which we can apply these operations could be identified. The idea is quite simple, and it corresponds to a greedy strategy: in the first step we identify all the subwords of w that are words in L. Then, we add to these words all the subwords of w that can be obtained from the words got in the first step using the $\blacktriangle$ and $\blacktriangledown$ operations together with their complements. Then, we try to obtain w starting from each of these words, and compute the minimum number of operations needed to do this. Consequently, we proceed as follows: if the current word is not a prefix of w, or the complement of such a prefix, we choose a word from L which we can apply the operation $\blacktriangleleft$ to, such that at least one of the obtained words is a subword of w and the length of this word is maximal among all the words that can be obtained by applying the $\blacktriangleleft$ operation to the current word.

If the current word is a prefix of w, then we choose a word from L which we can apply the operation $\blacktriangleright$ to, such that at least one of the words we obtain is a subword of w and the length of this word is maximal between all the words that can be obtained by applying the $\blacktriangleright$ operation to the current word.

It is plain that this strategy that is actually dynamic programming works as the new current word is always a subword of w. In the following, we show how this strategy can be implemented in time $\mathcal{O}(n^2 f(n))$. If the input word w belongs to $\blacklozenge_r^*(L)$, then the algorithm outputs the minimal i such that $w \in \blacklozenge_r^i(L)$, otherwise it outputs ∞. Data structures used by the algorithm are:

– The 2-dimensional arrays M, C, T, CT, D and N, with $|w|$ rows and columns.

– The 1-dimensional arrays $Left$, $Right$, $CLeft$ and $CRight$, with $|w|$ positions. The values stored in these arrays are initially set to 0.

– The queue Q that is initially empty.

Algorithm 1

```
function $Distance(w, L)$;
begin
$n := |w|$;
for $l = 1$ to $n$ do
   for $i = 1$ to $n - l + 1$ do
      $j = i + l - 1$;
      if $w[i..j] \in L$ then $M[i][j] = 1; Right[i] = j, Left[j] = i$;
         Add $([i, j], 1, 0)$ to $Q$, $D[i][j] = 1$;
      endif
      if $\overline{w[i..j]} \in L$ then $C[i][j] = 1, CRight[i] = j, CLeft[j] = i$;
   endfor
endfor
if $(D[1][n] = 1)$ then $return\ 0$;
for $l = 1$ to $n$ do
   for $i = 1$ to $n - l + 1$ do
      $j = i + l - 1$;
      $CT[i][j] = \max\{C[i][j], CT[i-1][j], CT[i][j-1]\}$;
      $T[i][j] = \max\{M[i][j], T[i-1][j], T[i][j-1]\}$;
      if $(C[i][j] = 1\ \&\ T[i][j] = 1\ \&\ D[i][j] \neq 1)$ then Add $([i, j], 1, 1), ([i, j], -1, 1)$ to $Q$,
         $N[i][j] = 1, D[i][j] = 1$;
      endif
      if $(M[i][j] = 1\ \&\ CT[i][j] = 1\ \&\ N[i][j] \neq 1)$ then Add $([i, j], -1, 1)$ to $Q$,
         $N[i][j] = 1$;
      endif
   endfor
endfor
Compute $the\ RmQ\ data\ structures\ for\ the\ arrays\ Left, CLeft$;
Compute $the\ RMQ\ data\ structures\ for\ the\ arrays\ Right, CRight$;
$b = false$
while $(b = false\ \&\ Q\ not\ empty)$ do
  Extract $([i, j], x, k)$ from $Q$
  if $(x = 1\ \&\ i \neq 1)$ then
     $t = RmQ_{CLeft}(i, j)$;
     if $D[t][j] \neq 1$ then Add $([t, j], 1, k + 1)$ to $Q$, $D[t][j] = 1$;
```

```
      if $N[t][j] \neq 1$ then Add $([t,j],-1,k+1)$ to $Q$, $N[t][j]=1$;
    endif
    if $(x=1$ & $i=1)$ then
      $t = RMQ_{CRight}(i,j)$;
      if $D[i][t] \neq 1$ then Add $([i,t],1,k+1)$ to $Q$, $D[i][t]=1$;
      if $N[i][t] \neq 1$ then Add $([i,t],-1,k+1)$ to $Q$, $N[i][t]=1$;
    endif
    if $(x=-1$ & $i \neq 1)$ then
      $t = RmQ_{Left}(i,j)$;
      if $D[t][j] \neq 1$ then Add $([t,j],1,k+1)$ to $Q$, $D[t][j]=1$;
      if $N[t][j] \neq 1$ then Add $([t,j],-1,k+1)$ to $Q$, $N[t][j]=1$;
    endif
    if $(x=-1$ & $i=1)$ then
      $t = RMQ_{Right}(i,j)$;
      if $D[i][t] \neq 1$ then Add $([i,t],1,k+1)$ to $Q$, $D[i][t]=1$;
      if $N[i][t] \neq 1$ then Add $([i,t],-1,k+1)$ to $Q$, $N[i][t]=1$;
    endif
    if $(D[1][n]=1)$ then $b = true$;
endwhile;
    if $(D[1][n]=1)$ then find in $Q$ the first tuple $([1,n],1,i), \forall i$; return $i$;
    else return $\infty$;
end.;
```

Informally, the algorithm works as follows:

- First, the subwords of w that are words from L are identified. If $w[i..j]$ is such a word then we insert the item $([i,j],1,0)$ in the queue Q. These are the only words that can be obtained from a subword of w in 0 derivation steps. Also, we use the arrays $Left$ (and $CLeft$) to store the starting position in w of the longest word from L (respectively $\overline{L}$) ending on a certain position in w, while the arrays $Right$ (and $CRight$) are used to store the ending position of the longest word from L (respectively $\overline{L}$) starting on a certain position.
- Then, we identify the words that can be obtained from a subsequence of w using a single application of the rules ▲, ▼: if $\overline{w}[i..j]$ can be obtained we add $([i,j],-1,1)$ to Q, if $w[i..j]$ can be obtained we add $([i,j],1,1)$ to Q.
- Finally, using the queue Q (in which the tuples $([i,j],x,k)$ are ordered increasingly according to the value k), we try to obtain new words (actually, the longest words) that can be derived from a subword of w, and still remain subwords of w or $\overline{w}$. Each time when a new item $([i,j],1,k)$ is extracted from Q we try to extend it as much as possible to the left (if $i \neq 1$), or to the right (if $i = 1$), and add items corresponding to the newly obtained words to the queue. The same strategy is used in the case when an item of the form $([i,j],-1,k)$ is extracted from Q.
- The algorithm ends in two situations (i) no more words can be obtained and w was not obtained yet, when it returns ∞, (ii) w was obtained, when it returns the minimum number of derivation steps used to obtain this word.

It is clear that the first two for cycles can be executed in $\mathcal{O}(n^2 f(n))$ time, while the next two ones can be executed in $\mathcal{O}(n^2)$ time. Next, Q contains an item with the first component $[i, j]$ at most two times during the computation, thus the maximum number of elements that may enter in Q is $\mathcal{O}(n^2)$. Consequently, the while cycle is executed at most $\mathcal{O}(n^2)$ times and every computation done in this cycle is executed in constant time. This shows that the overall time complexity of the above algorithm is $\mathcal{O}(n^2 f(n))$. The space needed by this algorithm is $\mathcal{O}(n^2 S(n))$, given that L is accepted by a RAM in $\mathcal{O}(f(n))$ time and $\mathcal{O}(S(n))$ space. □

It is clear that using some preprocessing which replaces the part of the algorithm consisting of the first two for cycles we can obtain an overall complexity of $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^2)$ space for regular languages. In the case of context-free languages we can obtain an overall complexity of $\mathcal{O}(n^3)$ time, using the Cocke-Younger-Kasami algorithm in the preprocessing phase, and $\mathcal{O}(n^2)$ space.

As a corollary of the previous theorem we can state:

Theorem 4. *The class* **P** *is closed under iterated superposition.*

References

1. Bottoni, P., Mauri, G., Mussio, P., Păun, G.: Grammars working on layered strings. Acta Cybernetica 13, 339–358 (1998)
2. Bottoni, P., Labella, A., Manca, V., Mitrana, V.: Superposition based on Watson-Crick-like complementarity. Theory of Computing Systems 39, 503–524 (2006)
3. Freund, R., Păun, G., Rozenberg, G., Salomaa, A.: Bidirectional sticker systems. In: Altman, R.B., Dunker, A.K., Hunter, L., Klein, T.E. (eds.) Third Annual Pacific Conf. on Biocomputing, Hawaii, pp. 535–546. World Scientific, Singapore (1998)
4. Harel, D., Tarjan, R.E.: Fast algorithms for finding nearest common ancestors. SIAM J. Comput. 13(2), 338–355 (1984)
5. Kari, L., Păun, G., Rozenberg, G., Salomaa, A., Yu, S.: DNA computing, sticker systems, and universality. Acta Informatica 35(5), 401–420 (1998)
6. Kari, L., Rozenberg, G.: The many facets of Natural Computing. Communications of the ACM 51(10), 72–83 (2008)
7. Kobayashi, S., Mitrana, V., Păun, G., Rozenberg, G.: Formal properties of PA-matching. Theoretical Comput. Sci. 262(1-2), 117–131 (2001)
8. Păun, G., Rozenberg, G., Salomaa, A.: DNA Computing. New Computing Paradigms. Springer, Berlin (1998); Tokyo, 1999
9. Păun, G., Rozenberg, G.: Sticker systems. Theoret. Comput. Sci. 204, 183–203 (1998)
10. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, vol. 3. Springer, Heidelberg (1997)

A Weighted μ-Calculus on Words

Ingmar Meinecke*

Institut für Informatik, Universität Leipzig,
D-04109 Leipzig, Germany
meinecke@informatik.uni-leipzig.de

Abstract. We define a weighted μ-calculus on finite and infinite words. Hereby, the μ-calculus does not use conjunction and the weights are taken from semirings satisfying certain completeness and continuity properties. For important semirings like distributive complete lattices, the tropical and the probabilistic semiring, we show that the formulas of the conjunction-free weighted μ-calculus define exactly the class of omega-rational formal power series.

Introduction

Quantitative aspects of systems comprise such different concepts like time, probabilities, capacities, and reasoning of several agents. Some give rise to multi-valued logics where the semantic domain is not longer the two-valued Boolean algebra but a De Morgan algebra. Several authors explored the model checking problem for multi-valued Kripke structures and different temporal logics, see [15,3,12,18] for an incomplete list of papers. Others considered quantitative games and model checking of a quantitative μ-calculus with discount factors, see *e.g.*[13].

Another approach is the one of weighted automata. Thereby, weights are taken from a semiring. A coincidence between weighted automata over finite words and rational formal power series was shown by Schützenberger [20]. Weighted automata on infinite words were explored first in recent years. To define and to characterize the behavior of those automata the underlying semiring has either to meet several completeness properties [11,10,9,7] or the automata work with discounting [8]. A characterization of weighted automata by weighted MSO logic was first given in [5].

Here, we propose a weighted μ-calculus for the description of weighted systems over finite and infinite words. Thereby, we hope to prepare the ground for weighted modal and temporal logics which may turn out as a better specification language of weighted systems. For Kripke structures the modal μ-calculus [16] has a central position between automata and logics and a close relationship to temporal logics. One kind of Kripke structures are finite and infinite words. For them, the classical μ-calculus defines exactly the class of rational and ω-rational languages [19,1], see also [2].

Our weighted μ-calculus introduced in Section 2 allows for weighted modalities (with weights from the semiring), disjunction, and least and greatest fixed point operator. Negation is excluded because in general semirings elements do not have a complement (in contrast to Boolean algebras). Conjunction exceeds the expressiveness of

* The author was supported by the German Research Foundation (DFG).

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 384–395, 2009.

weighted automata and is, therefore, excluded. Thus it is different from other quantitative μ-calculi which consider mostly very concrete weight structures and/or work with discounting, *cf.* [13,4].

Our main result, Theorem 4.5, characterizes the expressiveness of the weighted conjunction-free μ-calculus over words exactly as the ω-rational formal power series. A main obstacle towards this result is to find the necessary preconditions that have to be imposed on the underlying semiring. We started from the established concept of continuous semirings, *cf.* [6]. But whereas continuous semirings are continuous just with respect to the supremum, here, we need continuity also with respect to the infimum. Therefore, we introduce dual-continuous semirings in Section 1. However, even then the characterization of greatest fixed points by rational means does not succeed automatically. Therefore, we fix the central *Arden fixed point property* (that is needed for the main result and which is derived naturally from the Boolean setting) in Section 3 and show that this property is satisfied by several important semirings: distributive complete lattices (covering *e.g.* the fuzzy semiring and De Morgan algebras from multi-valued logics), the tropical semiring with the two operations $\min$ and $+$ which is useful for temporal aspects and optimization, and the probabilistic semiring for probabilistic settings. However, it turns out that for some common semirings like $\mathbb{N}$ with usual addition and multiplication the situation is more complicated and we have to consider an unusual infinite product in this case. Once established the Arden fixed point property, we adapt concepts presented in [2] to our setting to show the characterization result in Section 4.

1 Semirings, Formal Power Series, and Fixed Points

Let A be an alphabet. Let A^* and A^ω be the sets of finite and infinite words, respectively, and $A^\infty = A^* \cup A^\omega$. We consider *formal power series*, *i.e.*, mappings $S : A^\infty \to \mathbb{K}$ where $\mathbb{K} = (K, \oplus, \circ, \mathbb{0}, \mathbb{1})$ is a semiring, *i.e.*, a set K with two binary operations, called addition $\oplus$ and multiplication $\circ$, such that $(K, \oplus, \mathbb{0})$ is a commutative monoid, $(K, \circ, \mathbb{1})$ is a monoid, $\oplus$ distributes over $\circ$, $\mathbb{0} \neq \mathbb{1}$, and $\mathbb{0} \circ k = k \circ \mathbb{0} = \mathbb{0}$ for every $k \in K$.

To define a μ-calculus semantics, we have to consider structures where least and greatest fixed points of certain mappings exist.

1.1 Continuous and Dual-Continuous Semirings

Let $\leq$ be a partial order on K. A *chain* $C \subseteq K$ is a totally ordered subset of K. $D \subseteq K$ is *directed* if for all $k_1, k_2 \in D$ there is a $k \in D$ such that $k_1 \leq k$ and $k_2 \leq k$. Following [6], a *continuous semiring* $\mathbb{K}$ is a semiring $\mathbb{K} = (K, \oplus, \circ, \mathbb{0}, \mathbb{1})$ together with a partial order $\leq$ on K such that $\mathbb{0} \leq k$, each chain $C \subseteq K$ has a supremum $\sup C$, $k \oplus \sup C = \sup(k \oplus C)$, $k \circ \sup C = \sup(k \circ C)$, and $\sup C \circ k = \sup(C \circ k)$ for all chains C and $k \in K$. An alternative but equivalent definition would use directed sets instead of chains, *cf.* [6]. Note that in every continuous semiring $\mathbb{K}$ both addition and multiplication are monotone operations. Now we equip a continuous semiring $\mathbb{K}$ for every index set I and all families $(k_i \in K \mid i \in I)$ with *generalized sums* as follows

$$\sum_{i \in I} k_i = \sup\left\{ \sum_{i \in E} k_i \;\middle|\; E \text{ finite}, E \subseteq I \right\}. \tag{1}$$

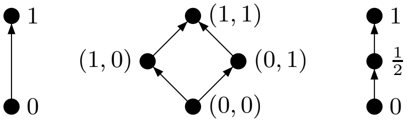


Fig. 1. Three De Morgan algebras from multi-valued logics

These sums are well-defined because $\{\sum_{i\in E} k_i \mid E \text{ finite}, E \subseteq I\}$ is directed. For posets K and K', a function $f : K' \to K$ is $\sup$-*continuous* if for any chain $C \subseteq K'$ such that $\sup C$ exists, $f(C)$ is a chain in K, has a supremum, and $f(\sup C) = \sup f(C)$. Similarly, $\inf$-*continuity* of f is defined. Continuity implies monotonicity. For an index set I, let K^I be ordered by the pointwise order.

Proposition 1.1. *Let $\mathbb{K}$ be a continuous semiring. Then the generalized sums $\sum_I : K^I \to K$ as defined in (1) are $\sup$-continuous for every index set I.*

Now we consider not only the supremum but also the infimum.

Definition 1.2. *$\mathbb{K}$ is a* dual-continuous semiring, *or a* dc-semiring *for short, if $\mathbb{K}$ is continuous, each chain $C \subseteq K$ has an infimum, $k \oplus \inf C = \inf(k \oplus C)$, $k \circ \inf C = \inf(k \circ C)$, and $\inf C \circ k = \inf(C \circ k)$ for all chains C and $k \in K$.*

Note that the infimum of the empty chain is the greatest element $\top$ of $\mathbb{K}$.

Example 1.3. The tropical semirings $\mathbb{T} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ and $\mathbb{T}_R = (\mathbb{R}^{\geq 0} \cup \{\infty\}, \min, +, \infty, 0)$with the partial order $r_1 \leq r_2$ if and only if $\min(r_1, r_2) = r_2$ (the converse of the usual order) are dc-semirings.

Example 1.4. $\mathbb{A} = (\mathbb{N} \cup \{-\infty, \infty\}, \max, +, -\infty, 0)$ together with the usual order is a dc-semiring when we put for all $n \in \mathbb{N} \cup \{-\infty, \infty\}$: $\max(n, \infty) = \infty$ and $-\infty + n = -\infty$. Generalized sums are given by sup.

Example 1.5. Distributive complete lattices with sup as addition and inf as multiplication are dc-semirings. Instances are $(\mathbb{N} \cup \{\infty\}, \min, \max, \infty, 0)$, the fuzzy semiring $([0, 1], \sup, \inf, 0, 1)$, the powerset semiring $(\mathfrak{P}(M), \cup, \cap, \emptyset, M)$ for any set M, or De Morgan algebras which are common in multi-valued logics, *cf.* [3]. In Figure 1 on the left the two-valued Boolean algebra $\mathbb{B}$ is shown. The intuition behind the Boolean algebra in the middle is that two agents evaluate a proposition independently as either false or true. The De Morgan algebra on the right is not a Boolean algebra anymore. Here, $\frac{1}{2}$ expresses uncertainty.

Example 1.6. The *probabilistic semiring* $\mathbb{P} = ([0, 1], \max, \cdot, 0, 1)$ together with the usual order $\leq$ is a dc-semiring. $\mathbb{P}$ is isomorphic to the dc-semiring $\mathbb{T}_R$ from Example 1.3 by $f : \mathbb{P} \to \mathbb{T}_R$ with $f(r) = \ln \frac{1}{r}$ for $r > 0$ and $f(0) = +\infty$.

Example 1.7. The extended natural numbers $\mathbb{N}^\infty = (\mathbb{N} \cup \{\infty\}, +, \cdot, 0, 1)$ constitute a semiring when $n + \infty = \infty$ for all $n \in \mathbb{N} \cup \{\infty\}$ and $n \cdot \infty = \infty$ for all $n \in (\mathbb{N} \setminus \{0\}) \cup \{\infty\}$. Again, equipped with the usual order the extended natural numbers form a dc-semiring. Generalized sums behave as follows: $\sum_{i\in I} k_i = \sum_{i\in E\subseteq\mathbb{N}} k_i$ whenever E is finite and $k_i = 0$ for all $i \in \mathbb{N} \setminus E$, and $\sum_{i\in I} k_i = \infty$ otherwise.

1.2 Formal Power Series and Fixed Points

From now on let $\mathbb{K}$ *be a dc-semiring.* By $\mathbb{K}\langle\langle A^\infty\rangle\rangle$ we denote the collection of all formal power series $S : A^\infty \to \mathbb{K}$ (similarly for $\mathbb{K}\langle\langle A^*\rangle\rangle$ and $\mathbb{K}\langle\langle A^\omega\rangle\rangle$). We put $(S, w) = S(w)$. The *finitary* series $S \in \mathbb{K}\langle\langle A^*\rangle\rangle$ can be understood as series from $\mathbb{K}\langle\langle A^\infty\rangle\rangle$ with $\operatorname{supp} S = \{w \in A^\infty \mid (S, w) \neq \mathbb{0}\} \subseteq A^*$, and, similarly, for $\mathbb{K}\langle\langle A^\omega\rangle\rangle$.

Let $S, T \in \mathbb{K}\langle\langle A^\infty\rangle\rangle$. We define the *sum* $S + T \in \mathbb{K}\langle\langle A^\infty\rangle\rangle$ by $(S + T, w) = (S, w) \oplus (T, w)$ for all $w \in A^\infty$. The *Cauchy product* $S \cdot T \in \mathbb{K}\langle\langle A^\infty\rangle\rangle$ is given by $(S \cdot T, w) = \sum\big((S, u) \circ (T, v) \mid w = uv, u \in A^*, v \in A^\infty\big)$ for all $w \in A^\infty$. Let kw ($k \in K, w \in A^\infty$) denote the series with $(kw, w) = k$ and $(kw, u) = \mathbb{0}$ for $u \neq w$. Let $\mathbb{0}_{A^\infty}$ be the constant zero series. Most properties of $\mathbb{K}$ lift to series:

Proposition 1.8. *Let* $\mathbb{K}$ *be a dc-semiring. Then* $(\mathbb{K}\langle\langle A^\infty\rangle\rangle, +, \cdot, \mathbb{0}_{A^\infty}, \mathbb{1}\varepsilon)$ *together with the pointwise order is a continuous semiring. Moreover, every chain in* $\mathbb{K}\langle\langle A^\infty\rangle\rangle$ *has an infimum and the sum is* inf*-continuous.*

Remark 1.9. Let $A = \{a, b\}$ and $L = bb\{a, b\}^* \cup \{a\}$. Then $L \inf_{n\in\mathbb{N}}(L^n A^\infty) \neq \inf_{n\in\mathbb{N}}(LL^n A^\infty)$. Hence, the Cauchy product is not inf-continuous, even for $\mathbb{B}$, *i.e.*, in the setting of languages.

Theorem 1.10. *Let* $\mathbb{K}$ *be a dc-semiring,* $S \in \mathbb{K}\langle\langle A^*\rangle\rangle$*, and* $S' \in \mathbb{K}\langle\langle A^\infty\rangle\rangle$*. Then* $f : \mathbb{K}\langle\langle A^\infty\rangle\rangle \to \mathbb{K}\langle\langle A^\infty\rangle\rangle$ *with* $f(T) = S' + ST$ *is a* sup*-continuous mapping and admits a least fixed point* $\operatorname{lfp}(f)$ *and a greatest fixed point* $\operatorname{gfp}(f)$*.*

Let $(f_0, w) = \mathbb{0}$ *and* $(f^0, w) = \top$ *for all* $w \in A^\infty$*. Put* $f_{\alpha+1} = f(f_\alpha)$ *and* $f^{\alpha+1} = f(f^\alpha)$ *for all ordinals* α *and put* $f_\beta = \sup_{\alpha<\beta} f_\alpha$ *and* $f^\beta = \inf_{\alpha<\beta} f^\alpha$ *whenever* β *is a limit ordinal. Then* $\operatorname{lfp}(f) = f_\omega = \sup_{i\in\mathbb{N}} f_i$ *and there is an ordinal* $\bar\alpha$ *such that* $\operatorname{gfp}(f) = f^{\bar\alpha}$*.*

Remark 1.11. In contrary to the $\operatorname{lfp}(f)$, the approximants f^α catch $\operatorname{gfp}(f)$ in general in more than ω steps. This is due to the fact that the generalized sums and, hence, the Cauchy product are not necessarily inf-continuous (already for $\mathbb{B}$, see Remark 1.9).

2 Weighted Conjunction-Free μ-Calculus

We propose a weighted μ-calculus over finite and infinite words. Weights from a semiring $\mathbb{K}$ are attached to modal operators, *i.e.*, we express the property: “It is possible to execute an action a with weight k” which is alike a transition of a weighted automaton.

2.1 Syntax

Let $\mathcal{V}$ be a set of countably many variables. Let x, y denote variables. The formulas φ of the *weighted conjunction-free* μ*-calculus* or *weighted* $\wedge$*-free* μ*-calculus* $\mu C_\vee(A, \mathbb{K})$ (for short $\mu C_\vee$ when A and $\mathbb{K}$ are understood) are defined as follows:

$$\varphi ::= k\varepsilon \mid \langle a\rangle_k\, x \mid \langle a\rangle_k\, \varphi \mid \varphi \vee \varphi \mid \mu x.\varphi \mid \nu x.\varphi$$

where $k \in K$, $a \in A$, and $x \in \mathcal{V}$. Note that every $x \in \mathcal{V}$ is preceded by at least one modality $\langle a\rangle_k$ which is due to technical reasons. Moreover, we add for every $k \in \mathbb{K}$ a constant term $k\varepsilon$ which will suggest termination (at a final state with weight k in terms of automata). The variable x occurs *free* in φ if the occurrence of x is not bounded by a μ- or ν-operator. The set $\text{free}(\varphi)$ consists of those variables which have at least one free occurrence in φ. A formula with no free variables is a *sentence*.

Remark 2.1. Contrary to Boolean algebras (or even to De Morgan algebras) there is no notion of negation for general semirings, *cf.* [5]. Hence, we drop negation. Even if a formula $\langle a\rangle_{\mathbb{0}}\,\varphi$ says "an action a can be executed with weight $\mathbb{0}$ such that φ holds for the a-successor" which is to state "there is no a-successor where φ holds", this is not a negation. This statement does not subsume the existence of a b-successor where φ holds (in the case $A = \{a, b\}$) which would be part of the negated statement. Such an existence has to be stated positively by a b-modality.

We also do exclude the universal box modality and conjunction. In the realm of words (where we have only one successor) a box modality is not necessary. The use of conjunction would be desirable but, in a weighted setting, it causes severe problems concerning expressiveness. It will be discussed at the end of this paper.

2.2 Semantics

An *interpretation* is a mapping $\iota : \mathcal{V} \to \mathbb{K}\,\langle\!\langle A^\infty\rangle\!\rangle$. For $R \in \mathbb{K}\,\langle\!\langle A^\infty\rangle\!\rangle$, the *update* of ι at x with R is the mapping $\iota[x/R]$ where $\iota[x/R](x) = R$ and $\iota[x/R](y) = \iota(y)$ for all $y \neq x$. Now, ι is extended to a semantical function $\sigma_\iota : \mu C_\vee(A, \mathbb{K}\,) \to \mathbb{K}\,\langle\!\langle A^\infty\rangle\!\rangle$:

- $\sigma_\iota(k\varepsilon) = k\varepsilon$ and $\sigma_\iota(\psi_1 \vee \psi_2) = \sigma_\iota(\psi_1) + \sigma_\iota(\psi_2)$,
- $\sigma_\iota(\langle a\rangle_k\, x) = (ka)\,\cdot\,\iota(x)$ and $\sigma_\iota(\langle a\rangle_k\,\psi) = (ka)\,\cdot\,\sigma_\iota(\psi)$,
- for $\mu x.\psi$ let $f : \mathbb{K}\,\langle\!\langle A^\infty\rangle\!\rangle \to \mathbb{K}\,\langle\!\langle A^\infty\rangle\!\rangle : R \mapsto \sigma_{\iota[x/R]}(\psi)$, then $\sigma_\iota(\mu x.\psi) = \text{lfp}(f)$,
- similarly, for $\nu x.\psi$ we set $\sigma_\iota(\nu x.\psi) = \text{gfp}(f)$.

Here, disjunction is interpreted as the sum of the semiring, *cf.* [5]. The modal operators can be understood as weighted transitions which is formally defined by the Cauchy product. We still have to show that σ_ι is well-defined, *i.e.*, that f is a monotone function. By induction on φ we show (also *cf.* [2, Prop. 1.2.23])

Proposition 2.2. *Let $\mathbb{K}$ be a dc-semiring. Then the semantic function σ_ι is well-defined for every interpretation ι. Moreover, $\sigma_\iota(\varphi) = \sigma_{\iota'}(\varphi)$ if $\iota(y) = \iota'(y)$ for all $y \in \text{free}(\varphi)$.*

Example 2.3. Consider $\mathbb{T} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ from Example 1.3 and the formula $\varphi = \mu x.\big(\langle a\rangle_0\, x \vee \langle b\rangle_1\, x \vee \langle c\rangle_0\, 0\varepsilon\big)$. By computing the approximations from Theorem 1.10 we get[1]: $f_1 = 0.c$, $f_2 = 0.ac + 1.bc + 0.c$, and more generally $(f_n, w) = |w|_b$ if $w \in \{a, b\}^*c$ and $|w| \leq n$, and $(f_n, w) = \infty$ otherwise. Thus, we get by $\text{lfp}(f) = \sup_{n\in\mathbb{N}} f_n$ that $(\sigma(\varphi), w) = |w|_b$ if $w \in \{a, b\}^*c$ and $(\sigma(\varphi), w) = \infty$ otherwise, *i.e.*, φ defines the language $\{a, b\}^*c$ and counts for those words the number of b's.

3 Rationality and Fixed Points

As we will show in Section 4, the formulas of $\mu C_\vee(A, \mathbb{K}\,)$ define exactly the ω-rational series which we introduce next.

[1] Here $f = k.w + l.u$ means $(f, w) = k$, $(f, u) = l$, and $(f, v) = \infty$ for all other v.

3.1 Rational and ω-Rational Series

Let $S \in \mathbb{K}\langle\langle A^*\rangle\rangle$ be *proper*, *i.e.*, $(S,\varepsilon) = \mathbb{0}$. Then for any $w \in A^*$ and $i \in \mathbb{N}$ we put $(S^i, w) = \sum_{w=w_1\dots w_i}(S,w_1)\circ\ldots\circ(S,w_i)$, $(S^*,w) = \sum_{i\in\mathbb{N}}(S^i,w)$ where $S^0 = \mathbb{1}\varepsilon$, and $S^+ = SS^*$. Due to the definition of the generalized sums in (1) we get:

Proposition 3.1. *For a dc-semiring $\mathbb{K}$ and $S \in \mathbb{K}\langle\langle A^*\rangle\rangle$: $S^* = \sup_{n\in\mathbb{N}}\big(\sum_{i=0}^n S^i\big)$.*

Next we define the *ω-iteration*. For this to work we need a countable infinite product in $\mathbb{K}$ which we introduce first. In general, the infinite product may be any function $\prod : \mathbb{K}^{\mathbb{N}} \to \mathbb{K}$ such that $\prod_{i\in\mathbb{N}} k_i = \mathbb{0}$ if there is an $i \in \mathbb{N}$ with $k_i = \mathbb{0}$. A natural definition of this product can be given under an additional assumption: If $\mathbb{K}$ is a dc-semiring and $\mathbb{0} \leq k \leq \mathbb{1}$ for all $k \in \mathbb{K}$ then the sequence of finite products $(\prod_{i=0}^n k_i)_{n\in\mathbb{N}}$ is descending and we put the infinite product as

$$\prod_{i\in\mathbb{N}} k_i = \inf_{n\in\mathbb{N}}\big(\prod_{i=0}^n k_i\big). \tag{2}$$

Let $\mathbb{K}$ be a dc-semiring with infinite product and let $S \in \mathbb{K}\langle\langle A^*\rangle\rangle$ be proper. The *ω-iteration* $S^\omega \in \mathbb{K}\langle\langle A^\infty\rangle\rangle$ of S is defined by $(S^\omega, w) = \mathbb{0}$ for $w \in A^*$ and by

$$(S^\omega, w) = \sum_{\substack{w=u_0u_1u_2u_3\cdots \\ u_i\in A^+}} \prod_{i\in\mathbb{N}}(S,u_i)$$

for all $w \in A^\omega$. Here, we add over all infinite decompositions of $w \in A^\omega$ into finite non-empty words, and take the countable infinite product of the (S,u_i).

The class of *rational formal power series* is the smallest class $\mathrm{Rat}\big(\mathbb{K}\langle\langle A^*\rangle\rangle\big)$ of finitary formal power series such that $k\varepsilon, ka \in \mathrm{Rat}\big(\mathbb{K}\langle\langle A^*\rangle\rangle\big)$ for all $k \in \mathbb{K}$ and $a \in A$, and which is closed under sum, Cauchy product, and Kleene star * for proper series. The class of *ω-rational formal power series* $\omega\mathrm{Rat}\big(\mathbb{K}\langle\langle A^\infty\rangle\rangle\big)$ is the least class such that $k\varepsilon, ka \in \omega\mathrm{Rat}\big(\mathbb{K}\langle\langle A^\infty\rangle\rangle\big)$ for all $k \in \mathbb{K}$ and $a \in A$, and which is closed under sum, Cauchy product, as well as Kleene star * and ω-iteration $^\omega$ for finitary proper series. The next proposition is folklore:

Proposition 3.2. *The class of proper rational series is the least class containing the monomials ka ($k \in \mathbb{K}$, $a \in A$) which is closed under sum, Cauchy product, and iteration $^+$. Moreover, for every $R \in \mathrm{Rat}\big(\mathbb{K}\langle\langle A^*\rangle\rangle\big)$ there are a proper $R' \in \mathrm{Rat}\big(\mathbb{K}\langle\langle A^*\rangle\rangle\big)$ and $k \in \mathbb{K}$ such that $R = R' + k\varepsilon$.*

Next we give a characterization of ω-rational formal power series by rational series. It was already noted in [8] in a more particular setting.

Lemma 3.3. *For any $S \in \omega\mathrm{Rat}\big(\mathbb{K}\langle\langle A^\infty\rangle\rangle\big)$ there are a series $R_0 \in \mathrm{Rat}\big(\mathbb{K}\langle\langle A^*\rangle\rangle\big)$ and finitely many proper series $R_i, T_i \in \mathrm{Rat}\big(\mathbb{K}\langle\langle A^*\rangle\rangle\big)$ for $i = 1,\ldots,n$ such that*

$$S = R_0 + \sum_{i=1}^n R_i T_i^\omega .$$

3.2 Calculation of Fixed Points

Now we turn to the concrete calculation of the least and greatest fixed point of $f : T \mapsto S' + ST$ where $S' \in \mathbb{K}\langle\langle A^\infty \rangle\rangle$, $S \in \mathbb{K}\langle\langle A^* \rangle\rangle$ and whose existence is guaranteed by Theorem 1.10 for $\mathbb{K}$ being a dc-semiring. The next theorem is well-known for formal power series over finite words where S^*S' is the unique fixed point, *cf.* [17]. Using the approximations of Theorem 1.10 and Proposition 3.1, we obtain

Theorem 3.4. *For a dc-semiring* $\mathbb{K}$, $S' \in \mathbb{K}\langle\langle A^\infty \rangle\rangle$, $S \in \mathbb{K}\langle\langle A^* \rangle\rangle$, *and* $f : \mathbb{K}\langle\langle A^\infty \rangle\rangle \to \mathbb{K}\langle\langle A^\infty \rangle\rangle$ *with* $f(T) = S' + ST$. *Then* $\mathrm{lfp}(f) = S^*S'$.

Now we turn to the calculation of $\mathrm{gfp}(f)$. We denote by $S'\restriction_{A^*}$ and $S'\restriction_{A^\omega}$ the restriction of S' to A^* and A^ω, respectively. Moreover, let $\mathbb{1}_L$ denote the *characteristic series* of $L \subseteq A^\infty$, *i.e.*, the series with $(\mathbb{1}_L, w) = \mathbb{1}$ if $w \in L$ and $(\mathbb{1}_L, w) = \mathbb{0}$ for $w \notin L$. Now consider the following functions for $S', T \in \mathbb{K}\langle\langle A^\infty \rangle\rangle$ and $S \in \mathbb{K}\langle\langle A^* \rangle\rangle$: $f : T \mapsto S' + ST$, $g : T \mapsto S'\restriction_{A^*} + ST$, $h : T \mapsto S'\restriction_{A^\omega} + ST$, and $l : T \mapsto ST$ and define the respective approximants f^α, g^α, h_α, and l^α as in Theorem 1.10.

Lemma 3.5. *Let* $\mathbb{K}$ *be a dc-semiring,* $S' \in \mathbb{K}\langle\langle A^\infty \rangle\rangle$, $S \in \mathbb{K}\langle\langle A^* \rangle\rangle$, *and* f^α, g^α, h_α, *and* l^α *as above. Then we have for every ordinal* α: $f^\alpha = g^\alpha + h_\alpha + l^\alpha$.

Proof (sketch). For $\alpha = 0$ and a successor ordinal α the assertion is straightforward. For β a limit ordinal, we can show $\inf_{\alpha<\beta}(h_\alpha + g^\alpha + l^\alpha) = \sup_{\alpha<\beta} h_\alpha + inf_{\alpha<\beta}(g^\alpha + l^\alpha)$ which proves the assertion for β. □

The proof of the next result uses properness of S and Theorem 3.4.

Proposition 3.6. *Let* $\mathbb{K}$ *be a dc-semiring. For* g^α *and* h_α *as defined above we have:*

1. *if* $(S, \varepsilon) = \mathbb{0}$ *then* $g^\alpha = S^*S'\restriction_{A^*}$ *for all* $\alpha \geq \omega$,
2. $h_\alpha = S^*S'\restriction_{A^\omega}$ *for all* $\alpha \geq \omega$.

Finally, we come to the calculation of l^α which turns out most difficult.

Definition 3.7 (Arden fixed point property). *Let* $\mathbb{K}$ *be a dc-semiring with infinite products and* $l : \mathbb{K}\langle\langle A^\infty \rangle\rangle \to \mathbb{K}\langle\langle A^\infty \rangle\rangle : T \mapsto ST$. *Then* $\mathbb{K}$ *has the* Arden fixed point property *if* $\mathrm{gfp}(l) = S^\omega$ *for all proper* $S \in \mathbb{K}\langle\langle A^* \rangle\rangle$.

Remark 3.8. Note that $\mathrm{gfp}(f) = S^\omega$ is equivalent to the existence of an ordinal β such that $l^\alpha = S^\omega$ for all $\alpha > \beta$. It is known that in the setting of languages the Boolean semiring $\mathbb{B}$ has the Arden fixed point property which is part of the well-known Arden's Lemma, *cf.* [2, L. 5.1.4]. Our above definition lifts the property from $\mathbb{B}$ to a more general semiring setting and is, therefore, natural.

Theorem 3.9. *Let* $\mathbb{K}$ *be a dc-semiring which allows for infinite products and has the Arden fixed point property. Let* $S' \in \mathbb{K}\langle\langle A^\infty \rangle\rangle$, $S \in \mathbb{K}\langle\langle A^* \rangle\rangle$ *be proper, and* $f : \mathbb{K}\langle\langle A^\infty \rangle\rangle \to \mathbb{K}\langle\langle A^\infty \rangle\rangle : T \mapsto S' + ST$. *Then* $\mathrm{gfp}(f) = S^*S' + S^\omega$.

Proof. There is an ordinal β such that $f^\beta = \mathrm{gfp}(f)$. By Lemma 3.5, $f^\beta = g^\beta + h_\beta + l^\beta$. Applying the Arden fixed point property and Proposition 3.6, we get $f^\beta = S^*S'\restriction_{A^*} + S^*S'\restriction_{A^\omega} + S^\omega = S^*S' + S^\omega$. □

A number of important dc-semirings have the Arden fixed point property.

Lemma 3.10. *Every distributive complete lattice* $\mathbb{L} = (\mathbb{L}, \vee, \wedge, 0, 1)$ *with an infinite product as defined by* (2) *has the Arden fixed point property.*

Proof (sketch). The assumption is known for $\mathbb{B}$. First we consider $\mathbb{L} = \mathbb{B}^M$ for an arbitrary set M, *i.e.*, $\mathbb{L}$ is isomorphic to the powerset lattice of M. Then we show the assumption by considering for every $m \in M$ the mapping $l_m : T_m \mapsto S_m T_m$ where $S_m, T_m \in \mathbb{B}\langle\langle A^\infty \rangle\rangle$ are the mth projection of S and T. We get $\mathrm{gfp}(l_m) = S_m^\omega$ by Arden's Lemma, and, altogether, $\mathrm{gfp}(l) = S^\omega$. For an arbitrary distributive complete lattice $\mathbb{L}$, we get by Birkhoff's and Stone's representation theorem that $\mathbb{L}$ is isomorphic to a lattice of sets, *i.e.*, a sub-lattice of $\mathbb{B}^M$ for some set M (*cf.* [14, Thm. II.1.19]). Hence, we can assume that all series considered take values in $\mathbb{B}^M$ instead of $\mathbb{K}$. □

Lemma 3.11. $\mathbb{T} = (\mathbb{N} \cup \{\infty\}, \min, +, \infty, 0)$ *and* $\mathbb{T}_R = (\mathbb{R}^{\geq 0} \cup \{\infty\}, \min, +, \infty, 0)$ *from Example 1.3 with an infinite product as defined by* (2) *have the Arden fixed point property.*

Proof (idea). Consider $\mathbb{T}_R$. For any fixed point T, $w \in A^\omega$ with $(T, w) \neq \infty$ (the case $(T, w) = \infty$ is trivial), and an arbitrary $\delta > 0$ we show that $(S^\omega, w) < (T, w) + \delta$ where $<$ is the usual order (the converse of the order on $\mathbb{T}_R$). We do so by constructing inductively two sequences $(\tilde{u}_i)_{i \in \mathbb{N}}$ and $(\tilde{v}_i)_{i \in \mathbb{N}}$ such that $\tilde{u}_i \in A^+$ and $\tilde{v}_i \in A^\omega$ for all $i \in \mathbb{N}$, $w = \tilde{v}_0 = \tilde{u}_0 \tilde{v}_1 \ldots = \cdots = \tilde{u}_0 \tilde{u}_1 \tilde{u}_2 \ldots$, and $(T, \tilde{v}_i) + \frac{\delta}{2^{i+1}} \geq (S, \tilde{u}_i) + (T, \tilde{v}_{i+1})$. Finally, we get $S^\omega \leq T$. Since S^ω is a fixed point, it is the greatest one with respect to the order on $\mathbb{T}_R$. □

Since the probabilistic semiring $\mathbb{P}$ is isomorphic to $\mathbb{T}_R$, we conclude

Lemma 3.12. *The probabilistic semiring* $\mathbb{P} = ([0, 1], \max, \cdot, 0, 1)$ *from Example 1.6 with infinite products as defined by* (2) *has the Arden fixed point property.*

Now, consider the dc-semiring $\mathbb{N}^\infty = (\mathbb{N} \cup \{\infty\}, +, \cdot, 0, 1)$ from Example 1.7. Here, we cannot define the infinite product as in (2) because 1 is not the greatest element. A natural definition would be the following: $\prod_{i \in \mathbb{N}} n_i = \prod_{i=0}^m n_i$ if $\forall j > m : n_j = 1$, it equals 0 if $\exists j : n_j = 0$, and it equals ∞ otherwise. Let $S = 1a$. Then $S^\omega = 1a^\omega$. But $\mathrm{gfp}(l) = \infty a^\omega$. Hence, here $S^\omega < \mathrm{gfp}(l)$ and, therefore, $\mathbb{N}^\infty$ with the above infinite product does not have the Arden fixed point property. Another somehow artificial product for $\mathbb{N}^\infty$ is the following:

$$\prod_{i \in \mathbb{N}} n_i = 0 \text{ if } \exists j : n_j = 0 \text{ and } \prod_{i \in \mathbb{N}} n_i = \infty \text{ otherwise.} \tag{3}$$

Similarly, we define for the arctic semiring $\mathbb{A} = (\mathbb{N} \cup \{-\infty, \infty\}, \max, +, -\infty, 0)$ from Example 1.4 an infinite product as follows:

$$\prod_{i \in \mathbb{N}} n_i = -\infty \text{ if } \exists j : n_j = -\infty \text{ and } \prod_{i \in \mathbb{N}} n_i = \infty \text{ otherwise.} \tag{4}$$

With these infinite products we can show the Arden fixed point property:

Lemma 3.13. *The semirings* $\mathbb{N}^\infty = (\mathbb{N} \cup \{\infty\}, +, \cdot, 0, 1)$ *from Example 1.7 with an infinite product as defined in* (3) *and* $\mathbb{A} = (\mathbb{N} \cup \{-\infty, \infty\}, \max, +, -\infty, 0)$ *from Example 1.4 with an infinite product as defined in* (4) *have the Arden fixed point property.*

Proof (sketch). Consider $\mathbb{N}^\infty$. Then $\infty \mathbb{1}_{(\mathrm{supp}\, S)^\omega} = S^\omega$. By calculating the approximations l^α, one shows that $(l^\alpha, w) = \infty$ for all $w \in (\mathrm{supp}\, S)^\omega$. If $w \notin (\mathrm{supp}\, S)^\omega$, then one considers $\eta : \mathbb{N}^\infty \to \mathbb{B}$ with $0 \mapsto 0$ and $k \mapsto 1$ for $k \neq 0$. We get $\eta(l^\alpha, w) = 0$ for α sufficiently large. Hence, $(l^\alpha, w) = 0$. For $\mathbb{A}$ we conclude similarly. □

4 Expressiveness of the Weighted $\wedge$-Free μ-Calculus

4.1 Calculating the Semantics Explicitely

We had given a fixed point semantics for all $\varphi \in \mu C_\vee$. By Theorems 3.4 and 3.9 we can calculate the fixed points of certain linear functions by the star and ω-iteration. The functions defined by formulas $\varphi \in \mu C_\vee$ are of this kind as we will show next. To state this result, we introduce the extended series semantics $\llbracket \varphi \rrbracket$ which will deal with variables as pure wild-cards which remain in the semantics as symbols. Therefore, we adopt the notion of extended languages as used by Arnold and Niwiński in [2]. More precisely, an *extended series* is a pair $\langle S, \{(x, S_x) \mid x \in X\}\rangle$ where $X \subseteq \mathcal{V}$ is a finite set of variables which may be empty, $S \in \mathbb{K}\langle\!\langle A^\infty \rangle\!\rangle$, and $S_x \in \mathbb{K}\langle\!\langle A^* \rangle\!\rangle$ a proper series for every $x \in X$. We define the *extended series semantics* by induction:

if $\varphi = k\varepsilon$, then $\llbracket \varphi \rrbracket = \langle k\varepsilon, \emptyset \rangle$,
if $\varphi = \langle a \rangle_k\, x$, then $\llbracket \varphi \rrbracket = \langle \mathbb{0}_{A^\infty}, \{(x, ka)\}\rangle$
where $\mathbb{0}_{A^\infty}$ is the constant zero series,
if $\varphi = \langle a \rangle_k\, \psi$, then $\llbracket \varphi \rrbracket = \langle (ka)T, \{(x, (ka)T_x) \mid x \in X\}\rangle$
where $\llbracket \psi \rrbracket = \langle T, \{(x, T_x) \mid x \in X\}\rangle$,
if $\varphi = \psi_1 \vee \psi_2$, then $\llbracket \varphi \rrbracket = \langle S_1 + S_2, \{(x, S_{1x} + S_{2x}) \mid x \in X_1 \cup X_2\}\rangle$
where $\llbracket \psi_j \rrbracket = \langle S_j, \{(x, S_{jx}) \mid x \in X_j\}\rangle$ for $j = 1, 2,$[1]
if $\varphi = \mu x.\psi$, then $\llbracket \varphi \rrbracket = \langle T_x^* T, \{(y, S_y = T_x^* T_y) \mid y \in X \setminus \{x\}\}\rangle$
where $\llbracket \psi \rrbracket = \langle T, \{(y, T_y) \mid y \in X\}\rangle$,
if $\varphi = \nu x.\psi$, then $\llbracket \varphi \rrbracket = \langle T_x^* T + T_x^\omega, \{(y, T_x^* T_y) \mid y \in X \setminus \{x\}\}\rangle$
where $\llbracket \psi \rrbracket = \langle T, \{(y, T_y) \mid y \in X\}\rangle$.

Using extended series we can prove inductively by help of Theorems 3.4 and 3.9

Theorem 4.1. *Let* $\mathbb{K}$ *be a dc-semiring with infinite products and the Arden fixed point property. Let* $\varphi \in \mu C_\vee(\mathbb{K}, A)$ *with* $\llbracket \varphi \rrbracket = \langle S, \{(x, S_x) \mid x \in X\}\rangle$ *and* ι *an interpretation. Then* $X = \mathrm{free}(\varphi)$ *and*

$$\sigma_\iota(\varphi) = S + \sum_{x \in X} S_x \iota(x).$$

Example 4.2. Consider again the formula $\varphi = \mu x.\big(\langle a \rangle_0\, x \vee \langle b \rangle_1\, x \vee \langle c \rangle_0\, \varepsilon\big)$ over the tropical semiring $\mathbb{T}$ from Example 2.3. Now we get first $\llbracket \langle a \rangle_0\, x \vee \langle b \rangle_1\, x \vee \langle c \rangle_0\, \varepsilon \rrbracket =$

[1] Here, we agree upon $S_{1x} = \mathbb{0}_{A^*}$ if $x \notin X_1$, and $S_{2x} = \mathbb{0}_{A^*}$ if $x \notin X_2$.

$\big\langle 0c, \{(x, 0a+1b)\}\big\rangle$. Finally, we have $[\![\varphi]\!] = \big\langle (0a+1b)^*(0c), \emptyset\big\rangle$ and thus $\big(\sigma(\varphi), w\big) = |w|_b$ if $w \in \{a,b\}^*c$ and ∞ otherwise.

For $\varphi = \nu y.\mu x.\big(\langle a\rangle_0\, y \vee \langle b\rangle_1\, x \vee \langle c\rangle_0\, \varepsilon\big)$ we get $[\![\varphi]\!] = \big\langle ((1b)^*(0a))^*(1b)^*(0c) + ((1b)^*(0a))^\omega, \emptyset\big\rangle$. Hence, we have $\big(\sigma(\varphi), w\big) = |w|_b$ if $w \in \{a,b\}^*c \cup \{b^*a\}^\omega$ and ∞ otherwise. Again, we count the number of b's but this time for finite words from $\{a,b\}^*c$ and also for infinite words from $(b^*a)^\omega$.

4.2 The Characterization Theorem

By the very definition of the extended series semantics we get

Proposition 4.3. *Let $\mathbb{K}$ be a dc-semiring with infinite products and the Arden fixed point property. Let $\varphi \in \mu C_\vee(A, \mathbb{K})$ with $[\![\varphi]\!] = \langle S, \{(x, S_x) \mid x \in X\}\rangle$. Then $S \in \omega\mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^\infty\rangle\!\rangle\big)$ and $S_x \in \mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^*\rangle\!\rangle\big)$ for all $x \in X$.*

Now we turn to the converse.

Proposition 4.4. *Let $\mathbb{K}$ be a dc-semiring with infinite products and the Arden fixed point property. Let $X \subseteq \mathcal{V}$ be finite, $S \in \omega\mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^\infty\rangle\!\rangle\big)$, and $S_x \in \mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^*\rangle\!\rangle\big)$ be proper for every $x \in X$. Then there is a formula $\varphi \in \mu C_\vee(A, \mathbb{K})$ with* $\mathrm{free}(\varphi) = X$ *and $[\![\varphi]\!] = \langle S, \{(x, S_x) \mid x \in X\}\rangle$.*

Proof. By Proposition 3.2, every proper rational S can be generated from the monomials ka by a finite number of applications of sum, Cauchy product, and iteration $^+$. First we show that for every proper $S \in \mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^*\rangle\!\rangle\big)$ and any variable $x \in \mathcal{V}$ there is a formula φ with $\mathrm{free}(\varphi) = \{x\}$ and $[\![\varphi]\!] = \langle \mathbb{0}_{A^\infty}, \{(x, S)\}\rangle$. For $S = ka$ we put $\varphi = \langle a\rangle_k\, x$. If $S = S_1 + S_2$ with $[\![\varphi_j]\!] = \langle \mathbb{0}_{A^\infty}, \{(x, S_j)\}\rangle$ for $j = 1, 2$, then we take $\varphi = \varphi_1 \vee \varphi_2$.

Next let $S = S_1 S_2$ where again $[\![\varphi_j]\!] = \langle \mathbb{0}_{A^\infty}, \{(x, S_j)\}\rangle$ for $j = 1, 2$. Now we can define the substitution $\varphi = \varphi_1[x/\varphi_2]$ such that $\mathrm{free}(\varphi) = \{x\}$ and $[\![\varphi]\!] = \langle \mathbb{0}_{A^\infty}, \{(x, S_1 S_2)\}\rangle$ (we omit the technical details).

If $S = S'^+$ with $[\![\varphi']\!] = \langle \mathbb{0}_{A^\infty}, \{(x, S')\}\rangle$ then we take $\varphi = \mu y.(\varphi'' \vee \varphi')$ where y is some fresh variable and $\varphi'' = \varphi'[x/y]$. Now we get $[\![\varphi]\!] = \langle T, \{(x, S)\}\rangle$ with

$$T = S'^*(\mathbb{0}_{A^\infty} + \mathbb{0}_{A^\infty}) = \mathbb{0}_{A^\infty} \text{ and } S = S'^*(\mathbb{0}_{A^\infty} + S') = S'^+.$$

Hence, the first claim is proven. Next, we turn to the variable-free part of the extended series. By the first claim we have for every proper $T \in \mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^*\rangle\!\rangle\big)$ a formula φ with $[\![\varphi]\!] = \langle \mathbb{0}_{A^\infty}, \{(x, T)\}\rangle$. Therefore, $\psi = \nu x.\varphi$ has the semantics $[\![\psi]\!] = \langle T^\omega, \emptyset\rangle$. Now take η with $[\![\eta]\!] = \langle \mathbb{0}_{A^\infty}, \{(x, R')\}\rangle$ for an arbitrary proper $R' \in \mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^*\rangle\!\rangle\big)$. Again we get with an appropriate substitution (again skipping the technicalities) $\eta' = \eta[x/\psi]$ the semantics $[\![\eta']\!] = \langle R'T^\omega, \emptyset\rangle$. Finally, for $\eta'' = \eta[x/\mathbb{1}\varepsilon]$ we have $[\![\eta'']\!] = \langle R', \emptyset\rangle$. By Proposition 3.2, there are for $R \in \mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^*\rangle\!\rangle\big)$ a proper $R' \in \mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^*\rangle\!\rangle\big)$ and $k \in \mathbb{K}$ such that $R = R' + k\varepsilon$. Hence, $\sigma = \eta'' \vee k\varepsilon$ has semantics $\langle R, \emptyset\rangle$. In view of Lemma 3.3, for every $S \in \omega\mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^\infty\rangle\!\rangle\big)$ there is a sentence $\varphi \in \mu C_\vee(A, \mathbb{K})$ with $[\![\varphi]\!] = \langle S, \emptyset\rangle$.

Let $S_x \in \mathrm{Rat}\big(\mathbb{K}\,\langle\!\langle A^*\rangle\!\rangle\big)$ be proper and φ_x a formula with extended series semantics $[\![\varphi_x]\!] = \langle \mathbb{0}_{A^\infty}, \{(x, S_x)\}\rangle$ whose existence is guaranteed by the first claim. Then $\psi = \varphi \vee \bigvee_{x \in X} \varphi_x$ defines $\langle S, \{(x, S_x) \mid x \in X\}\rangle$. □

Now we get immediately by Propositions 4.3, 4.4, and by Theorem 4.1

Theorem 4.5. *Let $\mathbb{K}$ be a dc-semiring with infinite products and the Arden fixed point property, A an alphabet, and $S \in \mathbb{K}\langle\langle A^\infty \rangle\rangle$. Then the following are equivalent:*

1. *S is ω-rational,*
2. *there is a sentence φ of the weighted $\wedge$-free μ-calculus with $[\![\varphi]\!] = \langle S, \emptyset \rangle$ and $\sigma(\varphi) = S$.*

A look on the proof of Proposition 4.4 shows that every ω-rational series is already the component of the semantics of some formula where the greatest fixed point operator ν alternates only once with the least fixed point operator μ. Moreover, if S is rational then S can be already defined by a sentence without a greatest fixed point operator.

Remark 4.6. The equivalence between weighted Büchi automata and ω-rationality was shown in [9, Thm. 30] for *idempotent*, *o-complete*, and *infinitely distributive* semirings. Roughly speaking, o-complete semirings $\mathbb{K}$ have to be ordered as a complete lattice, $\mathbb{0} \leq k \leq \mathbb{1}$ for all $k \in \mathbb{K}$, and both addition and multiplication commute with arbitrary suprema and infima. Hence, every o-complete $\mathbb{K}$ is dual-continuous but not necessarily vice versa. The other two conditions from [9, Thm. 30] are not needed for the result on the μ-calculus. In this sense, our setting is more general. On the other side, we have to guarantee the Arden fixed point property. Therefore, it is not clear how these two classes of semirings relate exactly. However, distributive complete lattices, the tropical semirings $\mathbb{T}$ and $\mathbb{T}_R$, and the probabilistic semiring are idempotent, o-complete, and infinitely distributive. Hence, for these semirings a formal power series S is definable in $\mu C_\vee(A, \mathbb{K})$ if and only if it is the behavior of some weighted Büchi automaton.

By results from [5] and [7] we get also an equivalence of weighted (Büchi) automata and weighted MSO logic provided $\mathbb{K}$ is a distributive complete lattice.

Open Problems

A central concept for our results was the Arden fixed point property which we have shown for some important semirings. Hereby, the non-continuity of the Cauchy product with respect to infima is the major difficulty. However, it would be nice to find a uniform proof of this property for a number of semirings or even a characterization of the class of semirings possessing the Arden fixed point property.

Another point of interest is conjunction. For a weighted MSO logics, conjunction is interpreted as the pointwise product of the respective semantics [5], *i.e.*, $([\![\varphi \wedge \psi]\!], w) = ([\![\varphi]\!], w) \circ ([\![\psi]\!], w)$. Then *e.g.* the semantics of the formula $\varphi = \mu y.(0\varepsilon \vee [\mu x.(\langle a \rangle_1\, x \vee 0\varepsilon) \wedge \langle a \rangle_0\, y])$ over the tropical semiring $\mathbb{T}$ is not ω-rational anymore (and here we do not even use the ν-operator). However, we would suppose that for complete distributive lattices the semantics stays within ω-rational series. For other semirings there may be other fragments using conjunction which do not exceed ω-rationality. Another possible approach could be an alternative semantics for conjunction, *e.g.* the infimum of the two respective semantics, *cf.* [13]. For the tropical semiring $\mathbb{T}$, this sounds reasonable. It has to be clarified whether such a new approach would preserve ω-rationality.

A broader perspective is to establish a semiring-based weighted μ-calculus and temporal logics as a specification language for weighted systems. Thereby, connections to other quantitative logics should be carefully explored and model checking techniques should be established at least for certain classes of semirings.

Acknowledgement. The author would like to thank Paul Gastin and Manfred Droste for some thought-provoking impulses.

References

1. Arnold, A., Niwiński, D.: Fixed point characterization of weak monadic logic definable set of trees. In: Nivat, M., Podelski, A. (eds.) Tree Automata and Languages, pp. 159–188. Elsevier, Amsterdam (1992)
2. Arnold, A., Niwiński, D.: Rudiments of μ-Calculus. Studies in Logic and the Foundations of Mathematics, vol. 146. North-Holland, Amsterdam (2001)
3. Bruns, G., Godefroid, P.: Model checking with multi-valued logics. In: Díaz, J., Karhumäki, J., Lepistö, A., Sannella, D. (eds.) ICALP 2004. LNCS, vol. 3142, pp. 281–293. Springer, Heidelberg (2004)
4. de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the future in systems theory. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 1022–1037. Springer, Heidelberg (2003)
5. Droste, M., Gastin, P.: Weighted automata and weighted logics. Theoretical Computer Science 380, 69–86 (2007)
6. Droste, M., Kuich, W.: Semirings and formal power series. In: Droste, M., Kuich, W., Vogler, H. (eds.) Handbook of Weighted Automata, ch. 1. Springer, Heidelberg (to appear, 2009)
7. Droste, M., Kuich, W., Rahonis, G.: Multi-valued MSO logics over words and trees. Fundamenta Informaticae 84, 305–327 (2008)
8. Droste, M., Kuske, D.: Skew and infinitary formal power series. Theoretical Computer Science 366, 199–227 (2006)
9. Droste, M., Püschmann, U.: On weighted Büchi automata with order-complete weights. Algebra and Computation 17(2), 235–260 (2007)
10. Droste, M., Rahonis, G.: Weighted automata and weighted logics on infinite words. Izvestiya VUZ. Matematika (2008)
11. Ésik, Z., Kuich, W.: A semiring-semimodule generalization of ω-regular languages I and II. J. of Automata, Languages and Combinatorics 10(2/3), 203–242, 243–264 (2005)
12. Fainekos, G.E.: An introduction to multi-valued model checking. Technical Report MS-CIS-05-16, Dept. of CIS, University of Pennsylvania (September 2005)
13. Fischer, D., Grädel, E., Kaiser, Ł.: Model checking games for the quantitative μ-calculus. In: Proceedings of STACS 2008, pp. 301–312 (2008)
14. Grätzer, G.: General Lattice Theory, 2nd edn. Birkhäuser, Basel (2003)
15. Gurfinkel, A., Chechik, M.: Multi-valued model checking via classical model checking. In: Amadio, R., Lugiez, D. (eds.) CONCUR 2003. LNCS, vol. 2761, pp. 266–280. Springer, Heidelberg (2003)
16. Kozen, D.: Results on the propositional mu-calculus. Theoretical Computer Science 27, 333–354 (1983)
17. Kuich, W., Salomaa, A.: Semirings, Automata, Languages. EATCS Monographs on Theoret. Comp. Sc, vol. 5. Springer, Heidelberg (1986)
18. Kupferman, O., Lustig, Y.: Lattice automata. In: Cook, B., Podelski, A. (eds.) VMCAI 2007. LNCS, vol. 4349, pp. 199–213. Springer, Heidelberg (2007)
19. Park, D.: Concurrency and automata on infinite sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
20. Schützenberger, M.: On the definition of a family of automata. Information and Control 4, 245–270 (1961)

Branching-Time Temporal Logics with Minimal Model Quantifiers*

Fabio Mogavero and Aniello Murano

Universitá degli Studi di Napoli "Federico II", 80126 Napoli, Italy
{mogavero,murano}@na.infn.it

Abstract. Temporal logics are a well investigated formalism for the specification and verification of reactive systems. Using formal verification techniques, we can ensure the correctness of a system with respect to its desired behavior (specification), by verifying whether a model of the system satisfies a temporal logic formula modeling the specification.

From a practical point of view, a very challenging issue in using temporal logic in formal verification is to come out with techniques that automatically allow to select small critical parts of the system to be successively verified. Another challenging issue is to extend the expressiveness of classical temporal logics, in order to model more complex specifications.

In this paper, we address both issues by extending the classical branching-time temporal logic CTL* with minimal model quantifiers (MCTL*). These quantifiers allow to extract, from a model, minimal submodels on which we check the specification (also given by an MCTL* formula). We show that MCTL* is strictly more expressive than CTL*. Nevertheless, we prove that the model checking problem for MCTL* remains decidable and in particular in PSPACE. Moreover, differently from CTL*, we show that MCTL* does not have the tree model property, is not bisimulation-invariant and is sensible to unwinding. As far as the satisfiability concerns, we prove that MCTL* is highly undecidable. We further investigate the model checking and satisfiability problems for MCTL* sublogics, such as MPML, MCTL, and MCTL$^+$, for which we obtain interesting results. Among the others, we show that MPML retains the finite model property and the decidability of the satisfiability problem.

1 Introduction

Temporal logics, which are a special kind of *modal logics* geared towards the description of the temporal ordering of events [Pnu77], have been adopted as a powerful tool for specifying and verifying correctness of concurrent systems [Pnu81], as they allow to express the temporal ongoing behavior of a system in a well-structured way.

Two possible views regarding the nature of time induce two different types of temporal logics: *linear* and *branching-time* [Lam80]. In linear-time temporal logics, such as LTL [Pnu77], time is treated as if each moment in time has a unique possible future. Thus, linear temporal logic formulas are interpreted over linear sequences. In branching-time temporal logics, such as CTL [CE81], CTL$^+$, and CTL* [EH85], each

* Work partially supported by MIUR PRIN Project no.2007-9E5KM8.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 396–409, 2009.

moment in time may split into various possible futures. Accordingly, the structures over which branching temporal logic formulas are interpreted are infinite trees. Many important parallel computer programs exhibit ongoing behavior that is characterized naturally in terms of infinite execution traces, possibly organized into tree-like structures that reflect the high degree of nondeterminism inherent in parallel computation.

In formal system design, one of the most significant developments has been the discovery of algorithmic methods for verifying temporal-logic properties of finite-state systems [CE81, QS82]. In temporal-logic model checking, we verify the correctness of a finite-state system with respect to a desired behavior by checking whether a labeled state-transition graph, called *Kripke structure*, that models the system satisfies a temporal logic formula that specifies this behavior. Hence, the name *model checking* for the verification method derived from this viewpoint. Since model checking has many practical applications (see [Eme90] for more motivations and background) it is important to classify temporal logics according to the computational complexity of their model checking problem. Indeed, the complexity for branching-time temporal logics is well understood: for CTL, CTL$^+$, and CTL* it is PTIME-COMPLETE, Δ_2^p-COMPLETE, and PSPACE-COMPLETE, respectively.

From a practical point of view, a very challenging issue in using temporal logics in formal specification and verification is to come out with automatic techniques that allow to select small critical parts of the system in order to restrict system verification to them. This necessity is mainly due to the fact that in a concurrent setting, the system under consideration is typically a parallel composition of many modules. Promising approaches to restrict the verification techniques to subsystems of interest are assume guarantee techniques [AL93], modular model checking [KV95, KV97], the exploitation of partial order information [Pel96], localization reduction [Kur94], and semantic minimization for eliminate unnecessary states from a system model [ECJB97]. Note that all these approaches have in common the fact that the modularity of the system is known in advance. Another important issue in system design and verification is to look for new temporal logics that are more expressive than the classical ones. In fact, although CTL* is a very powerful logic, there are several important but complex properties that require a more powerful framework. To overcome this limitation, several attempts have been carried out in literature in order to extend these logics by introducing appropriate semantics or operators usually guided by embedded contexts [AHK02, BLMV06, BMM09].

In this paper, we address both the above issues by introducing the branching-time temporal logic MCTL*. This logic is an extension of the classical branching-time temporal logic CTL* with minimal model quantifiers, which allow to extract, given a model, minimal and conservative submodels of it on which we successively check a given property. The goal is to check local properties of system components in order to deduce the global behavior of the entire one. Therefore, the introduced logic exploits the novel idea of checking a particular module of a whole composition system while its single modules are not known in advance. In more details, MCTL* extends CTL* by also allowing two special (*minimal model*) quantifiers: Λ and Ξ. These quantifiers allow to write state formulas such as $\varphi_1 \Lambda \varphi_2$ and $\varphi_1 \Xi \varphi_2$, which respectively read as *"all minimal and conservative models of* φ_2 *are models of* φ_1*"* and *"there exists a minimal model of* φ_2

that is model of φ_1*"*, for suitable and well-found concepts of minimality and conservativeness among Kripke structures. In accordance with this point of view, we call φ_2 the *submodel extractor*, φ_1 the *submodel verifier*, and our modular verification method an *extract-verify* paradigm. Our choice of considering only minimal and conservative submodels is justified by the fact that in this way we precisely select the parts of the system that are actually responsible for the particular behavior of interest. For an example of an application of the introduced logic see Example 1 in Section 3. It is worth recalling that logics having the ability to modify the model under evaluation (and then check the specification on the resulting part) have been also considered in other contexts. For example, we recall the arbitrary public announcement logic [FvD08] and the sabotage modal logic [LR03]. However, the first allow to extract, according to a submodel extractor formula, submodels that do not necessarily satisfy the formula itself, and the second does not extract submodels using a formula at all.

In this paper, we investigate MCTL* and its sublogics MCTL$^+$, MCTL and MPML (where M indicates the extension of the respective logics with minimal model quantifiers) from a theoretical point of view. As far as the expressivity regards, we show that all these logics are strictly more expressive than the corresponding classical ones. Unfortunately, this power comes at a price. Indeed, we show that the satisfiability for MCTL is highly undecidable. Moreover and differently from CTL, we have that introduced logics neither have the tree model property nor are bisimulation-invariant, while they all are sensible to unwinding. We also investigate succinctness and the model checking problem for the introduced logics, from which we got interesting results. Among the others, we show that MCTL is as succinct as MCTL$^+$(differently from the classical case of CTL and CTL$^+$). Moreover, as CTL$^+$ [LMS01], both MCTL and MCTL$^+$ have a Δ_2^p-COMPLETE (i.e., PTIME$^{\text{NPTIME}}$) model checking. As far as we know, our result provides the second example, after CTL$^+$, of Δ_2^p-COMPLETE problems in the field of formal verification. Since for this class very few complete problems are known, we believe that the obtained result is interesting as its own. Finally, we show that the propositional modal logic (PML) augmented with minimal model quantifiers (MPML) retains both the finite model property and the decidability of the satisfiability problem.

2 Preliminaries

Given a *set* X of *objects* (numbers, words, etc.), we denote by $|X|$ its cardinality, called *size* of X, and by 2^X the *powerset* of X. As special sets, we consider $\mathbb{Z}$, $\mathbb{N}$, and $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$, as respectively, the sets of *relative*, *natural*, and *positive natural numbers*.

A *Kripke structure* $\mathcal{K} = \langle AP, W, R, L\rangle$ is an ordered tuple, where AP is a set of *atomic propositions*, $W = \mathsf{dom}(\mathcal{K})$ is a non-empty set of *worlds*, $R \subseteq W \times W$ is a *binary relation*, and $L : W \mapsto 2^{AP}$ is the labeling function that maps each world to a set of atomic propositions true in that world. We denote the size $|\mathcal{K}|$ of $\mathcal{K}$ by $|W| + |R|$. An infinite Kripke structure is a structure of infinite size. Now, let $\mathcal{K}' = \langle AP', W', R', L'\rangle$ be another Kripke structure. We say that $\mathcal{K}'$ is a *substructure* of $\mathcal{K}$, in symbols $\mathcal{K}' \preccurlyeq \mathcal{K}$, iff *(i)* $AP' \subseteq AP$, *(ii)* $W' \subseteq W$, *(iii)* $R' \subseteq R \cap (W' \times W')$, and *(iv)* for all $w \in W'$, it holds that $L'(w) = L(w) \cap AP'$. Moreover, we say that $\mathcal{K}$ and $\mathcal{K}'$ are *comparable* iff *(i)* $\mathcal{K} \preccurlyeq \mathcal{K}'$ or *(ii)* $\mathcal{K}' \preccurlyeq \mathcal{K}$ holds, otherwise they are *incomparable*. For a set of structures $\mathfrak{S}$, we define the set of *minimal substructures* (antichain) $\mathsf{minstructs}(\mathfrak{S})$ as the set consisting of the $\preccurlyeq$-minimal elements of $\mathfrak{S}$. I.e., it is the set containing all and only the structures $\mathcal{K} \in \mathfrak{S}$ such that for all $\mathcal{K}' \in \mathfrak{S}$, it holds that *(i)* $\mathcal{K} \preccurlyeq \mathcal{K}'$, or *(ii)* $\mathcal{K}'$ is not comparable with $\mathcal{K}$. Note that all structures in $\mathsf{minstructs}(\mathfrak{S})$ are incomparable among them. A structure $\mathcal{K}$ is *minimal* w.r.t. a set $\mathfrak{S}$ (or simply minimal, when the context clarify the set $\mathfrak{S}$) iff $\mathcal{K} \in \mathsf{minstructs}(\mathfrak{S})$. A set of structures $\mathfrak{S}$ is minimal iff $\mathfrak{S} = \mathsf{minstructs}(\mathfrak{S})$.

For sake of space, all other classical concepts of *tree*, *path*, *set of maximal paths* $\mathsf{paths}(\mathcal{K}, w)$ of a structure $\mathcal{K}$ starting in a world $w \in \mathsf{dom}(\mathcal{K})$, and unwinding $\mathcal{U}^{\mathcal{K}}_{w}$ of $\mathcal{K}$ in w, are omitted (see [KVW00] for detailed definitions).

3 The Minimal Model Quantifiers Temporal Logic Extensions

In this section, we introduce an extension of the classical branching-time temporal logic CTL* with minimal model quantifiers, which allow to extract minimal submodels on which we successively check a given property. To formally define the extended logic, we use the CTL* state and path formulas framework.

The *full computation tree logic with minimal model quantifiers* (MCTL*, for short) extends CTL* by further using two special quantifiers, the universal Λ and the existential Ξ ones. Informally, a model satisfies a state formula $\varphi_1 \Lambda \varphi_2$ iff all its minimal and conservative submodels satisfying φ_2 (φ_2 is the *submodel extractor*) are also models satisfying φ_1 (φ_1 is the *submodel verifier*). As in CTL*, in MCTL* the two path quantifiers A and E can prefix a linear time formula composed by an arbitrary combination and nesting of the four linear temporal operators X (*"effective next"*), $\tilde{\mathsf{X}}$ (*"hypothetical next"*), U (*"until"*), and R (*"release"*). The formal syntax of MCTL* follows.

Definition 1. (Syntax) MCTL* state *(*φ*) and* path *(*ψ*)* formulas *are built inductively from* AP *using the following context-free grammar, where* $p \in \mathrm{AP}$*:*

1. $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \Lambda \varphi \mid \varphi \Xi \varphi \mid \mathsf{A}\psi \mid \mathsf{E}\psi,$
2. $\psi ::= \varphi \mid \neg\psi \mid \psi \wedge \psi \mid \psi \vee \psi \mid \mathsf{X}\psi \mid \tilde{\mathsf{X}}\psi \mid \psi \mathsf{U} \psi \mid \psi \mathsf{R} \psi.$

The class of MCTL* *formulas is the set of state formulas generated by the above grammar. In addition, the simpler classes of* MCTL$^+$*,* MCTL*, and* MPML *formulas are obtained, respectively, by avoiding nesting of temporal operators, by forcing each temporal operator occurring into a formula to be coupled with a path quantifier, and by excluding from* MCTL *path formulas the until and release operators.* □

The *length* $|\varphi|$ of a formula φ is defined inductively on the structure of φ in the classical way, and by also considering $|\varphi_1 \Lambda \varphi_2|$ and $|\varphi_1 \Xi \varphi_2|$ to be equal to $1 + |\varphi_1| + |\varphi_2|$.

We now define the semantics of MCTL* w.r.t. a Kripke structure $\mathcal{K}$. For a world $w \in \mathsf{dom}(\mathcal{K})$, we write $\mathcal{K}, w \models \varphi$ to indicate that a state formula φ holds at w, and, for a path $\pi \in \mathsf{paths}(\mathcal{K})$, we write $\mathcal{K}, \pi, k \models \psi$ to indicate that a path formula ψ holds on π at position $0 \leq k < |\pi|$. Note that, the relation $\mathcal{K}, \pi, k \models \psi$ does not hold for any point $k \in \mathbb{N}$, with $k \geq |\pi|$. The semantics of state and path formulas involving $\neg$, $\wedge$, and $\vee$, the classical path quantifiers E and A, and the classical temporal operators is defined as usual in CTL*. Here we only give the semantics of the remaining part.

Definition 2. (Semantics) *Given a Kripke structure* $\mathcal{K} = \langle \mathrm{AP}, \mathrm{W}, \mathrm{R}, \mathrm{L} \rangle$*, a world* $w \in \mathrm{W}$*, and two* MCTL* *state formulas* φ_1 *and* φ_2 *it holds:*

1. $\mathcal{K}, w \models \varphi_1 \Lambda \varphi_2$ *iff for all* $\mathcal{K}' \in \mathsf{minstructs}(\mathfrak{S}(\mathcal{K}, w, \varphi_2))$ *it holds that* $\mathcal{K}', w \models \varphi_1$*;*
2. $\mathcal{K}, w \models \varphi_1 \Xi \varphi_2$ *iff there is* $\mathcal{K}' \in \mathsf{minstructs}(\mathfrak{S}(\mathcal{K}, w, \varphi_2))$ *such that* $\mathcal{K}', w \models \varphi_1$*;*

where $\mathfrak{S}(\mathcal{K}, w, \varphi) = \{\mathcal{K}' \preccurlyeq \mathcal{K} \mid w \in \mathsf{dom}(\mathcal{K}') \wedge \forall \mathcal{K}'' \preccurlyeq \mathcal{K} : \mathcal{K}' \preccurlyeq \mathcal{K}'' \rightarrow \mathcal{K}'', w \models \varphi\}$

It is clear that, MCTL* (resp., MPML, MCTL, and MCTL$^+$) formulas without minimal model quantifiers are CTL* (resp., PML, CTL, and CTL$^+$) formulas.

Let $\mathcal{K}$ be a Kripke structure and φ a MCTL* formula. Then, $\mathcal{K}$ is a *model* for φ, denoting this by $\mathcal{K} \models \varphi$, iff there is $w \in \mathsf{dom}(\mathcal{K})$ such that $\mathcal{K}, w \models \varphi$. In this case, we also say that $\mathcal{K}$ is a model for φ on w. A MCTL* formula φ is said *satisfiable* iff there exists a model for it, moreover it is *invariant* on two Kripke structures $\mathcal{K}$ and $\mathcal{K}'$ iff either $\mathcal{K} \models \varphi$ and $\mathcal{K}' \models \varphi$ or $\mathcal{K} \not\models \varphi$ and $\mathcal{K}' \not\models \varphi$, i.e., $\mathcal{K}$ and $\mathcal{K}'$ agree on φ.

A Kripke structure $\mathcal{K}$ is *conservative* w.r.t. a formula φ iff, for all models $\mathcal{K}'$ extending $\mathcal{K}$, i.e., with $\mathcal{K} \preccurlyeq \mathcal{K}'$, it holds that $\mathcal{K}' \models \varphi$. Note that this concept of conservativeness is automatically embedded in the definition of $\mathfrak{S}(\mathcal{K}, w, \varphi)$, since we consider only models $\mathcal{K}' \in \mathfrak{S}(\mathcal{K}, w, \varphi)$ that, if extended, continue to satisfy the formula φ. To better understanding the meaning and the importance of the conservativeness, consider the Kripke structure $\mathcal{K}$ built by a chain of three states $w_0 \rightarrow w_1 \rightarrow w_2$, in which the final state w_2 is the only one labeled by p. Moreover, consider the two submodels $\mathcal{K}'$ and $\mathcal{K}''$ of $\mathcal{K}$ built, respectively, by w_0 and $w_0 \rightarrow w_1$. Clearly $\mathcal{K}' \preccurlyeq \mathcal{K}'' \preccurlyeq \mathcal{K}$ and, for $\varphi = \mathsf{E}\tilde{\mathsf{X}}\,\mathsf{F}\,p$, we have that $\mathcal{K}' \models \varphi$, $\mathcal{K}'' \not\models \varphi$, and $\mathcal{K} \models \varphi$. Hence, we have that $\mathcal{K}'$ satisfies φ, but it is not conservative, since $\mathcal{K}''$ (that extend $\mathcal{K}'$) does not satisfy φ.

For all state formulas φ_1 and φ_2 (resp., path formulas ψ_1 and ψ_2), we say that φ_1 is *equivalent* to φ_2, formally $\varphi_1 \equiv \varphi_2$, (resp., ψ_1 is *equivalent* to ψ_2, formally $\psi_1 \equiv \psi_2$) iff for all Kripke structures $\mathcal{K}$ and worlds $w \in \mathsf{dom}(\mathcal{K})$, it holds that $\mathcal{K}, w \models \varphi_1$ iff $\mathcal{K}, w \models \varphi_2$ (resp., for all paths $\pi \in \mathsf{paths}(\mathcal{K}, w)$, it holds that $\mathcal{K}, \pi, 0 \models \psi_1$ iff $\mathcal{K}, \pi, 0 \models \psi_2$).

In the rest of the paper, we mainly consider formulas in *existential normal form* or in *positive normal form*, i.e., formulas in which only existential (minimal model and path) quantifiers occur or negation is applied only to atomic propositions, respectively. In fact, it is to this aim that we have considered in the syntax of MCTL* both the connectives $\wedge$ and $\vee$, the quantifiers Λ, Ξ, A and E, and the dual operators $\tilde{\mathsf{X}}$ and R. Indeed, all formulas can be converted in existential or positive normal form by using De Morgan's laws and the following equivalences, which directly follow from the semantics of the logic. Let φ_1 and φ_2 be state formulas and ψ, ψ_1, and ψ_2 be path formulas, then it holds that $\neg(\varphi_1 \Lambda \varphi_2) \equiv \neg\varphi_1 \Xi \varphi_2$, $\neg\mathsf{A}\psi \equiv \mathsf{E}\neg\psi$, $\neg\mathsf{X}\,\psi \equiv \tilde{\mathsf{X}}\,\neg\psi$, and $\neg(\psi_1 \mathsf{U}\, \psi_2) \equiv \neg\psi_1 \mathsf{R}\, \neg\psi_2$. In order to abbreviate writing formulas, we also use the boolean values $\mathfrak{t}$ (*"true"*) and $\mathfrak{f}$ (*"false"*) and the path temporal operators $\mathsf{F}\,\psi \equiv \mathfrak{t}\,\mathsf{U}\,\psi$ (*"future"*) and $\mathsf{G}\,\psi \equiv \mathfrak{f}\,\mathsf{R}\,\psi$ (*"globally"*). Moreover, note that the following equivalences also hold: $\mathsf{E}(\psi_1 \vee \psi_2) \equiv \mathsf{E}\psi_1 \vee \mathsf{E}\psi_2$, $\tilde{\mathsf{X}}\,\psi \equiv \tilde{\mathsf{X}}\,\mathfrak{f} \vee \mathsf{X}\,\psi$, $\psi_1 \mathsf{R}\,\psi_2 \equiv (\psi_2 \mathsf{U}\,(\psi_1 \wedge \psi_2)) \vee \mathsf{G}\,\psi_2$, $\mathsf{X}\,(\psi_1 \wedge \psi_2) \equiv \mathsf{X}\,\psi_1 \wedge \mathsf{X}\,\psi_2$, and $\mathsf{G}\,(\psi_1 \wedge \psi_2) \equiv \mathsf{G}\,\psi_1 \wedge \mathsf{G}\,\psi_2$.

Example 1. ***(Arbiter system)*** Consider an arbiter system used to control a two-users access to a shared memory location (see Figure 1 for a model $\mathcal{K}$ of it), where only the resquest (`r`) and the acknowledge (`a`) signals are known. Suppose now that we want to verify that the idle state i and the common request state $\langle r_1, r_2 \rangle$ are unique w.r.t. the order of user request or arbiter acknowledge. We can perform this check by applying MCTL* model checking in the state i using a formula $\varphi = \varphi_1 \wedge \varphi_2$, where $\varphi_1 = \mathsf{AG}\,(i \rightarrow$

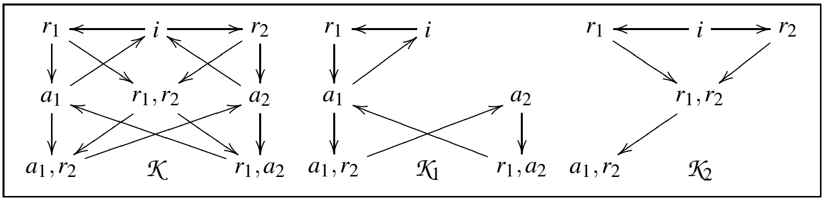


Fig. 1. A model of an arbiter system for shared memory locations and its submodels

$\mathsf{X}\,\mathsf{t}) \wedge \mathsf{E}(\mathsf{F}\,(a_1 \wedge \mathsf{X}\,\mathsf{F}\,i) \wedge \mathsf{F}\,(a_2 \wedge \mathsf{X}\,\mathsf{F}\,i))$ checks if the "acknowledge subsystem" reaches the same idle state and $\varphi_2 = \mathsf{AG}\,(r_1 \wedge r_2 \rightarrow \mathsf{X}\,\mathsf{t}) \wedge (\mathsf{EX}\,(r_1 \wedge \mathsf{X}\,(r_2 \wedge \mathsf{X}\,\mathsf{t})) \wedge \mathsf{EX}\,(r_2 \wedge \mathsf{X}\,r_1))$ checks if the common request state reached by the "request subsystem" is unique. For two minimal and conservative submodels of φ_1 and φ_2 in $\mathcal{K}$ see $\mathcal{K}_1$ and $\mathcal{K}_2$ in Figure 1. Note that also their "mirror images" are submodels of φ_1 and φ_2.

One may note that the above check can not be achieved using a classical logic such as CTL*. Indeed, we may have a bisimilar model of $\mathcal{K}$, with more idle or common request states, in which no CTL* formula can check that these states are not unique. □

By means of counterexamples, we show that the introduced extended logics are more expressive than the corresponding classical ones. Indeed, since they can distinguish among models that are invariant for the classical logics, as described in the previous example. The result is reported in the following theorem.

Theorem 1. *For* MPML, MCTL, MCTL$^+$, *and* MCTL* *it holds that they* (i) *do not have the tree model property;* (ii) *are neither invariant under unwinding nor under partial unwinding;* (iii) *are not invariant under bisimulation; and* (iv) *are more expressive than* PML, CTL, CTL$^+$, *and* CTL*, *respectively.*

Proof. Item (i) To prove this item, we consider a formula with an existential minimal model quantifier such that it requires to extract a graph submodel, which can not be a tree, in order to be satisfied. Consider the MPML formula $\varphi = \varphi_1 \,\Xi\, \varphi_2$, where $\varphi_1 = \mathsf{EX}(\beta \wedge \mathsf{EX}\,\mathsf{EX}\,\gamma)$, $\varphi_2 = \alpha \wedge \mathsf{EX}(\beta \wedge \mathsf{EX}\,\delta) \wedge \mathsf{EX}(\gamma \wedge \mathsf{EX}(\delta \wedge \mathsf{EX}\,\gamma))$, $\alpha = a \wedge b$, $\beta = \neg a \wedge b$, $\gamma = a \wedge \neg b$, and $\delta = \neg a \wedge \neg b$. This formula is satisfiable. In Figure 2 we show $\mathcal{K}_1$, $\mathcal{K}_2$, $\mathcal{K}_3$, and $\mathcal{K}_4$ as the only minimal models of φ_2, where only $\mathcal{K}_1$ is a tree and $\mathcal{K}_3$ and $\mathcal{K}_4$ are the only models of φ. Indeed, only $\mathcal{K}_3$ and $\mathcal{K}_4$ satisfy φ_1. Since any model of φ must include $\mathcal{K}_3$ or $\mathcal{K}_4$ as submodel, it follows that no tree model can satisfy φ. Since MPML is a sublogic of MCTL, MCTL$^+$, and MCTL* the thesis easily follows.

Item (ii) Suppose by contradiction that MPML is invariant under unwinding. Then, for each satisfiable formula φ, since there exists a model $\mathcal{K}$ such that $\mathcal{K}, w \models \varphi$, it holds that $\mathcal{U}^{\mathcal{K}}_{w}, w \models \varphi$. But $\mathcal{U}^{\mathcal{K}}_{w}$ is a tree, so each satisfiable formula has a tree model, but this contradicts the previous item. To prove that this logic is also not invariant under partial unwinding, consider the model $\mathcal{K}$, built by a single world w with a loop relation on it, and its "one step" unwinding $\mathcal{K}'$, formed by two worlds, w and v, linked together by a relation and with a relation loop on the second one (see Figure 2). Moreover, consider, the MPML formula $\varphi = (\mathsf{EX}\,\mathsf{EX}\,\mathsf{t}) \,\Xi\, (\mathsf{EX}\,\mathsf{t})$. It holds that $\mathfrak{S}(\mathcal{K}, w, \mathsf{EX}\,\mathsf{t}) = \{\mathcal{K}\}$, while $\mathfrak{S}(\mathcal{K}', w, \mathsf{EX}\,\mathsf{t}) = \{\mathcal{K}', \mathcal{K}''\}$, with $\mathcal{K}'' \preccurlyeq \mathcal{K}'$ and where $\mathcal{K}''$ is equal to $\mathcal{K}'$ once the loop on the last node is removed. It is easy to verify that $\mathcal{K} \models \mathsf{EX}\,\mathsf{EX}\,\mathsf{t}$, but $\mathcal{K}'' \not\models \mathsf{EX}\,\mathsf{EX}\,\mathsf{t}$,

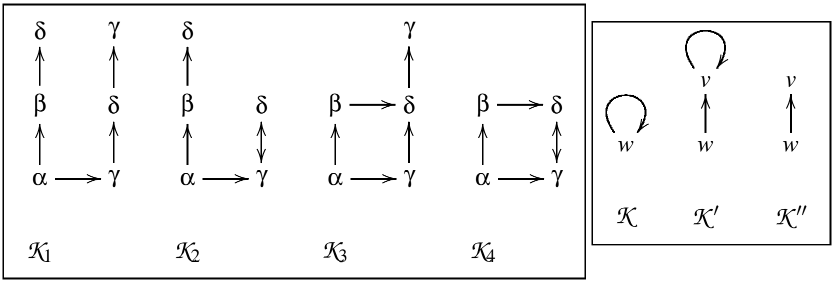

δ γ δ γ
β δ β δ β δ β δ
α γ α γ α γ α γ
$\mathcal{K}_1$ $\mathcal{K}_2$ $\mathcal{K}_3$ $\mathcal{K}_4$
v v
w w w
$\mathcal{K}$ $\mathcal{K}'$ $\mathcal{K}''$

Fig. 2. Minimal models of φ_2 in item *i* and models of $\mathsf{EX}\,\mathsf{t}$ in item *ii* of Theorem 1

so we have that only $\mathcal{K}$ is a model of φ. This shows that MPML is able to distinguish between a model and one of its partial unwindings, thus the thesis follows.

Item (iii) Since an unwinding is a particular case of a bisimilarity relation, we have also that MPML is not bisimilar, i.e., it is possible to express a MPML property satisfied on a model $\mathcal{K}$, but not on a bisimilar model $\mathcal{K}'$ of $\mathcal{K}$.

Item (iv) This item follows from the fact that PML, CTL, CTL$^+$, and CTL* are invariant under bisimulation, while their extensions with minimal model quantifiers are not. Therefore, the extended logics can characterize more models than the classical ones and thus they are more expressive. □

We now show that MPML has the strong finite model property. To this aim, we first introduce some extra notations. For a Kripke structure $\mathcal{K}$, by $\mathsf{dep}(\mathcal{K})$ we denote the maximal length of a path in the unwinding $\mathcal{U}_{\mathcal{K}}^{w}$, for all worlds $w \in \mathsf{dom}(\mathcal{K})$. Moreover, given a MPML formula φ, we denote by $\mathsf{dep}(\varphi)$ the *depth* of φ, i.e., the maximal number of nested occurrences of path quantifiers in φ, but those appearing in its submodel verifiers. Formally, the depth function is inductively defined as follows: $\mathsf{dep}(p) = 0$, for $p \in \mathrm{AP}$; $\mathsf{dep}(\neg\varphi) = \mathsf{dep}(\varphi)$; $\mathsf{dep}(\varphi_1 \wedge \varphi_2) = \mathsf{dep}(\varphi_1 \vee \varphi_2) = \max\{\mathsf{dep}(\varphi_1), \mathsf{dep}(\varphi_2)\}$; $\mathsf{dep}(\varphi_1 \,\Lambda\, \varphi_2) = \mathsf{dep}(\varphi_1 \,\Xi\, \varphi_2) = \mathsf{dep}(\varphi_2)$; $\mathsf{dep}(\mathsf{A}\psi) = \mathsf{dep}(\mathsf{E}\psi) = 1 + \mathsf{dep}(\varphi)$, where $\psi = \mathsf{X}\,\varphi$ or $\psi = \tilde{\mathsf{X}}\,\varphi$. It is easy to see that $\mathsf{dep}(\varphi) = O(|\varphi|)$.

Theorem 2. MPML *has the strong finite model property, i.e., each* MPML *satisfiable formula* φ *has a finite model* $\mathcal{K}$ *with size* $|\mathcal{K}| \leq g(|\varphi|)$, *where g is a recursive function, and depth* $\mathsf{dep}(\mathcal{K}) \leq \mathsf{dep}(\varphi)$.

In [EH85] it is shown that CTL$^+$ is equivalent to CTL by using an exponential blow-up translation. Also, in [Wil99] it is shown that this blow-up is unavoidable. In the next theorem, we show that MCTL and MCTL$^+$ are polynomially equivalent and then, as an immediate corollary, we obtain that MCTL is exponentially more succinct than CTL.

Theorem 3. MCTL *is polynomially equivalent to* MCTL$^+$.

Proof. (Sketch.) Given a MCTL$^+$ formula φ we show that there exists a MCTL formula φ' equivalent to φ such that $|\varphi'| = O(|\varphi|^3)$.

W.l.o.g we assume that φ is in existential normal form (we recall that any MCTL$^+$ formula can be linearly translated in this form). Moreover, due to classical formula equivalences [EH85], we can also assume that φ has one E quantifier by recursively applying the translation algorithm to nested subformulas containing an E. So we can assume that φ is of the form $\mathsf{E}\psi$ and ψ is a Boolean combination of subformulas of the form $\varphi'_i \mathsf{U}\, \varphi''_i$, $\mathsf{G}\,\varphi_1$, $\mathsf{X}\,\varphi_2$, and $\tilde{\mathsf{X}}\,\mathfrak{f}$, where each φ'_i, φ''_i, φ_1 and φ_2 are MCTL formulas (found by recursive applications of the translation algorithm). In practice, this turns out to use, as base case of the translation idea, the four equivalences listed below:

i) $\mathsf{E}(\bigwedge_{i=1}^{n} \varphi'_i \mathsf{U}\, \varphi''_i \wedge \mathsf{G}\,\varphi_1 \wedge \tilde{\mathsf{X}}\,\mathfrak{f}) \equiv \bigwedge_{i=1}^{n} \varphi''_i \wedge \varphi_1 \wedge \mathsf{E}\tilde{\mathsf{X}}\,\mathfrak{f}$;
ii) $\mathsf{E}(\mathsf{G}\,\varphi_1 \wedge \mathsf{X}\,\varphi_2) \equiv \varphi_1 \wedge \mathsf{EX}\,(\varphi_2 \wedge \mathsf{EG}\,\varphi_1)$;
iii) $\mathsf{E}(\bigwedge_{i=1}^{n} \varphi'_i \mathsf{U}\, \varphi''_i \wedge \mathsf{G}\,\varphi_1) \equiv \bigvee_{i=1}^{n} (f'_i \,\Xi\, f''_i)$;
iv) $\mathsf{E}(\bigwedge_{i=1}^{n} \varphi'_i \mathsf{U}\, \varphi''_i \wedge \mathsf{G}\,\varphi_1 \wedge \mathsf{X}\,\varphi_2) \equiv \bigwedge_{i=1}^{n} \varphi''_i \wedge \varphi_1 \wedge \mathsf{EX}\,(\varphi_2 \wedge \mathsf{EG}\,\varphi_1) \vee \bigvee_{i=1}^{n} (f'_i \,\Xi\, f''_i)$;
where $f'_i = \bigwedge_{1 \leq h < k \leq n}^{h,k \neq i} (\mathsf{EF}\,(\varphi''_h \wedge \mathsf{EF}\,\varphi''_k) \vee \mathsf{EF}\,(\varphi''_k \wedge \mathsf{EF}\,\varphi''_h))$ and $f''_i = \mathsf{E}((\varphi'_i \wedge \varphi_1)\mathsf{U}\,(\varphi''_i \wedge \mathsf{EG}\,\varphi_1)) \wedge \bigwedge_{j=1;j\neq i}^{n} \mathsf{E}(\varphi'_j \mathsf{U}\,(\varphi''_j \wedge \mathsf{EF}\,\varphi''_i))$.

The first two equivalences, which do not contain the minimal model quantifier Ξ, are derivable by simply applying classical transformations. The proof of the last two, instead, can be obtained by formally showing (by induction) that each model satisfying the first member of an equivalence must satisfy also the second one and vice versa. Here, we omit this part for the sake of space, while we give an intuition of the third equivalence, which shows, as in the fourth one, how to avoid the exponential blow-up incurred by the classical translation in CTL for the corresponding case.

The key step in the translation is the selection of the right submodel of the extractor formula f''_i, through the verifier formula f'_i, which must satisfy $\varphi = \mathsf{E}(\bigwedge_{i=1}^{n} \varphi'_i \mathsf{U}\, \varphi''_i \wedge \mathsf{G}\,\varphi_1)$. If a model $\mathcal{K}$ satisfies φ in a world $w \in \mathsf{dom}(\mathcal{K})$, for all $\mathcal{K}' \in \mathsf{minstructs}(\mathfrak{S}(\mathcal{K}, w, \varphi))$ it holds that $\mathcal{K}' \in \mathsf{minstructs}(\mathfrak{S}(\mathcal{K}, w, f''_i))$, for a given index i. Moreover, for all paths $\pi \in \mathsf{paths}(\mathcal{K}', w)$ such that $\mathcal{K}', \pi, 0 \models \bigwedge_{i=1}^{n} \varphi'_i \mathsf{U}\, \varphi''_i$, we have that $\mathcal{K}', \pi, 0 \models \mathsf{F}\,(\varphi''_h \wedge \mathsf{F}\,\varphi''_k)$ or $\mathcal{K}', \pi, 0 \models \mathsf{F}\,(\varphi''_k \wedge \mathsf{F}\,\varphi''_h)$, for all indexes h and k, with $h < k$ and $h, k \neq i$. Hence, it holds that $\mathcal{K}', w \models f'_i$ and then $\mathcal{K}, w \models f'_i \,\Xi\, f''_i$.

Vice versa, consider a model $\mathcal{K}$ such that $\mathcal{K}, w \models f'_i \,\Xi\, f''_i$, for a given index i. Then, it holds that there exists a minimal model $\mathcal{K}' \in \mathsf{minstructs}(\mathfrak{S}(\mathcal{K}, w, f''_i))$ such that $\mathcal{K}', w \models f'_i$. Now, suppose by contradiction that $\mathcal{K}', w \not\models \varphi$. Then there exist at least three different and not directly connected parts $\mathcal{K}'_1$, $\mathcal{K}'_2$, and $\mathcal{K}'_3$, with $\mathcal{K}'_1, \mathcal{K}'_2, \mathcal{K}'_3 \preccurlyeq \mathcal{K}'$, and three paths $\pi_1 \in \mathsf{paths}(\mathcal{K}'_1, w)$, $\pi_2 \in \mathsf{paths}(\mathcal{K}'_2, w)$, and $\pi_3 \in \mathsf{paths}(\mathcal{K}'_3, w)$, such that each path formula $\varphi'_j \mathsf{U}\, \varphi''_j$, with $j \neq i$, is satisfied on just two of these paths. Then, each formula $\mathsf{E}(\varphi'_j \mathsf{U}\,(\varphi''_j \wedge \mathsf{EF}\,\varphi''_i))$ is satisfied in at least two ways in two different submodels of $\mathcal{K}'$ and then there exists a submodel $\mathcal{K}'' \preccurlyeq \mathcal{K}'$, $\mathcal{K}'' \neq \mathcal{K}'$ such that $\mathcal{K}'', w \models f''_i$. So, $\mathcal{K}'$ is not minimal, but this contradicts the assumption. Hence, $\mathcal{K}', w \models \varphi$.

Finally, note that it is fundamental that minimal model quantifiers are conservative. Otherwise, we could have a model $\mathcal{K}$ such that $\mathcal{K}, w \models \mathsf{t} \,\Xi\, \mathsf{EG}\,\varphi_1$ even if $\mathcal{K}, w \not\models \mathsf{EG}\,\varphi_1$. This means that in the above discussion, we could have $\mathcal{K}, w \not\models \varphi$, since there are no paths satisfying $\mathsf{G}\,\varphi_1$, but $\mathcal{K}, w \models f'_i \,\Xi\, f''_i$, for some i. □

Corollary 1. MCTL *is exponentially more succinct than* CTL.

4 Model Checking

In this section, we solve the model checking for the introduced logics, showing that the considered extract-verify paradigm retains the decidability of this problem.

We start with a lemma that shows how to calculate a polynomial certificate for particular MCTL and MCTL* formulas. This result will be then useful to show the corresponding upper bound results for the addressed model checking problems.

Lemma 1. *Let $\mathcal{K}$ be a Kripke structure, $w \in \mathsf{dom}(\mathcal{K})$ be a world and $\varphi = \varphi_1 \,\Xi\, \varphi_2$ be a* MCTL *(resp.,* MCTL***) formula, with φ_1 and φ_2* CTL *(resp.,* CTL***) formulas. Then, there exists a polynomial certificate $\mathcal{K}'$ of the testing $\mathcal{K}, w \models \varphi$, which is verifiable in* PTIME *(resp.,* PSPACE*).*

Proof. To check that the test $\mathcal{K}, w \models \varphi$ is in NPTIME (resp., in PSPACE), we verify that there exists a minimal and conservative submodel $\mathcal{K}'$ of $\mathcal{K}$ (the certificate of the test) of polynomial size (since $|\mathcal{K}'| \leq |\mathcal{K}|$) satisfying φ_2 in w such that $\mathcal{K}', w \models \varphi_1$. To this aim, we split the verification procedure into the following four phases: *(i)* testing of $\mathcal{K}', w \models \varphi_2$, *(ii)* checking the minimality of $\mathcal{K}'$, *(iii)* checking the conservativeness for $\mathcal{K}'$, and *(iv)* testing of $\mathcal{K}', w \models \varphi_1$. The first and last items are easily achievable in PTIME (resp., in PSPACE) by applying a classical CTL (resp., CTL*) model checking algorithm. Instead, to verify that $\mathcal{K}'$ is minimal w.r.t. the formula φ_2, we check that for all maximal and proper submodels $\mathcal{K}''$ of $\mathcal{K}'$, it holds that $\mathcal{K}'', w \not\models \varphi_2$. Now, note that all models $\mathcal{K}''$ are in number $O(|\mathcal{K}'|)$ since each of them is obtained by removing only one component from $\mathcal{K}'$. So, we deduce that also the check for minimality can be done in PTIME (resp., in PSPACE). Finally, it remains to verify whether $\mathcal{K}'$ is conservative, i.e., for all models $\mathcal{K}''$, with $\mathcal{K}' \preccurlyeq \mathcal{K}''$, it holds $\mathcal{K}'', w \models \varphi_2$. To do this, we can check that, for all subformula φ' of φ_2 and for all worlds $w \in \mathsf{dom}(\mathcal{K}')$, it holds that $\mathcal{K}', w \models \varphi'$ iff $\mathcal{K}, w \models \varphi'$. Since the number of all subformulas φ' is polynomial in the size of φ_2, and thus in the size of φ, it follows that also the check for conservativeness is in PTIME (resp., in PSPACE). To sum up, we have that to verify a certificate for the test $\mathcal{K}, w \models \varphi$ is in PTIME (resp., PSPACE), and therefore we have done with the proof. □

Using the above result, we are now able to prove the following two theorems.

Theorem 4. MCTL* *has a* PSPACE-COMPLETE *model checking problem.*

Proof. For the lower bound, we recall that for CTL*, which is a sublogic of MCTL*, the model checking problem is already PSPACE-HARD. We now proceed with the upper bound. To this aim, let $\mathcal{K}$ be a Kripke structure and φ an MCTL* in existential normal form, we construct a recursive algorithm that checks in PSPACE whether $\mathcal{K}, w \models \varphi$.

First of all, we enumerate all subformulas $\overline{\varphi} = \varphi' \,\Xi\, \varphi''$ of φ and we associate to each of them a fresh different atomic proposition. More formally, suppose that we have a sequence $(\overline{\varphi_1}, \ldots, \overline{\varphi_n})$ of such subformulas, then we associate to each $\overline{\varphi_i}$ the proposition ep_i. Now, consider Θ_m as the set of all formulas $\overline{\varphi} = \varphi' \,\Xi\, \varphi''$ subformulas of φ such that φ' contains just m and φ'' contains at most m nested occurrences of the Ξ quantifier, or vice versa. Also, consider Θ'_m as the set of formulas $\tilde{\varphi}$ obtained from each $\overline{\varphi} \in \Theta_m$ by replacing every occurrence of a minimal model quantifier, but the most external one, with the relative atomic proposition. Note now that, for all m, each $\tilde{\varphi} \in \Theta'_m$ is a MCTL* formula of the type $\tilde{\varphi} = \varphi' \,\Xi\, \varphi''$, with φ' and φ'' CTL* formulas and that $|\Theta'_m| = O(|\varphi|)$.

Now, set $\mathcal{K}_0 = \mathcal{K}$, we construct a sequence of Kripke structures $\mathcal{K}_m$ such that, for all $\tilde{\varphi} \in \Theta_m$ and $w \in \mathsf{dom}(\mathcal{K})$, $ep_i \in \mathrm{L}_m(w)$ iff $\mathcal{K}_{m-1}, w \models \tilde{\varphi}$, where ep_i is the atomic proposition relative to $\overline{\varphi}$. The latter can be checked by applying the PSPACE procedure of Lemma 1. Then, the result follows by recursively applying the above procedure. □

Theorem 5. MCTL *and* MCTL$^+$ *have a* Δ_2^p-COMPLETE *model checking problem.*

Proof. First note that, since by Theorem 3 MCTL and MCTL$^+$ are one into the other polynomial reducible, MCTL$^+$ simply extends CTL$^+$with minimal model quantifiers, and the latter has a Δ_2^p-COMPLETE model checking problem [LMS01], we have that MCTL has a Δ_2^p-HARD model checking. What remains to show is that it is in Δ_2^p. To prove this, we use a variation of the deterministic algorithm of Theorem 4, which, instead to call a PSPACE procedure to know if $\mathcal{K}, w \models \varphi_1 \Xi \varphi_2$ or not, call a NPTIME oracle (in accordance with Lemma 1), which solve the check in a single step. Now, since all other instructions of the algorithm are based on a classical CTL model checking procedure that can be executed in PTIME, we easily obtain a Δ_2^p model checking procedure for MCTL. Moreover, by Theorem 3, it holds that a MCTL$^+$ formula can be polynomially translated into a MCTL one, so the thesis follows also for this logic. □

Corollary 2. MPML *has a* Δ_2^p *model checking problem.*

5 Satisfiability

In this section we study the satisfiability for the introduced logics. We show that for all of them, but MPML, the question is undecidable. For MPML, we show decidability by using a brute force procedure via strong finite model property [BdRV04]. The result is reported in the following theorem.

Theorem 6. *The satisfiability problem for* MPML *is decidable.*

Proof. By Theorem 2, MPML has the strong finite model property w.r.t. a precise recursive function g. So for a given MPML formula φ we can construct a non deterministic Turing machine that, once the value of the function $g(|\varphi|)$ is computed, it guesses a model $\mathcal{K}$ of size at most equal to this value and then checks if it satisfies the formula by applying the decidable model checking procedure given by Corollary 2. Since φ is satisfiable iff it is satisfied on a model $\mathcal{K}$ of size at most $g(|\varphi|)$ and the built machine systematically examines all these kinds of models, the thesis easily follows. □

In rest of this section, we show undecidability of the satisfiability problem for MCTL, MCTL$^+$, and MCTL*through a reduction of a *domino problem* to it. This approach, often used in undecidability proofs in logic (see for example [BS99]), is classically known as *"undecidability via tiling"* [BdRV04].

The well-known domino problem, proposed for the first time by Wang [Wan61], consists of placing a given number of tile types on an infinite grid, satisfying a predetermined set of constraints on adjacent tiles. Its standard version asks for a compatible tiling of the whole plane $\mathbb{Z} \times \mathbb{Z}$. However, as stated by Knuth [Knu68], a compatible tiling of the first quadrant yields compatible tilings of arbitrary large finite rectangles, which in turn yields a compatible tiling of the whole plane. Since the existence of a solution for the original problem is known to be Π_0^1-COMPLETE [Ber66, Rob71], we have undecidable results (Π_0^1-HARD) also for the above variants of the classical domino problem. A formal definition of the $\mathbb{N} \times \mathbb{N}$ tiling problem follows.

Definition 3. **(Tiling System)** *A* $\mathbb{N} \times \mathbb{N}$ tiling system $\mathcal{D} = (\mathcal{T}, \mathcal{H}, \mathcal{V})$ *is a structure built on a non-empty set* $\mathcal{T}$ *of* domino types *and two* horizontal *and* vertical matching pairs $\mathcal{H}, \mathcal{V} \subseteq \mathcal{T}^2$. *The domino problem asks for a compatible tiling of the first quadrant* $(\mathbb{N} \times \mathbb{N})$ *of the plane, which is a* solution mapping $\tau : \mathbb{N}^2 \mapsto \mathcal{T}$ *such that, for all* $x, y \in \mathbb{N}$ *with* $\tau(x, y) = t$, *it holds that if* $\tau(x+1, y) = t'$, *then* $(t, t') \in \mathcal{H}$, *and if* $\tau(x, y+1) = t'$, *then* $(t, t') \in \mathcal{V}$. □

In the literature, an extension of the above problem has been also introduced as the *recurrent domino problem*. This problem, in addition to the tiling of the semiplane $\mathbb{N} \times \mathbb{N}$, asks whether there exists a distinguished tile type that occurs infinitely often in the first row of the grid. This problem is known to be more complex of the classical one. Indeed, it turns to be Σ_1^1-COMPLETE [Har84]. The formal definition follows.

Definition 4. **(Recurrent Tiling System)** *A* $\mathbb{N} \times \mathbb{N}$ recurrent tiling system $\mathcal{RD} = (\mathcal{T}, \mathcal{H}, \mathcal{V}, t^*)$ *is a structure in which* $\mathcal{D} = (\mathcal{T}, \mathcal{H}, \mathcal{V})$ *is a* $\mathbb{N} \times \mathbb{N}$ *tiling system and* $t^* \in \mathcal{T}$ *is a distinguished tile type. The recurrent domino problem asks for a solution mapping* τ *such that the set of horizontal index* $\{m \mid \tau(m, 0) = t^*\}$ *has an infinite cardinality.* □

By showing a reduction from the recurrent domino problem, we can prove in particular that the satisfiability for the MCTL logic is Σ_1^1-HARD. We achieve this reduction by showing that a given recurrent tiling system $\mathcal{RD}$ can be *"embedded"* into a model $\mathcal{K}_{\mathcal{RD}}$ of a particular formula $\varphi_{\mathcal{RD}}$ in such a way that $\varphi_{\mathcal{RD}}$ is satisfiable (i.e., it has a model) if and only if $\mathcal{RD}$ allows for a compatible tiling. To this aim we extend the proof structure used by Baader and Sattler [BS99]. For the sake of clarity, we split the reduction into four tasks, as described as follows:

Task 1 - (*Grid Specification*): It is possible to represent a "square structure" of $\mathbb{N} \times \mathbb{N}$, which consists of points (x, y), $(x+1, y)$, $(x, y+1)$, and $(x+1, y+1)$, in order to yield a complete covering of the semi-plane via a repeating regular grid structure. The basic idea is to use the minimal model quantifiers to force the horizontal successor of $(x, y+1)$ and the vertical successor of $(x+1, y)$ to correspond to the unique point $(x+1, y+1)$, with the aim to represent a square structure model on which to place the domino types. Formally, this can be expressed by using the following formula φ_{GS}, with $\alpha = a \wedge b$, $\beta = \neg a \wedge b$, $\gamma = a \wedge \neg b$, and $\delta = \neg a \wedge \neg b$:

$$\begin{aligned}
\varphi_H(\varphi') &= (\alpha \rightarrow \mathsf{EX}(\gamma \wedge \varphi')) \wedge (\beta \rightarrow \mathsf{EX}(\delta \wedge \varphi')) \wedge (\gamma \rightarrow \mathsf{EX}(\alpha \wedge \varphi')) \wedge (\delta \rightarrow \mathsf{EX}(\beta \wedge \varphi'));\\
\varphi_V(\varphi') &= (\alpha \rightarrow \mathsf{EX}(\beta \wedge \varphi')) \wedge (\beta \rightarrow \mathsf{EX}(\alpha \wedge \varphi')) \wedge (\gamma \rightarrow \mathsf{EX}(\delta \wedge \varphi')) \wedge (\delta \rightarrow \mathsf{EX}(\gamma \wedge \varphi'));\\
\varphi_S &= \varphi_V(\varphi_H(\varphi_V(\mathsf{t}))) \,\Xi\, (\varphi_V(\varphi_H(\mathsf{t})) \wedge \varphi_H(\varphi_V(\varphi_V(\mathsf{t}))));\\
\varphi_{U_H} &= \varphi_H(\varphi_H(\mathsf{t}) \wedge \varphi_V(\mathsf{t})) \,\Lambda\, (\varphi_H(\varphi_H(\mathsf{t})) \wedge \varphi_H(\varphi_V(\mathsf{t})));\\
\varphi_{U_V} &= \varphi_V(\varphi_H(\mathsf{t}) \wedge \varphi_V(\mathsf{t})) \,\Lambda\, (\varphi_V(\varphi_H(\mathsf{t})) \wedge \varphi_V(\varphi_V(\mathsf{t})));\\
\varphi_A &= ((\alpha \vee \delta) \rightarrow \mathsf{AX}(\beta \vee \gamma)) \wedge ((\beta \vee \gamma) \rightarrow \mathsf{AX}(\alpha \vee \delta));\\
\varphi_{GS} &= \varphi_S \wedge \varphi_{U_H} \wedge \varphi_{U_V} \wedge \varphi_A.
\end{aligned}$$

Task 2 - (*Compatible Tiling*): It is possible to express that a tiling is locally compatible, i.e., the two horizontal $(x+1, y)$ and vertical $(x, y+1)$ points have admissible domino types with respect to the (x, y) point. The idea here is to associate to each domino type $t \in \mathcal{T}$ an atomic proposition T_t and express the horizontal and vertical matching conditions via suitable object labeling. Note that these constraints are very easy to express. Indeed, they can be expressed in PML. Formally, we have:

$$\varphi_{CT} = \bigvee_{i \in \mathcal{T}} (T_i \wedge \bigwedge_{j \in \mathcal{T}}^{j \neq i} \neg T_j \wedge \varphi_H(\bigvee_{(i,j) \in \mathcal{H}} T_j) \wedge \varphi_V(\bigvee_{(i,j) \in \mathcal{V}} T_j)).$$

Task 3 - (*Recurrent Tile*): It is possible to assert that the distinguished tile type t^* occurs infinitely often on the first row of the semi-plane. This task can be easily achieved by using the kind of recursion available in the basic logic CTL. By means of this recursion, we can impose that the relative atomic proposition T_{t^*} is satisfied in an infinite number of worlds $v \in \mathsf{dom}(\mathcal{K}_{\mathcal{RD}})$, linearly reachable from the origin $w \in \mathsf{dom}(\mathcal{K}_{\mathcal{RD}})$ of the grid. Formally, we have:
$\varphi_{RT} = \varphi_V(\neg \varepsilon) \wedge (\varepsilon \rightarrow \varphi_H(\mathsf{EF}\,(\varepsilon \wedge T_{t^*}))).$

Task 4 - (*Global Reachability*): Finally, it is possible to impose that the above three conditions hold on all points in $\mathbb{N} \times \mathbb{N}$. As for the recurrent tile condition, also this task can be achieved by the simple recursion given by CTL. Formally, we have:
$\varphi_{GR} = \mathsf{AG}\,(\varphi_{GS} \wedge \varphi_{CT} \wedge \varphi_{RT}).$

We now give a formal proof of the undecidability, introducing the formula $\varphi_{\mathcal{RD}}$ who assemble all the above concepts.

Theorem 7. *The satisfiability problem for* MCTL*,* MCTL$^+$*, and* MCTL* *is highly undecidable. In particular, it is* Σ_1^1-HARD.

Proof. To prove the undecidability of the logic, we show the equivalence between find the solution of the recurrent tiling problem, with the distinguished tile type t^*, and the satisfiability of the formula $\varphi_{\mathcal{RD}} = \alpha \wedge \varepsilon \wedge \varphi_{GR}$.

Assume, for the direct reduction, that there exists a solution mapping τ. Then, we can build a satisfying model $\mathcal{K} = \langle \mathrm{AP}, \mathrm{W}, \mathrm{R}, \mathrm{L} \rangle$ that satisfies $\varphi_{\mathcal{RD}}$ as follows:

- $\mathrm{AP} = \{a, b, \varepsilon\} \cup \{T_t \mid t \in \mathcal{T}\}$;
- $\mathrm{W} = \mathbb{N} \times \mathbb{N}$;
- $\mathrm{R} = \{((m,n),(m+1,n)) \mid m,n \in \mathbb{N}\} \cup \{((m,n),(m,n+1)) \mid m,n \in \mathbb{N}\}$;
- for all $m,n \in \mathbb{N}$, it holds that: $a \in \mathrm{L}((m,n))$ if $n \equiv 0 \pmod 2$, $b \in \mathrm{L}((m,n))$ if $m \equiv 0 \pmod 2$, $\varepsilon \in \mathrm{L}((0,0))$ and $\varepsilon \in \mathrm{L}((m,0))$ if $\tau(m,0) = t^*$, and $T_t \in \mathrm{L}((m,n))$ if $\tau(m,n) = t$.

It is easy to see that $\mathcal{K} \models \varphi_{\mathcal{RD}}$, since $\mathcal{K}, (0,0) \models \varphi_{\mathcal{RD}}$.

Conversely, let $\mathcal{K}$ be a model such that there exists a world $w \in \mathsf{dom}(\mathcal{K})$ such that $\mathcal{K}, w \models \varphi_{\mathcal{RD}}$. First, we show that $\mathcal{K}$ is a *grid-like model* and then that is possible to construct a solution mapping τ from it. Indeed, since $\mathcal{K}, w \models \varphi_{\mathcal{RD}}$, we have that for all worlds $v \in \mathsf{dom}(\mathcal{K})$ reachable from w, (i.e., $(w,v) \in \mathrm{R}^n$, for some $n \in \mathbb{N}$) it holds that $\mathcal{K}, v \models \varphi_{GS}$ and thus $\mathcal{K}, v \models \varphi_S$. Now, it is not difficult to see that $\mathcal{K}$ must contain a square submodel in v (see proof of item *(i)* of Theorem 1 and structures $\mathcal{K}_3$ and $\mathcal{K}_4$ in Figure 1 for an example of models of φ_{GS}, where also holds that $\mathcal{K}_i, v \models \alpha$, for $i \in \{3,4\}$). Moreover, $\mathcal{K}, v \models \varphi_A$, so there are only two kinds of successors for v, i.e., if $\mathcal{K}, v \models \alpha$ or $\mathcal{K}, v \models \delta$ then v has successor worlds u, with $\mathcal{K}, u \models \beta$ or $\mathcal{K}, u \models \gamma$ and vice versa. Finally, since $\mathcal{K}, v \models \varphi_{U_H} \wedge \varphi_{U_V}$, if $\mathcal{K}, v \models \alpha$ or $\mathcal{K}, v \models \delta$, v has only one successor u' with $\mathcal{K}, u' \models \beta$ and only one successor u'' with $\mathcal{K}, u'' \models \gamma$ and viceversa. Now, it is clear that each world v reachable from w (including w itself) has only two successors u' and u'', which have a common successor o. Hence, $\mathcal{K}$ is a *grid-like model*. To extract a solution mapping τ from $\mathcal{K}$ is a routine task, so left to the reader. □

6 Conclusions

In this paper, we have introduced the branching-time temporal logic MCTL* as an extension of the classical branching-time temporal logic CTL* with minimal model quantifiers. These quantifiers allow to extract minimal submodels of a system model (even when the modularity of the system is not known in advance) on which we successively check a given property of the introduced logic.

We have deeply investigated, from a theoretical point of view, MCTL* and some of its sublogics. As far as the expressivity regards, we have showed that MCTL* is strictly more expressive than CTL*. Unfortunately, this power comes at a price. Indeed, the satisfiability problem for MCTL*, as well as for its sublogic MCTL, has been proved to be highly undecidable. Moreover, MCTL* does not have the tree model property, it is not bisimulation-invariant, and it is sensible to unwinding, opposed to CTL*.

As good news, we have showed that the sublogic MPML of MCTL* retains both the finite model property and the decidability of the satisfiability problem. Moreover, we have showed that the model checking problem for MCTL* remains decidable and in PSPACE. In more details and differently from CTL*, the PSPACE upper bound we provide is both in the size of the system and in the size of specification. Since for CTL it is only PSPACE in the size of the formula, it is left as an open question whether this extra complexity can be avoided. Anyway, although practical applications of MCTL* are not in the target of this paper, we argue that the extra blow-up for MCTL*should not have any consequence in practical applications as it can be absorbed in classical symbolic model checking algorithms, which are already exponential. Last but not least, we have investigated succinctness and the model checking problems for MCTL$^+$ and MCTL. We have shown that, differently from the classical case of CTL and CTL$^+$, MCTL is as succinct as MCTL$^+$. Moreover, as for CTL$^+$, the model checking problem for MCTL and MCTL$^+$ is Δ_2^p-COMPLETE (i.e., PTIME$^{\text{NPTIME}}$).

As future work, it would be worth investigating if the bisimulation-invariant fragment of MCTL* (i.e., the set of formulae that agree on bisimilar Kripke structures) is equally expressive as CTL*. In other words, we would like to check whether there exists an MCTL* formula which does not distinguish bisimilar structures, but it is still not expressible in CTL*. Then, if such a fragment is not equivalent to CTL*, it would be also relevant to investigate the related decidability problems.

Acknowledgement. We wish to thank Moshe Y. Vardi for useful discussions and the DLT 2009 referees for many helpful comments and suggestions.

References

[AHK02] Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-Time Temporal Logic. JACM 49(5), 672–713 (2002)

[AL93] Abadi, M., Lamport, L.: Composing Specifications. TOPLAS 15(1), 73–132 (1993)

[BdRV04] Blackburn, P., de Rijke, M., Venema, Y.: Modal Logic. Cambridge University Press, Cambridge (2004)

[Ber66] Berger, R.: The Undecidability of the Domino Problem. MAMS 66, 1–72 (1966)

[BLMV06] Bonatti, P.A., Lutz, C., Murano, A., Vardi, M.Y.: The Complexity of Enriched μ-Calculi. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4052, pp. 540–551. Springer, Heidelberg (2006)

[BMM09] Bianco, A., Mogavero, F., Murano, A.: Graded Computation Tree Logic. In: LICS 2009 (to appear, 2009)

[BS99] Baader, F., Sattler, U.: Expressive Number Restrictions in Description Logics. JLC 9(3), 319–350 (1999)

[CE81] Clarke, E.M., Emerson, E.A.: Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In: Kozen, D. (ed.) Logic of Programs 1981. LNCS, vol. 131, pp. 52–71. Springer, Heidelberg (1982)

[ECJB97] Elseaidy, W.M., Cleaveland, R., Baugh Jr., J.W.: Modeling and Verifying Active Structural Control Systems. SCP 29(1-2), 99–122 (1997)

[EH85] Emerson, E.A., Halpern, J.Y.: Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. JCSS 30(1), 1–24 (1985)

[Eme90] Emerson, E.A.: Temporal and Modal Logic. In: Handbook of Theoretical Computer Science, Formal Models and Sematics (B), vol. B, pp. 995–1072 (1990)

[FvD08] French, T., van Ditmarsch, H.P.: Undecidability for Arbitrary Public Announcement Logic. In: AIML, pp. 23–42 (2008)

[Har84] Harel, D.: A Simple Highly Undecidable Domino Problem. In: CLC 1984(1984)

[Knu68] Knuth, D.E.: The Art of Computer Programming, Fundamental Algorithms, vol. I. Addison-Wesley, Reading (1968)

[Kur94] Kurshan, R.P.: The Complexity of Verification. In: STOC 1994, pp. 365–371 (1994)

[KV95] Kupferman, O., Vardi, M.Y.: On the Complexity of Branching Modular Model Checking. In: Lee, I., Smolka, S.A. (eds.) CONCUR 1995. LNCS, vol. 962, pp. 408–422. Springer, Heidelberg (1995)

[KV97] Kupferman, O., Vardi, M.Y.: Modular Model Checking. In: de Roever, W.-P., Langmaack, H., Pnueli, A. (eds.) COMPOS 1997. LNCS, vol. 1536, pp. 381–401. Springer, Heidelberg (1998)

[KVW00] Kupferman, O., Vardi, M.Y., Wolper, P.: An Automata-Theoretic Approach to Branching-Time Model Checking. JACM 47(2), 312–360 (2000)

[Lam80] Lamport, L.: “Sometime" is Sometimes “Not Never": On the Temporal Logic of Programs. In: POPL 1980, pp. 174–185 (1980)

[LMS01] Laroussinie, F., Markey, N., Schnoebelen, P.: Model Checking CTL+ and FCTL is Hard. In: Honsell, F., Miculan, M. (eds.) FOSSACS 2001. LNCS, vol. 2030, pp. 318–331. Springer, Heidelberg (2001)

[LR03] Löding, C., Rohde, P.: Model Checking and Satisfiability for Sabotage Modal Logic. In: Pandya, P.K., Radhakrishnan, J. (eds.) FSTTCS 2003. LNCS, vol. 2914, pp. 302–313. Springer, Heidelberg (2003)

[Pel96] Peled, D.: Combining Partial Order Reductions with On-the-Fly Model Checking. FMSD 8(1), 39–64 (1996)

[Pnu77] Pnueli, A.: The Temporal Logic of Programs. In: FOCS 1977, pp. 46–57 (1977)

[Pnu81] Pnueli, A.: The Temporal Semantics of Concurrent Programs. TCS 13, 45–60 (1981)

[QS82] Queille, J.-P., Sifakis, J.: Specification and Verification of Concurrent Systems in CESAR. In: CISP 1982, pp. 337–351. Springer, Heidelberg (1982)

[Rob71] Robinson, R.M.: Undecidability and Nonperiodicity for Tilings of the Plane. IM 12, 177–209 (1971)

[Wan61] Wang, H.: Proving Theorems by Pattern Recognition II. BSTJ 40, 1–41 (1961)

[Wil99] Wilke, T.: CTL+ is Exponentially More Succinct than CTL. In: Pandu Rangan, C., Raman, V., Ramanujam, R. (eds.) FSTTCS 1999. LNCS, vol. 1738, pp. 110–121. Springer, Heidelberg (1999)

Simulations by Time-Bounded Counter Machines

Holger Petersen*

Reinsburgstraße 75
D-70197 Stuttgart
Germany

Abstract. We investigate the efficiency of simulations of storages by several counters. A simulation of a pushdown store is described which is optimal in the sense that reducing the number of counters of a simulator leads to an increase in time complexity. The lower bound also establishes a tight counter hierarchy in exponential time. Then we turn to simulations of a set of counters by a different number of counters. We improve and generalize a known simulation in polynomial time and we show a tight hierarchy result for machines working in the same polynomial bound with an increasing number of counters. We also prove hierarchies for machines with a fixed number of counters and with growing polynomial time bounds.

1 Introduction

It is a classical result due to Minsky [6] that machines equipped with two counters are universal. Therefore adding more counters to such a machine does not increase its power from a computability point of view. It may however influence the efficiency of simulating other devices and accepting certain languages. While the well-known simulation of a tape by two counters requires a double exponential increase in time [8, Theorem 2.1 (3)], two pushdown stores (and thus a tape) can be simulated by three counters with a single exponential overhead. For a storage consisting of several counters Greibach has given an explicit polynomial bound on the overhead of a simulation by three counters [2].

It is thus natural to ask how the complexity of these simulations is influenced by increasing the number of counters of the simulator. Since storages like tapes and queues can be simulated efficiently by pushdown stores, we will concentrate on simulations of a single pushdown store or a set of counters by several counters.

When considering upper bounds on simulations it is interesting to show lower bounds which might prove optimality of the simulations. The early work on counter machines by Fischer et al. [1] and Laing [5] established counter hierarchies for realtime recognition, thus showing that fewer counters never suffice for a general simulation in realtime. For palindrome recognition an exponential

[*] Work done while the author was with the FMI, Univ. Stuttgart, Universitätsstr. 38, D-70569 Stuttgart, Germany.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 410–418, 2009.

lower bound was established in [1, Theorem 1.3]. It seems to be a much more difficult task to separate counter machines having a polynomial, non-linear time-bound. Greibach was only partially successful when she showed that adding $s+1$ counters increases the power of counter machines operating in time n^s [2, Theorem 3.3]. Intuitively the obstacle to a tighter separation is that in polynomial time the counters have to store information in encoded form and the encoding will cause overhead that requires adding more counters than in realtime in order to obtain a separation.

Hromkovič and Schnitger compared the computational power of nondeterminism, randomization, and determinism for polynomial time counter machines [4].

In the present work we will improve simulations of a pushdown store and of many counters by few. We will also succeed in separating machines with a growing number of counters in the same time bound and of machines with a fixed number of counters and increasing time bounds.

2 Preliminaries

A *k-counter machine* is a device equipped with a deterministic finite control unit, k counters, each capable of storing an integer, and a one-way input tape. The set of states of the finite control unit is divided into disjoint sets of *autonomous states* and *polling states*. In an autonomous state the next step is determined by the current state and by the set of counters which contain zero, while in a polling state the next step also depends on the symbol read from the input tape. Each step may change the state of the control unit and alter the counters by some bounded amount. Initially the machine starts in a designated *initial state*. A *computation* consists of a sequence of steps. As in [2] we assume that the input tape has an endmarker. An input is *accepted* if the machine falls off the endmarker while entering a final state. The set of strings accepted by machine M is denoted by $L(M)$. Formal definitions of these concepts can be found in [1].

By Theorem 1.2 of [1] we may assume that the counter machines store only non-negative integers on the counters and alter at most one counter by at most one per step.

A machine M *accepts in time* $t(n)$ if on input $w \in L(M)$ the computation of M terminates within $t(|w|)$ steps, where $|w|$ is the length of w. Machine M *operates in time* $t(n)$ if on every input w the computation of M terminates after at most $t(n)$ steps, it *recognizes L in time* $t(n)$ if M operates in this time bound and accepts exactly the strings in L. We denote the class of languages accepted in time $t(n)$ by r-counter machines by

$$r\text{-CM DTIME}(t(n)) = \{L(M) \mid M \text{ is an } r\text{-counter machine} \\ \text{accepting in time } t(n).\}$$

Linear speed-up holds for super-linear, polynomial time-bounds. In contrast, linear time and realtime (one step per input symbol) define different classes of languages [1, Theorem 5.3].

We will also consider the space used by a counter machine, which is the maximum integer stored on any of its counters in the course of a computation.

The set of *marked palindromes* over a binary alphabet is

$$L = \{xcx^T \mid x \in \{a,b\}^*\},$$

where x^T denotes the reversal of x.

Next we will review some families of languages. The separation in [1] of counter machines working in realtime is achieved with the help of the infinite family of languages

$$L_k = \{0^{m_1}10^{m_2}1\cdots 0^{m_k}\beta_i 0^{m_i} \mid 1 \le i \le k, \\ \text{each } m_j \ge 1, \text{ and each } \beta_i \notin \{0,1\}\}.$$

Independently, Laing also separated machines with an increasing number of counters [5]. His separating languages take the form

$$M_k = \{w \in \{a_0, a_1, \ldots, a_k\}^* \mid |w|_{a_0} = |w|_{a_1} = \cdots = |w|_{a_k}\},$$

where $|w|_x$ denotes the number of symbols x in w.

Greibach [2] defined languages for $m \ge 0$ in the following way:

$$L'_{2m+1} = L \cap (a^+b^+)^m a^+ c\{a,b\}^*,$$
$$L'_{2m+2} = L \cap (a^+b^+)^{m+1} c\{a,b\}^*.$$

Language L'_k is very similar to L_k, encoding k numbers as lengths of blocks. But while for testing membership in L_k only one length has to be checked after reading β_i, in L'_k all blocks appear in reversed order after the symbol c.

The following family of languages N_k introduced in the present work has a structure that deviates from the ones defined above:

$$N_k = \{a^m u b^p c^q \$ c^q b^p u^T \mid m, p, q \ge 1, u \in \{0,1\}^{k \log m}\}.$$

3 Results

Theorem 1. *A sequence of $t(n)$ operations on a pushdown store containing symbols chosen from an alphabet of constant size $m \ge 2$ can be simulated by a machine with $k+1 \ge 2$ counters in time $O(m^{t(n)/k})$.*

Proof. The algorithm generalizes and strengthens the well-known simulation of a tape by three counters, see e.g. [3, Theorem 8.14]. We improve the construction by omitting the bottom symbol and encoding the pushdown contents in a less wasteful manner. In order to take advantage of more than three counters of the simulator, we encode the contents of the pushdown store on the first k counters while the $k+1$-st counter serves as auxiliary storage. A problem that we have to resolve is that the time bound (and thus the bound on the size of the pushdown store) depends on the length of the input, which is not known before the last symbol has been consumed. Therefore the distribution of information on the counters has to adapt to a growing input length.

The simulator M initializes counters 1 through k with a value of one, which represents empty storages. Simulating a push operation of symbol a_b with the help of counter j is done as follows. Machine M initializes counter $k+1$ with zero and repeatedly subtracts one from counter j while counter $k+1$ is incremented by m. When counter j reaches zero, number $b-1$ is added to counter $k+1$ and then the count is transferred back to counter j. Simulating a pop operation with the help of counter j is done in an analogous way by repeatedly subtracting m and computing the remainder modulo m. Emptiness is indicated by the value one. Notice that this explicit encoding of an empty storage allows us to utilize all m-ary digits including zero.

The information is distributed among the counters by storing a symbol at height i on the simulated pushdown store on counter $((i-1) \bmod k)+1$. The machine M can keep track of the currently used counter with the help of its finite control.

An upper bound on the numbers stored is $m^{t(n)/k+1}$. The maximum number of steps is attained if only push operations are simulated and then the time bound is $O(m^{t(n)/k})$. □

The above defined language L of palindromes with a marked center can be accepted in an obvious way by a machine with a single pushdown store over a binary alphabet in n steps. We therefore get:

Corollary 1. *The language L can be recognized in $O(2^{n/k})$ steps by a counter machine with $k+1$ counters.*

In order to obtain a tight separation we will next show that at least one counter of a deterministic machine with several counters has to store a “small” value at certain points of the computation. Hence not all counters can be used to transfer a lot of information across the input string.

Lemma 1. *If a counter machine M has q states then immediately after reading a prefix u of its input, the contents of at least one counter are bounded from above by $(q+1)\cdot|u|$.*

Proof. We show the lemma by induction on the length i of a prefix of the input which M reads.

For $i=0$ the machine is in its initial configuration with all counters empty and therefore the lemma holds.

For $i>0$ we distinguish two cases. If in the computation between reading the prefix of length $i-1$ and reading symbol i a counter becomes zero, then we consider the last instance when this is true for some counter j. Since M eventually consumes symbol i, there is a polling state in which symbol i is read. If M is in this state when counter j is zero, then the counter reaches at most one and the lemma holds. Otherwise in the first autonomous step counter j is necessarily incremented. Each of the at most $q-1$ autonomous states can appear at most once after the first step and before reading symbol i, because otherwise M works in a loop with all counters non-zero by the assumption that no counter reaches zero and M would not terminate. When M reads input symbol i it can increment counter j again. We conclude that counter j stores at most $(q+1) \leq (q+1) \cdot i$.

In the other case no counter ever reaches zero between reading symbol $i-1$ and symbol i. We make use of the induction hypothesis that at the beginning of this portion of the computation at least one counter is bounded by $(q+1)\cdot(i-1)$. By the same reasoning as in the other case this counter is incremented at most q times (the step with some counter empty does not occur in this case). The resulting bound is $q + (q+1) \cdot (i-1) \leq (q+1) \cdot i$. □

Counter machines operating in realtime and having a growing number of counters have been separated by Fischer et al. [1] and Laing [5] with the help of the languages L_k and M_k defined above. Note that the alphabets of these languages are growing with the levels being separated. In contrast we will now show a separation with the help of a fixed and more natural language, namely the set of marked palindromes.

Theorem 2. *The language L of marked palindromes can be recognized in time $O(2^{n/k})$ by a counter machine with $k+1$ counters, but no counter machine with k counters operating in time $O(2^{n/k})$ can recognize L.*

Proof. The upper bound is just Corollary 1. For the lower bound we assume that a counter machine M with q states and k counters accepts L. For any two words $x, y \in \{a,b\}^*$ with $x \neq y$ the machine has to be in different configurations after processing them, since xcx^T has to be accepted while ycx^T should be rejected. If $b(n)$ is the maximum number stored in some counter of M after reading an input string of length n, then an upper bound on the number of different configurations is $q \cdot (b(n)+1)^{k-1} \cdot k \cdot (q+1) \cdot n$ by Lemma 1. We conclude

$$q \cdot (b(n)+1)^{k-1} \cdot k \cdot (q+1) \cdot n \geq 2^n$$

and

$$b(n) = \Omega(2^{n/(k-1)}/n^{(k-1)}).$$

Since $2^{n/(k-1)} = 2^{n/k} \cdot 2^{n/k(k-1)}$, we have that the lower bound on the maximum number stored (and thus on the time complexity of M) grows faster than $2^{n/k}$. Hence M cannot operate in time $O(2^{n/k})$. □

Remark: In the previous proof we have made use of the fact that a machine recognizing L in a given time bound has to respect that bound also on input not belonging to L. The proof of the weaker lower bound in [1, Theorem 1.3 (a)] is based on accepted words only.

Greibach [2, Theorem 2.4] described a simulation of an r-counter machine accepting in space n^s and time n^t by a machine with three counters. We generalize this to simulators with $p \geq 3$ counters and obtain a better time bound (see also [9, Theorem 20.4]).

Theorem 3. *For $p, r \geq 3$ and $s, t \geq 1$ an r-counter machine accepting in space n^s and time n^t can be simulated by a p-counter machine in space $n^{rs/(p-2)}$ and time $n^{rs/(p-2)+t}$.*

Proof. The idea is to encode the r counters being simulated as values stored on $p-2$ counters, while two counters serve as scratch memory.

The machine M with p counters stores the at most $rs(\log n+1)$ bits necessary to encode the values of all r counters in $p-2$ sections of approximately the same size. Let us call the r bits of equal significance in all counters being simulated a group. The least significant bits are encoded on counter 1, while the most significant bits are stored on counter $p-2$. Initially all sections contain exactly one group with all bits zero and another set bit marking the most significant group. The effect of this initialization is that counters 1 through $p-2$ contain 2^r.

An increment operation on counter i is simulated by a bit-wise increment of the binary number encoded by the i-th bit of every group. If a carry propagates to a not yet existing bit position in counter $p-2$ because there are not enough groups, a new group is allocated. Machine M keeps track of the new groups in its finite control and as soon as this number becomes $p-2$ it redistributes the groups over the counters by shifting the $p-3$ least significant groups out of counter $p-2$, shifting in these groups (as the most significant ones) into counter $p-3$ while shifting out $p-4$ groups and so forth until the last group is shifted into counter 1. For these operations counters $r-1$ and r are available as auxiliary storage. A decrement operation is carried out in a similar way, but without the need to update the number of groups.

For the time complexity we note that a constant set of integers of size $O(2^{(rs\log n)/(p-2)}) = O(n^{rs/(p-2)})$ has to be be processed bit by bit, which is possible in time $O(n^{rs/(p-2)})$ per step, showing the claimed time bound. □

By substituting $w^q r$ for r and $(r+2)$ for p we obtain the following improvement of [2, Theorem 2.4 (2)] (see also [9, Theorem 20.6]):

Corollary 2. *For $q, r, s, t \geq 1$ and $w \geq 2$ a $w^q r$-counter machine accepting in space n^s and time n^t can be simulated by an $(r+2)$-counter machine in space $n^{w^q s}$ and time $n^{w^q s+t}$.*

Next we will consider separating counter machines with a polynomial time bound. Laing's language M_{rs} can be accepted by a machine with $r+2$ counters in time n^{s+1} by applying Theorem 3 to a realtime-machine with rs counters. This however does not lead to an improved separation in general, since L'_{rs} can be accepted by an $(r+s)$-counter machine within n^s steps [2, p. 282]. Our next result will however show the stronger statement that $r+2$ counters suffice for accepting L_{rs+1} and L'_{rs+1} in the same time bound.

Lemma 2. *For $r \geq 3$ and $s \geq 1$ the languages $L_{rs-2s+1}$ and $L'_{rs-2s+1}$ defined above can be accepted by r-counter machines in time n^s.*

Proof. The idea is to encode the length of blocks on the counters, where these blocks consist of zeroes for $L_{rs-2s+1}$ and of equal symbols a or b for $L'_{rs-2s+1}$.

For $r \geq 4$ the first $(r-3)s$ blocks are encoded in the following way. Counters $r-2$ through r serve as scratch memory. For $1 \leq i \leq (r-3)$ blocks $(i-1)s+1$ to is are encoded on counter i in binary by first storing the length of the block on counter $r-2$. Using the counter $r-1$ the length is successively divided by two and the bits decoded are stored on counter i with the help of every s-th bit. This requires division of counter i by two and temporarily storing the lower bits on counter r with the help of a multiplication by two. Since a value of size $O(n^s)$ is successively divided and multiplied by two, the time complexity is also $O(n^s)$. Then another $s-1$ blocks are encoded on counter $r-2$ with the technique from the proof of Theorem 3 using counters $r-1$ and r as scratch. Each of the at most n symbols requires one increment operation, which can be done in time $O(n^{s-1})$. This gives complexity $O(n^s)$. Finally the lengths of two blocks are stored on counters $r-1$ and r. The total number of blocks stored is thus $(r-3)s+(s-1)+2 = rs-2s+1$.

For decoding notice that for $L'_{rs-2s+1}$ after the symbol c blocks of identical length as in the first portion of the input have to appear, only the order is reversed. By applying the same technique as above the stored information can be decoded and compared to the input. For $L_{rs-2s+1}$ this task is even easier, since after symbol β_i has been read only one of the counts has to be accessed while all other counters become available as scratch memory. □

With the help of our simulations we obtain the following improvement over [2, Theorem 3.3 (2)] with respect to increasing time bounds.

Theorem 4. *In time n^s $(s \geq 1)$ machines with $r \geq 3$ counters are less powerful than machines with a time bound n^{2s+1}. Formally:*

$$r\text{-}CM\ DTIME(n^s) \subsetneq r\text{-}CM\ DTIME(n^{2s+1})$$

Proof. By the counting argument a machine with r counters cannot accept L_{rs-s+2} in time n^s. In time n^{2s+1} this language can be accepted by the method from the proof of Lemma 2 because $r(2s+1)-2(2s+1)+1 \geq rs-s+2$. □

Greibach [2, Theorem 3.3] has shown, that for $r, s \geq 1$

$$r\text{-CM DTIME}(n^s) \subsetneq (r+s+1)\text{-CM DTIME}(n^s).$$

Our improved upper and lower bounds on the languages L_k now yield an almost tight separation.

Theorem 5. *The language L_{rs+1} cannot be accepted in time n^s $(s \geq 1)$ by a machine with $r \geq 1$ counters, but can be accepted in the same time-bound by a machine with $r+2$ counters.*

Proof. Lemma 2 shows the upper bound. By the usual counting argument and Lemma 1 a machine with r counters can encode at most $(r-1)s+1 = rs-s+1$ blocks in time n^s and thus cannot accept L_{rs+1}. □

By considering a different family of languages we will next show a tight hierarchy result for counter machines with the same polynomial time bound.

Theorem 6. *In time n^s $(s \geq 1)$ a machine with $r \geq 1$ counters is less powerful than a machine with $r+1$ counters. Formally:*

$$r\text{-}CM\ DTIME(n^s) \subsetneq (r+1)\text{-}CM\ DTIME(n^s)$$

Proof. For $r = 1$ the languages accepted by r-counter machines are context-free, while a 2-counter machine can accept $\{a^n b^n c^n \mid n \geq 0\}$ in realtime.

We now show, that a machine M with $r + 1 \geq 3$ counters can accept $N_{(r-1)s}$ in time n^s.

The machine M first reads the maximal prefix consisting of the symbol a and stores its length m on counter 2. Then it repeatedly divides the contents of counter 2 by two using counter 3 as scratch memory and counts the number of iterations on counter 1. In this way M computes $\log m$ as the contents of counter 1.

Then M encodes the following string u over $\{0, 1\}$ on $r - 1$ counters using the technique from the proof of Theorem 1 with counter 2 as scratch memory, while the length of u is checked to be $(r - 1)s \log m$ with the help of counter 1. After encoding u counters 1 and 2 are used to store the numbers of b's and c's. After reading the symbol \$ the encoded information is compared with the input.

Now we show that no machine with r counters can accept $N_{(r-1)s}$. By Lemma 1 such a machine can reach $O(n^{(r-1)s+1})$ configurations in at most n^s steps. For a given n we now consider all strings of the form $a^{n/4} u b^p c^q$ with $u \in \{0, 1\}^{(r-1)s(\log n-2)}$ and $p, q \leq n/3$. For n sufficiently large these strings have length at most n. Clearly every machine accepting $N_{(r-1)s}$ has to be in a unique configuration for every string. The number of these strings is at least $2^{(r-1)s(\log n-2)} \cdot n/3 \cdot n/3 = \Omega(n^{(r-1)s+2})$. Thus no machine with r counters can accept $N_{(r-1)s}$. □

Finally we consider languages over a single letter alphabet. In this setting the combinatorial approach of the previous results fails. We can however apply a separation result from the area of finite multihead automata over a single letter alphabet, which is based on a diagonal argument combined with transformational methods [7].

Theorem 7. *In time n^s $(s \geq 1)$ a machine with $r \geq 3$ counters is less powerful than a machine with r counters and time bound n^{2rs+10} for languages over a single letter alphabet, or formally:*

$$\exists L \subseteq \{0\}^* : L \in \left[r\text{-}CM\ DTIME(n^{2rs+10}) - r\text{-}CM\ DTIME(n^s)\right]$$

Proof. By Theorem 3 an r-counter machine accepting in time (and hence space) n^s can be simulated by an $rs + 2$-counter machine in space n. Monien showed in [7, Theorem 3] that there is a tight hierarchy for two-way counter machines with linear bounds on the counters. Therefore there is a machine M_1 with $rs + 3$ linear bounded counters which cannot be simulated with the help of $rs + 2$ such counters. A two-way input tape can be simulated with two additional linear bounded counters that measure the distance of the simulated input head to the endmarkers. We therefore obtain a language, which can be accepted by a one-way counter-machine M_2 with $rs + 5$ linear bounded counters. Note that the bound on the number of configurations implies a time bound in $O(n^{rs+5})$. Then we simulate M_2 according to Theorem 3 with the help of an r-counter automaton M_3 in time n^{2rs+10}. Now $L(M_3)$ is a language accepted by an r-counter machine in time n^{2rs+10} but not in time n^s. □

4 Open Problems

The upper bounds reported in Theorem 1 through 3 carry over to nondeterministic machines, but the lower bounds depend on the determinism of the machines.

Although we have separated machines with an increasing number of counters working in polynomial time, the optimality of the simulation in Theorem 3 remains open. Also the separation in Theorem 7 can most likely be improved.

Acknowledgments

I thank Benjamin Hoffmann and the anonymous referees for several helpful remarks.

References

1. Fischer, P.C., Meyer, A.R., Rosenberg, A.L.: Counter machines and counter languages. Mathematical Systems Theory 2, 265–283 (1968)
2. Greibach, S.A.: Remarks on the complexity of nondeterministic counter languages. Theoretical Computer Science 1, 269–288 (1976)
3. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation, 3rd edn. Pearson, London (2007)
4. Hromkovič, J., Schnitger, G.: On the power of randomized multicounter machines. Theoretical Computer Science 330, 135–144 (2005)
5. Laing, R.: Realization and complexity of commutative events. Technical Report 03105-48-T, University of Michigan (1967)
6. Minsky, M.L.: Recursive unsolvability of Post's problem of "tag" and other topics in theory of Turing machines. Annals of Mathematics 74, 437–455 (1961)
7. Monien, B.: Two-way multihead automata over a one-letter alphabet. R.A.I.R.O. — Informatique Théorique et Applications 14, 67–82 (1980)
8. van Emde Boas, P.: Machine models and simulations. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. A, pp. 1–66. Elsevier, Amsterdam (1990)
9. Wagner, K., Wechsung, G.: Computational Complexity. Mathematics and its Applications. D. Reidel Publishing Company, Dordrecht (1986)

Weighted Timed MSO Logics

Karin Quaas

Institut für Informatik, Universität Leipzig
04009 Leipzig, Germany
quaas@informatik.uni-leipzig.de

Abstract. We aim to generalize Büchi's fundamental theorem on the coincidence of recognizable and MSO-definable languages to a weighted timed setting. For this, we investigate subclasses of weighted timed automata and show how we can extend existing timed MSO logics with weights. Here, we focus on the class of weighted event-recording automata and define a weighted extension of the full logic $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$ introduced by D'Souza. We show that every weighted event-recording automaton can effectively be transformed into a corresponding sentence of our logic and vice versa. The methods presented in the paper can be adopted to weighted versions of timed automata and Wilke's logic of relative distance. The results indicate the robustness of weighted timed automata models and may be used for specification purposes.

Introduction

Recently, the model of *weighted timed automata* has received much attention in the real-time community as it can be used to model continuous consumption of resources [2,3,5,4,11]. The goal of this paper is to generalize Büchi's and Elgot's fundamental theorems about the coincidence of languages *recognizable* by finite automata and languages *definable* by sentences in a monadic second-order (MSO) logic [6,15] to weighted timed automata. For this, we introduce a weighted timed MSO logic, which may be used for specifying *quantitative* aspects of timed automata, e.g. *how often* a certain property is satisfied by the system.

In this paper, we focus on a weighted version of *event-recording automata*, a subclass of timed automata introduced by Alur et al. [1]. Recent results on event-recording automata include works on alternative characterizations using regular expressions [7] and MSO logic [14], real-time logics [25,17], and inference/learning [16]. The main advantage of event-recording automata is that they - as opposed to timed automata - always can be determinized. This simplifies some of our constructions compared to the ones necessary for the class of weighted timed automata.

Our work is motivated by recent works on *weighted logics* by Droste and Gastin [8,10]. The authors introduce a weighted MSO logic for characterizing the behaviour of weighted automata defined over a semiring. They extend classical MSO logic with formulas of the form k (for k an element of the semiring), which may be used to define the weight of a transition of a weighted automaton. They

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 419–430, 2009.

show that the behaviour of weighted automata coincides with the semantics of sentences of a fragment of the logic. Recently, this result has been generalized to weighted settings of infinite words [12], trees [13], pictures [20], traces [21], texts [18] and nested words [19].

Here, we aim to generalize the result to a weighted timed setting. The basis of our work is the MSO logic $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$ introduced by D'Souza and used for the logical characterization of event-recording automata [14]. We extend it with two kinds of weighted formulas whose semantics correspond to the weights of edges and locations, respectively, in weighted event-recording automata. For proving a Büchi-type theorem we show that for every sentence φ in our logic there is a weighted event-recording automaton whose behaviour corresponds to the semantics of φ and vice versa.

For this, we use parts of the proofs presented by Droste and Gastin [10]. However, in the weighted timed setting we are faced with two new problems. First, due to the weights assigned to locations, the Hadamard product, which is used for defining the semantics of conjunction in our logic, does not preserve recognizability. Second, there are formulas φ such that there are no weighted event-recording automata whose behaviours correspond to the semantics of $\forall x.\varphi$ and $\forall X.\varphi$, respectively. To overcome these problems, we define a suitable fragment of our logic, for which, with the support of some new notions and techniques, we are able to show the result.

1 (Weighted) Event-Recording Automata

Let Σ, $\mathbb{N}$ and $\mathbb{R}_{\geq 0}$ denote an alphabet, the natural numbers and the positive reals, respectively. A *timed word* is a finite sequence $(a_1, t_1)...(a_k, t_k) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$ such that the sequence $\bar{t} = t_1...t_k$ of timestamps is non-decreasing. Sometimes we denote a timed word as above by $(\bar{a}, \bar{t})$, where $\bar{a} \in \Sigma^*$. We write $T\Sigma^*$ for the set of timed words over Σ. A set $L \subseteq T\Sigma^*$ is called a *timed language*. With Σ we associate a set $C_\Sigma = \{x_a | a \in \Sigma\}$ of *event-recording clock variables* ranging over $\mathbb{R}_{\geq 0}$. The variable x_a measures the time distance between the current event in a timed word and the last occuring a. Formally, given a timed word $w = (a_1, t_1)...(a_k, t_k)$, we let $\mathrm{dom}(w)$ be the set $\{1, ..., k\}$ and define for every $i \in \mathrm{dom}(w)$ a clock valuation function $\gamma_i^w : C_\Sigma \to \mathbb{R}_{\geq 0} \cup \{\bot\}$ by

$$\gamma_i^w(x_a) = \begin{cases} t_i - t_j & \text{if there exists a } j \text{ such that } 1 \leq j < i \text{ and } a_j = a, \\ & \text{and for all } m \text{ with } j < m < i, \text{ we have } a_m \neq a \\ \bot & \text{otherwise.} \end{cases}$$

We further use $|w|$ to denote the length of w. We define *clock constraints* ϕ over C_Σ to be conjunctions of formulas of the form $x = \bot$ or $x \sim c$, where $x \in C_\Sigma$, $c \in \mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$. We use $\Phi(C_\Sigma)$ to denote the set of all clock constraints over C_Σ. A clock valuation γ_i^w satisfies ϕ, written $\gamma_i^w \models \phi$, if ϕ evaluates to true according to the values given by γ_i^w. An *event-recording automaton (ERA)* over Σ is a tuple $\mathcal{A} = (S, S_0, S_f, E)$, where

- S is a finite set of locations (states)
- $S_0 \subseteq S$ is a set of initial locations
- $S_f \subseteq S$ is a set of final locations
- $E \subseteq S \times \Sigma \times \Phi(C_\Sigma) \times S$ is a finite set of edges.

For w as above, we let a *run* of $\mathcal{A}$ on w be a finite sequence $s_0 \xrightarrow{e_1} s_1 \xrightarrow{e_2} ... \xrightarrow{e_k} s_k$ of edges $e_i = (s_{i-1}, a_i, \phi_i, s_i) \in E$ such that $\gamma_i^w \models \phi_i$ for all $1 \leq i \leq k$. We say that a run r is *successful* if $s_0 \in S_0$ and $s_k \in S_f$. We define the timed language $L(\mathcal{A}) = \{w \in T\Sigma^* |$ there is a successful run of $\mathcal{A}$ on $w\}$. We say that a timed language $L \subseteq T\Sigma^*$ is *recognizable over* Σ if there is an ERA $\mathcal{A}$ over Σ such that $L(\mathcal{A}) = L$.

Remark 1. The methods presented in this paper can easily be extended to event-clock automata additionally equipped with *event-predicting* clock variables [1].

An ERA $\mathcal{A}$ is *deterministic* if $|S_0| = 1$ and whenever (s, a, ϕ_1, s_1) and (s, a, ϕ_2, s_2) are two different edges in $\mathcal{A}$, then for all clock valuations γ we have $\gamma \not\models \phi_1 \wedge \phi_2$. A timed language is called *deterministically recognizable over* Σ if there is a deterministic ERA over Σ recognizing it.

Proposition 1. *[1] The class of recognizable timed languages is closed under boolean operations and equal to the class of deterministically recognizable timed languages.*

We extend ERA to be equipped with weights taken from a commutative semiring. For this, we let $\mathcal{K}$ be a *commutative semiring*, i.e., an algebraic structure $\mathcal{K} = (K, +, \cdot, 0, 1)$ such that $(K, +, 0)$ and $(K, \cdot, 1)$ are commutative monoids, multiplication distributes over addition and 0 is absorbing. As examples consider the semiring of natural numbers $(\mathbb{N}, +, \cdot, 0, 1)$, the Boolean semiring $(\{0, 1\}, \vee, \wedge, 0, 1)$ and the tropical semiring $(\mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +, \infty, 0)$. Furthermore, we let $\mathcal{F}$ be a family of functions from $\mathbb{R}_{\geq 0}$ to $\mathcal{K}$. For instance, if $\mathcal{K}$ is the tropical semiring, $\mathcal{F}$ may be the family of linear functions of the form $\mu(\delta) = k \cdot \delta$ mapping every $\delta \in \mathbb{R}_{\geq 0}$ to a value $k \cdot \delta$ in K (for some $k \in \mathbb{R}_{\geq 0}$). Given $f_1, f_2 \in \mathcal{F}$, we define the pointwise product $f_1 \odot f_2$ of f_1 and f_2 by $(f_1 \odot f_2)(\delta) = f_1(\delta) \cdot f_2(\delta)$.

A *weighted event-recording automaton* (WERA) over Σ, $\mathcal{K}$ and $\mathcal{F}$ is a tuple $\mathcal{A} = (S, S_0, S_f, E, C)$ such that (S, S_0, S_f, E) is an ERA over Σ and $C = \{C_\mathcal{E}\} \cup \{C_s | s \in S\}$ is a cost function, where $C_\mathcal{E} : E \to K$ assigns a weight to each edge, and $C_s \in \mathcal{F}$ gives us the weight for staying in location s per time unit for each $s \in S$. A WERA $\mathcal{A}$ maps to each timed word $w \in T\Sigma^*$ a weight in K as follows: first, we define the *running weight* $rwt(r)$ of a run r as above to be $\prod_{i \in \mathrm{dom}(w)} C_{s_{i-1}}(t_i - t_{i-1}) \cdot C_\mathcal{E}(e_i)$, where $t_0 = 0$. The running weight of the empty run is defined to be $1 \in K$. Then, the *behaviour* $\|\mathcal{A}\| : T\Sigma^* \to K$ of $\mathcal{A}$ is given by $(\|\mathcal{A}\|, w) = \sum\{rwt(r) : r$ is a successful run of $\mathcal{A}$ on $w\}$. A function $\mathcal{T} : T\Sigma^* \to K$ is called a *timed series*. A timed series $\mathcal{T}$ is said to be *recognizable over* $\mathcal{K}$*,* Σ *and* $\mathcal{F}$ if there is a WERA $\mathcal{A}$ over $\mathcal{K}$, Σ and $\mathcal{F}$ such that $\|\mathcal{A}\| = \mathcal{T}$.

We define the function $\mathbb{1} : \mathbb{R}_{\geq 0} \to K$ by $\delta \mapsto 1$ for every $\delta \in \mathbb{R}_{\geq 0}$. In the following, we fix a commutative semiring $\mathcal{K}$ and a family $\mathcal{F}$ of cost functions from $\mathbb{R}_{\geq 0}$ to K containing $\mathbb{1}$.

For $L \subseteq T\Sigma^*$, the *characteristic series* 1_L is defined by $(1_L, w) = 1$ if $w \in L$, 0 otherwise. Notice that an ERA over Σ can be seen as a WERA over the Boolean semiring, Σ and the family of constant functions. The timed series recognized by such a WERA is the characteristic series $1_{L(\mathcal{A})}$. However, due to the determinizability of ERA, $1_{L(\mathcal{A})}$ can also be recognized over arbitrary semirings:

Lemma 1. *If $L \subseteq T\Sigma^*$ is recognizable over Σ, then 1_L is recognizable over $\mathcal{K}$, Σ and $\mathcal{F}$.*

Given timed series $\mathcal{T}, \mathcal{T}_1, \mathcal{T}_2$ and $k \in K$, we define the *sum* $\mathcal{T}_1 + \mathcal{T}_2$, the *Hadamard product* $\mathcal{T}_1 \odot \mathcal{T}_2$ and the *scalar products* $k \cdot \mathcal{T}$ and $\mathcal{T} \cdot k$ pointwise, i.e., by $(\mathcal{T}_1 + \mathcal{T}_2, w) = (\mathcal{T}_1, w) + (\mathcal{T}_2, w)$, $(\mathcal{T}_1 \odot \mathcal{T}_2, w) = (\mathcal{T}_1, w) \cdot (\mathcal{T}_2, w)$, $(k \cdot \mathcal{T}, w) = k \cdot (\mathcal{T}, w)$ and $(\mathcal{T} \cdot k, w) = (\mathcal{T}, w) \cdot k$ respectively. If $\mathcal{K}$ is the Boolean semiring, $+$ and $\odot$ correspond to the union and intersection of timed languages, respectively.

Later in the paper, we need closure properties of recognizable timed series under these operations. It can be shown in a straightforward manner that sum and scalar products preserve recognizability of timed series.

Lemma 2. *Recognizable timed series over $\mathcal{K}$, Σ and $\mathcal{F}$ are closed under $+$, $k\cdot$ and $\cdot k$ (for $k \in K$).*

In contrast to this, in general recognizable timed series are not closed under the Hadamard product. We illustrate this in the next example.

Example 1. Let $\mathcal{K} = (\mathbb{R}_{\geq 0} \cup \{\infty\}, \min, +, \infty, 0)$, $\Sigma = \{a\}$ and $\mathcal{F}$ be the family of linear functions $C : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. We define the WERA $\mathcal{A}^i$ over $\mathcal{K}$, Σ and $\mathcal{F}$ for each $i = 1, 2$ by $\mathcal{A}^i = (\{p^i, q^i\}, \{p^i\}, \{q^i\}, \{(p^i, a, true, q^i)\}, C^i)$ with $C^i_{\mathcal{E}}((p^i, a, true, q^i)) = 0$, $C^i_{q^i}(\delta)$ arbitrary, $C^1_{p^1}(\delta) = 2 \cdot \delta$ and $C^2_{p^2}(\delta) = 3 \cdot \delta$. Let $w \in T\Sigma^*$. If $w \neq (a, t)$ for some $t \in \mathbb{R}_{\geq 0}$, then $(\|\mathcal{A}^i\|, w) = 0$ for each $i = 1, 2$ and thus $(\|\mathcal{A}^1\| \odot \|\mathcal{A}^2\|, w) = 0$. So let $w = (a, t)$ for some $t \in \mathbb{R}_{\geq 0}$. Then we have $(\|\mathcal{A}^1\| \odot \|\mathcal{A}^2\|, w) = 2 \cdot t + 3 \cdot t = 5 \cdot t$. Clearly, this timed series is recognizable over the family of linear functions. If $\mathcal{K}$ and $\mathcal{F}$ are as above, for building a WERA recognizing the Hadamard product of the behaviours of two given WERA, we can use the usual product automaton construction together with defining a cost function such that the cost of each edge and location equals the pointwise product of the costs of the two corresponding edges and locations in the original WERA. This can be done since the pointwise product of each pair of linear functions is a linear function and thus in $\mathcal{F}$. However, this is not always the case. For instance, assume that $\mathcal{A}^i$ are WERA over the semiring $(\mathbb{R}_{\geq 0}, +, \cdot, 0, 1)$. Then, we have $(\|\mathcal{A}^1\| \odot \|\mathcal{A}^2\|, w) = 2 \cdot t \cdot 3 \cdot t = 6 \cdot t^2$. It can be easily seen that there is no WERA $\mathcal{A}$ over the family $\mathcal{F}$ of **linear** functions such that $\|\mathcal{A}\| = \|\mathcal{A}^1\| \odot \|\mathcal{A}^2\|$.

For this reason, we define the notion of *non-interfering* timed series. So for $i = 1, 2$, let $\mathcal{A}^i = (S^i, S^i_0, S^i_f, E^i, C^i)$ be two WERA. We say that $\mathcal{A}^1$ and $\mathcal{A}^2$ are *non-interfering* if for all pairs $(s_1, s_2) \in S^1 \times S^2$, whenever there is a run from (s_1, s_2) into $S^1_f \times S^2_f$, then $C^1_{s_1} = \mathbb{1}$ or $C^2_{s_2} = \mathbb{1}$. Observe that this implies

$C^1_{s_1} \odot C^2_{s_2} \in \mathcal{F}$. This enables us to use a product automaton construction for building a WERA recognizing $\|\mathcal{A}^1\| \odot \|\mathcal{A}^2\|$. Also notice that the premise of the condition is decidable for the whole class of weighted timed automata [2]. Two timed series $\mathcal{T}_1$ and $\mathcal{T}_2$ are non-interfering over $\mathcal{K}$, Σ and $\mathcal{F}$ if there are non-interfering WERA $\mathcal{A}^1$ and $\mathcal{A}^2$ over $\mathcal{K}$, Σ and $\mathcal{F}$ with $\|\mathcal{A}^i\| = \mathcal{T}_i$ for $i = 1, 2$.

Lemma 3. *1. If for all $f_1, f_2 \in \mathcal{F}$ we have $f_1 \odot f_2 \in \mathcal{F}$, then recognizable timed series over $\mathcal{K}$, Σ and $\mathcal{F}$ are closed under $\odot$.*

2. If $\mathcal{T}_1$ and $\mathcal{T}_2$ are non-interfering over $\mathcal{K}$, Σ and $\mathcal{F}$, then $\mathcal{T}_1 \odot \mathcal{T}_2$ is recognizable over $\mathcal{K}$, Σ and $\mathcal{F}$.

2 Weighted Timed MSO Logic

Next, we introduce a weighted timed MSO logic for specifying properties of timed series. Our logic is an extension of the logic $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$ introduced by D'Souza, which we briefly recall here. Formulas of $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$ are built inductively from atomic formulas $P_a(x)$, $x = y$, $x < y$, $x \in X$, $\lhd_a(x) \sim c$ using the connectives $\vee$, $\neg$, $\exists x.$ and $\exists X.$, where x, y are first-order variables, X is a second-order variable, $a \in \Sigma$, $c \in \mathbb{N}$ and $\sim \in \{<, \leq, =, \geq, >\}$ or $(\sim c) = (= \bot)$. As usual, we may also use $\rightarrow$, $\leftrightarrow$, $\wedge$, $\forall x.$ and $\forall X.$ as abbreviations. Formulas of $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$ are interpreted over timed words. For this, we associate with $w = (a_1, t_1)...(a_k, t_k)$ the relational structure consisting of the domain $\mathrm{dom}(w)$ together with the unary relations $P_a = \{i \in \mathrm{dom}(w) | a_i = a\}$ and $\lhd_a(.) \sim c = \{i \in \mathrm{dom}(w) | \gamma^w_i(x_a) \sim c\}$ as well as the usual $<$ and $=$ relations on $\mathrm{dom}(w)$. Now, for $\varphi \in \mathrm{MSO}_{\mathrm{er}}(\Sigma)$, let $\mathrm{Free}(\varphi)$ be the set of free variables, $\mathcal{V} \supseteq \mathrm{Free}(\varphi)$ be a finite set of first- and second-order variables, and σ be a $(\mathcal{V}, w)$-assignment mapping first-order (second-order, resp.) variables to elements (subsets, resp.) of $\mathrm{dom}(w)$. For $i \in \mathrm{dom}(w)$, we let $\sigma[x \rightarrow i]$ be the assignment that maps x to i and agrees with σ on every variable $\mathcal{V} \backslash \{x\}$. Similarly, we define $\sigma[X \rightarrow I]$ for any $I \subseteq \mathrm{dom}(w)$. A timed word $(\bar{a}, \bar{t})$ and a $(\mathcal{V}, (\bar{a}, \bar{t}))$-assignment σ is encoded as timed word $((\bar{a}, \sigma), \bar{t})$ over the extended alphabet $\Sigma_{\mathcal{V}}$. A timed word over $\Sigma_{\mathcal{V}}$ is written as $((\bar{a}, \sigma), \bar{t})$, where $\bar{a}$ is the projection over Σ and σ is the projection over $\{0, 1\}^{\mathcal{V}}$. Then, σ represents a *valid* assignment over $\mathcal{V}$ if for each first-order variable $x \in \mathcal{V}$, the x-row of σ contains exactly one 1. In this case, σ is identified with the $(\mathcal{V}, (\bar{a}, \bar{t}))$-assignment such that for every first-order variable $x \in \mathcal{V}$, $\sigma(x)$ is the position of the 1 in the x-row, and for each second-order variable $X \in \mathcal{V}$, $\sigma(X)$ is the set of positions with a 1 in the X-row. We define $N_{\mathcal{V}} = \{((\bar{a}, \sigma), \bar{t}) \in T(\Sigma_{\mathcal{V}})^* | \sigma \text{ is a valid } (\mathcal{V}, (\bar{a}, \bar{t}))\text{-assignment}\}$. The definition that $((\bar{a}, \sigma), \bar{t})$ satisfies φ, written $((\bar{a}, \sigma), \bar{t}) \models \varphi$, is as usual. We let $L_{\mathcal{V}}(\varphi) = \{((\bar{a}, \sigma), \bar{t}) \in N_{\mathcal{V}} | ((\bar{a}, \sigma), \bar{t}) \models \varphi\}$. The formula φ *defines* the timed language $L(\varphi) = L_{\mathrm{Free}(\varphi)}(\varphi)$. A timed language $L \subseteq T\Sigma^*$ is *$MSO_{er}(\Sigma)$-definable* if there exists a sentence $\varphi \in \mathrm{MSO}_{\mathrm{er}}(\Sigma)$ such that $L(\varphi) = L$.

Theorem 1. *[14] A timed language $L \subseteq T\Sigma^*$ is $MSO_{er}(\Sigma)$-definable if and only if L is recognizable over Σ.*

Now, we turn to the logic $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$, defined inductively as follows. The *atomic* formulas are formulas of the form $P_a(x)$, $x = y$, $x < y$, $x \in X$, $\lhd_a(x) \sim c$ and their negations, where $x, y, X, a, c, \sim$ are as above. Atomic formulas and formulas of the form k and $C_\mu(x)$, where $k \in K$ and $\mu \in \mathcal{F}$, can be combined using the operators $\wedge$, $\vee$, $\exists x.$, $\forall x.$, $\exists X.$ and $\forall X$. Notice that we only allow to apply negation to basic formulas. Let $\varphi \in \mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ and $\mathcal{V} \supseteq \mathrm{Free}(\varphi)$. The $\mathcal{V}$-semantics of φ is a timed series $[\![\varphi]\!]_\mathcal{V} : T(\Sigma_\mathcal{V})^* \to K$. Let $(\bar{a}, \bar{t}) \in T\Sigma^*$. If σ is a valid $(\mathcal{V}, (\bar{a}, \bar{t}))$-assignment, $[\![\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) \in K$ is defined inductively as follows:

$$\begin{aligned}
[\![\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= 1_{L_\mathcal{V}(\varphi)}((\bar{a}, \sigma), \bar{t}) \text{ if } \varphi \text{ is atomic} \\
[\![k]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= k \\
[\![C_\mu(x)]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= \mu(t_{\sigma(x)} - t_{\sigma(x)-1}) \\
[\![\varphi \vee \varphi']\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= [\![\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) + [\![\varphi']\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) \\
[\![\varphi \wedge \varphi']\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= [\![\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) \cdot [\![\varphi']\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) \\
[\![\exists x.\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= \sum_{i \in \mathrm{dom}((\bar{a}, \bar{t}))} [\![\varphi]\!]_{\mathcal{V} \cup \{x\}}((\bar{a}, \sigma[x \to i]), \bar{t}) \\
[\![\forall x.\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= \prod_{i \in \mathrm{dom}((\bar{a}, \bar{t}))} [\![\varphi]\!]_{\mathcal{V} \cup \{x\}}((\bar{a}, \sigma[x \to i]), \bar{t}) \\
[\![\exists X.\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= \sum_{I \subseteq \mathrm{dom}((\bar{a}, \bar{t}))} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}((\bar{a}, \sigma[X \to I]), \bar{t}) \\
[\![\forall X.\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) &= \prod_{I \subseteq \mathrm{dom}((\bar{a}, \bar{t}))} [\![\varphi]\!]_{\mathcal{V} \cup \{X\}}((\bar{a}, \sigma[X \to I]), \bar{t})
\end{aligned}$$

For σ not a valid $(\mathcal{V}, (\bar{a}, \bar{t}))$-assignment, we define $[\![\varphi]\!]_\mathcal{V}((\bar{a}, \sigma), \bar{t}) = 0$. We write $[\![\varphi]\!]$ for $[\![\varphi]\!]_{\mathrm{Free}(\varphi)}$.

Remark 2. If we let $\mathcal{K}$ be the Boolean semiring, then $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$ corresponds to $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ as every formula in $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$ is equivalent to a formula where negation is applied to atomic subformulas only. Also, every such formula $\varphi \in \mathrm{MSO}_{\mathrm{er}}(\Sigma)$ can be seen to be a formula of our logic.

Example 2. Consider the formula $\varphi = \exists x. \lhd_a (x) < 2$ and let $w = (a, 1.7)(b, 3.0)(a, 3.6)(a, 6.0)$. If we interpret φ as an $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$-formula over the Boolean semiring or, equivalently, as an $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$-formula, we have $[\![\varphi]\!](w) = 1$. If on the other hand, we let $\mathcal{K}$ be the semiring over the natural numbers with ordinary addition and multiplication, we have $[\![\varphi]\!](w) = 2$, i.e., we *count* the number of positions x in w where $\lhd_a(x) < 2$ is satisfied. Counting how often a certain property holds gives rise to interesting applications in the field of verification.

Let $\mathcal{L} \subseteq \mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$. A timed series $\mathcal{T} : T\Sigma^* \to \mathcal{K}$ is called $\mathcal{L}$-*definable* if there is a sentence $\varphi \in \mathcal{L}$ such that $[\![\varphi]\!] = \mathcal{T}$. The goal of this paper is to find a suitable fragment $\mathcal{L}$ of $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ such that $\mathcal{L}$-definable timed series precisely correspond to recognizable timed series over $\mathcal{K}$, Σ and $\mathcal{F}$, i.e., we want to generalize Theorem 1 to the weighted setting. It is not surprising that $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ does not constitute a suitable candidate for $\mathcal{L}$ since this is already not the case in the untimed setting [9]. In the next section, we explain the problems that occur when we do not restrict the logic and step by step develop solutions resulting in the logic $\mathrm{sRMSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ for which we are able to give the following Büchi-type theorem.

Theorem 2. *A timed series $\mathcal{T} : T\Sigma^* \to K$ is recognizable over $\mathcal{K}$, Σ and $\mathcal{F}$ if and only if $\mathcal{T}$ is definable by some sentence in $sRMSO_{er}(\mathcal{K}, \Sigma, \mathcal{F})$. The respective transformations can be done effectively provided that the operations of $\mathcal{K}$ and $\mathcal{F}$ are given effectively.*

3 From Logic to Automata

In this section, we want to prove the direction from right to left in Theorem 2 and show that for every formula φ of our weighted timed MSO logic, $[\![\varphi]\!]$ is a recognizable timed series. We do this similarly to the corresponding proof for the classical setting [27], i.e., by induction over the structure of the logic.

For the induction base, we show that for every atomic formula φ in $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ there is a WERA recognizing $[\![\varphi]\!]$. For φ of the form $P_a(x)$, $x = y$, $x < y$, $x \in X$ and its negations, this can be done as in the classical setting [27]. In Fig. 1, we give the WERA recognizing the timed series $[\![\varphi]\!]$ for φ being one of $\lhd_a(x) \sim c$, k and $C_\mu(x)$.

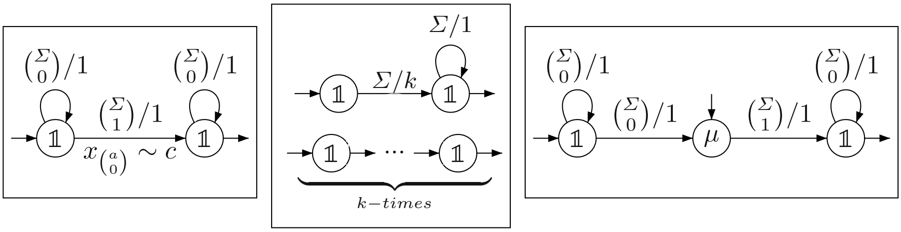


Fig. 1. WERA with behaviours $[\![\lhd_a(x) \sim c]\!]$, $[\![k]\!]$ and $[\![C_\mu(x)]\!]$

For the induction step, we need to show closure properties of recognizable timed series under the constructs of the logic. For disjunction and existential quantification, we can give proofs very similar to the classical case (see Thomas for the case of formal languages [26] or Droste and Gastin for the case of (untimed) series [9,10]). However, we will see that for the remaining operators of our logic, we cannot give easy extensions of the classical proofs.

First of all, in Sect. 1 we have seen that recognizable timed series in general are not closed under the Hadamard product. Since the semantics of conjunction is defined using the Hadamard product, this means that we have to restrict the usage of conjunction. More precisely, we either have to require that $\mathcal{F}$ is such that for all $f_1, f_2 \in \mathcal{F}$ we have $f_1 \odot f_2 \in \mathcal{F}$, or we have to formulate a syntactical restriction implying that whenever two formulas φ_1 and φ_2 are combined by a conjunction, then $[\![\varphi_1]\!]$ and $[\![\varphi_2]\!]$ are non-interfering.

Lemma 4. *Let $\varphi_1, \varphi_2 \in MSO_{er}(\mathcal{K}, \Sigma, \mathcal{F})$ such that $[\![\varphi_1]\!]$ and $[\![\varphi_2]\!]$ are recognizable. Assume that whenever φ_1 contains the subformula $C_{\mu_1}(x_1)$ and φ_2 contains $C_{\mu_2}(x_2)$, then x_1, x_2 are free in both φ_1 and φ_2, and either φ_1 or φ_2 is of the form $\psi \wedge \neg(x_1 = x_2)$ for some $\psi \in MSO_{er}(\mathcal{K}, \Sigma, \mathcal{F})$. Then $[\![\varphi_1]\!]$ and $[\![\varphi_2]\!]$ are non-interfering.*

We give the intuition behind this lemma via an example. Consider the formula $C_{\mu_1}(x_1) \wedge C_{\mu_2}(x_2)$ and let $\mathcal{A}_i$ be a WERA such that $\|\mathcal{A}_i\| = [\![C_{\mu_i}(x_i)]\!]$ for each $i = 1, 2$ (see Fig.2). We use s_1 (s_2, resp.) to denote the location in $\mathcal{A}_1$ ($\mathcal{A}_2$, resp.) with cost function μ_1 (μ_2), resp.). We want to enforce that in the product automaton of $\mathcal{A}_1$ and $\mathcal{A}_2$, from the pair (s_1, s_2) there is no run to a final location. This is the case if from s_1 and s_2 no common letter can be read. Observe that from s_1 (s_2, resp.) every outgoing edge is labeled with (a, σ) such that $\sigma(x_1) = 1$ ($\sigma(x_2) = 1$, resp.) for every $a \in \Sigma$. Hence, in the product automaton every edge from (s_1, s_2) must be labeled with a letter of the form (a, σ) such that $\sigma(x_1) = \sigma(x_2) = 1$ for every $a \in \Sigma$. By requiring x_1 and x_2 to refer to different positions in a timed word, we can exclude that there is an edge from (s_1, s_2) labeled with a letter of this form. This is done by conjoining the formula above with $\neg(x_1 = x_2)$.

Second, examples [10] show that unrestricted application of $\forall x.$ and $\forall X.$ do not preserve recognizability. For instance, let $\mathcal{K} = (\mathbb{N}, +, \cdot, 0, 1)$ be the semiring of the natural numbers and $\mathcal{F}$ be the family of constant functions. We consider the formula $\varphi = \forall y. \exists x. C_{\mathbb{1}}(x)$. Then we have $[\![\varphi]\!](w) = |w|^{|w|}$. However, this cannot be recognized by any WERA as this timed series grows too fast (see [9] for a detailed proof which can also be applied to the timed setting). Similar examples can be given for $\forall X$. Hence, we need to restrict both the usage of $\forall x.$ and $\forall X.$ in our logic. We adopt the approach of Droste and Gastin [10].

For dealing with $\forall X.$, the idea is to restrict the application of $\forall X.$ to so-called *syntactically unambiguous* formulas. These are formulas $\varphi \in \mathrm{MSO}_{\mathrm{er}}(\Sigma)$ such that - even though interpreted over a semiring - the semantics $[\![\varphi]\!]$ of φ always equals 0 or 1[1]. We define the set of syntactically unambiguous formulas φ^+ and φ^- for $\varphi \in \mathrm{MSO}_{\mathrm{er}}(\Sigma)$ inductively as follows:

1. If φ is of the form $P_a(x)$, $x < y$, $x = y$, $x \in X$, $\lhd_a(x) \sim c$, then $\varphi^+ = \varphi$ and $\varphi^- = \neg\varphi$.
2. If $\varphi = \neg\psi$ then $\varphi^+ = \psi^-$ and $\varphi^- = \psi^+$.
3. If $\varphi = \psi \vee \zeta$ then $\varphi^+ = \psi^+ \vee (\psi^- \wedge \zeta^+)$ and $\varphi^- = \psi^- \wedge \zeta^-$
4. If $\varphi = \exists x.\psi$ then $\varphi^+ = \exists x.\psi^+ \wedge \forall y.(y < x \wedge \psi(y))^-$ and $\varphi^- = \forall x.\psi^-$
5. If $\varphi = \exists X.\psi$ then $\varphi^+ = \exists X.\psi^+ \wedge \forall Y.(Y < X \wedge \psi(Y))^-$ and $\varphi^- = \forall X.\psi^-$

where $X < Y = \exists y. y \in Y \wedge \neg(y \in X) \wedge \forall z.[z < y \longrightarrow (z \in X \longleftrightarrow z \in Y)]^+$. Notice that for each $\varphi \in \mathrm{MSO}_{\mathrm{er}}(\Sigma)$ we have $[\![\varphi^+]\!] = 1_{L(\varphi)}$ and $[\![\varphi^-]\!] = 1_{L(\neg\varphi)}$. Thus

[1] Recall that every $\mathrm{MSO}_{\mathrm{er}}(\Sigma)$-formula can also be seen as an $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$-formula and may have a semantics different from 0 or 1; see e.g. Ex.2.

the semantics of syntactically unambiguous formulas are recognizable by Theorem 1 and Lemma 1. Moreover, if φ is syntactically unambiguous, one can easily see that also $\forall X.\varphi$ is syntactically unambiguous and thus $[\![\forall X.\varphi]\!]$ is recognizable.

Next, we explain how to deal with $\forall x$. The approach used by Droste and Gastin [10] is to restrict the subformula φ in $\forall x.\varphi$ to so-called *almost unambiguous* formulas. Formulas of this kind can be transformed into equivalent formulas of the form $\bigvee_{1\leq i\leq n} k_i \wedge \psi_i^+$ for some $n \in \mathbb{N}$, $k_i \in K$ and syntactically unambiguous formulas ψ_i^+ for each $i \in \{1,...,n\}$. One can easily see that the series corresponding to the semantics of such a formula has a finite image. Moreover, closure properties of recognizable series under sum, Hadamard- and scalar products can be used to prove that the semantics of such a formula is recognizable by a weighted automaton. Finally, this particular form of the formula is the base of an efficient construction of a weighted automaton recognizing $[\![\forall x.\varphi]\!]$. Here, we use a very similar approach. However, we have to redefine the notion of almost unambiguous formulas a bit in order to include subformulas of the form $C_\mu(x)$.

Let x be a first-order variable. We say that a formula φ is *almost unambiguous over* x if it is in the disjunctive and conjunctive closure of syntactically unambiguous formulas, constants $k \in K$ and formulas $C_\mu(x)$ (for $\mu \in \mathcal{F}$), such that $C_\mu(x)$ may appear at most once in every subformula of φ of the form $\varphi_1 \wedge \varphi_2$. Using similar methods as in [10], one can show that every almost unambiguous formala can be transformed into an equivalent formula of the form $\bigvee_{1\leq i\leq n} C_{\mu_i}(x) \wedge k_i \wedge \psi_i^+$ for some $n \in \mathbb{N}$, $k_i \in K$, $\mu_i \in \mathcal{F}$ and $\psi_i \in \mathrm{MSO_{er}}(\Sigma)$ for every $i \in \{1,...,n\}$. Clearly, the semantics of formulas of this form is not guaranteed to have a finite image. As a counter example consider for instance the case where $\mathcal{F}$ is the family of linear functions. However, using Lemmas 2 and 3 as well as Theorem 1, one can prove that the semantics of every formula of this form (and thus of every almost unambiguous formula over x) is recognizable. So now assume that φ is almost unambiguous over x. The main challenge of this paper was to prove that $[\![\forall x.\varphi]\!]$ is recognizable. We were able to adapt the proof proposed by Droste and Gastin to the timed setting by applying an additional normalization technique to solve problems having their origin in formulas of the form $C_\mu(x)$. The proof is rather technical and omitted here; for the details see the full length version of this paper [23].

Finally, we define the fragment of $\mathrm{MSO_{er}}(\mathcal{K},\Sigma,\mathcal{F})$ used in Theorem 2. A formula $\varphi \in \mathrm{MSO_{er}}(\mathcal{K},\Sigma,\mathcal{F})$ is called *syntactically restricted* if it satisfies the following conditions:

1. Whenever φ contains a conjunction $\varphi_1 \wedge \varphi_2$ as subformula, φ_1 contains the subformula $C_{\mu_1}(x_1)$ and φ_2 contains $C_{\mu_2}(x_2)$, then x_1, x_2 are free in both φ_1 and φ_2, and either φ_1 or φ_2 is of the form $\psi \wedge \neg(x_1 = x_2)$ for some $\psi \in \mathrm{MSO_{er}}(\mathcal{K},\Sigma,\mathcal{F})$.
2. Whenever φ contains $\forall x.\psi$ as a subformula, then ψ is an almost unambiguous formula over x.
3. Whenever φ contains $\forall X.\psi$ as a subformula, then ψ is a syntactically unambiguous formula.

We let $\mathrm{sRMSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ denote the set of all syntactically restricted formulas of $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$. Notice that each of these conditions can be checked for in easy syntax tests. Hence, the logic $\mathrm{sRMSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ is a *decidable* fragment, i.e., for each formula in $\mathrm{MSO}_{\mathrm{er}}(\mathcal{K}, \Sigma, \mathcal{F})$ we can decide whether it is syntactically restricted or not.

We want to give some final remarks on the correctness of the proof methods described above. Although not explicitly mentioned in the individual steps, we make use of renaming operations in the proofs for closure under the constructs of our logic. For instance, we adopt the classical proof method for showing that the application of $\exists x.$ preserves the recognizability of the semantics of a formula φ with $\mathrm{Free}(\varphi) = \mathcal{V}$ by using a renaming $\pi : \Sigma_{\mathcal{V}} \to \Sigma_{\mathcal{V}\setminus\{x\}}$ which erases the x-row (see e.g. [26,10]). However, it is well-known that recognizable timed languages are not closed under renaming [1]. We solve this problem using an approach proposed by D'Souza [14] and consider so-called *quasi*-WERA. Timed languages recognizable by quasi-WERA share the same closure properties as recognizable timed languages, but additionally are closed under so-called *valid* renamings [14]. So, in the inductive proof described above, we actually show that the semantics of every formula in our logic is recognizable by a quasi-WERA rather than a WERA. Since quasi-WERA-recognizable timed series form a strict subclass of recognizable timed series, we get the final implication. For the sake of simplicity, we only mention this here; the correct proof can be found in [23].

4 From Automata to Logic

For the implication from left to right in Theorem 2, we extend the proof proposed by Droste and Gastin to the timed setting, briefly explained in the following. Let $\mathcal{A} = (S, S_0, S, E, C)$ be a WERA. We choose an enumeration $(e_1, ..., e_m)$ of E with $m = |E|$ and assume $e_i = (s_i, a_i, \phi_i, s_i')$. We define a syntactically unambiguous formula $\psi(X_1, ..., X_m)$ without any second-order quantifiers describing the successful runs of $\mathcal{A}$ (where for each $i \in \{1, ..., m\}$, X_i stands for the edge e_i). This can be done similarly to the classical setting [26]. The guards of the edges in E can be defined by a formula of the form $\forall x. \bigwedge_{1\le i\le m} \big(x \in X_i \xrightarrow{+} \bigwedge_{a\in\Sigma} (\bigwedge_{(x_a\sim c)\in\phi_i} \triangleleft_a(x) \sim c)\big)$ where $\varphi \xrightarrow{+} \psi$ is an abbreviation for $\varphi^- \vee (\varphi^+ \wedge \psi^+)$. Then, for every non-empty timed word $(\bar{a}, \bar{t})$ and valid $(\{X_1, ..., X_m\}, (\bar{a}, \bar{t}))$-assignment σ, we have $[\![\psi(X_1, ..., X_m)]\!]((\bar{a}, \sigma), \bar{t}) = 1$, if there is a successful run of $\mathcal{A}$ on $(\bar{a}, \bar{t})$, and $[\![\psi(X_1, ..., X_m)]\!]((\bar{a}, \sigma), \bar{t}) = 0$, otherwise. Notice that we need to use syntactically unambiguous formulas here in order to avoid getting weights different from 1 or 0. Now, we "add weights" to ψ to obtain a formula ξ whose semantics corresponds to the running weight of a successful run of $\mathcal{A}$ on $(\bar{a}, \bar{t})$ as follows:

$$\xi = \psi \wedge \bigwedge_{e_i\in E} \forall x. \big(\neg(x \in X_i) \vee [x \in X_i \wedge C_{\mu_{s_i}}(x) \wedge C_{\mathcal{E}}(e_i)]\big)\,.$$

For the empty timed word ε, we define a formula $\varphi = (\|\mathcal{A}\|, \varepsilon) \wedge \forall x. \neg(x \le x)$. Finally, we let $\zeta = \exists X_1 ... \exists X_m.(\xi \vee \varphi)$, and we obtain $[\![\zeta]\!] = \|\mathcal{A}\|$. Hence, we have shown the second implication, which finishes the proof of Theorem 2.

5 Conclusion

We have presented a weighted timed MSO logic, which is - at least to our knowledge - the first MSO logic allowing for the description of both timed and quantitative properties. On the one hand, we provide the real-time-community with a new tool, because sometimes it may be easier to specify properties in terms of logic rather than by automata devices. On the other hand, the coincidence between recognizable and definable timed series, together with a previous work on WERA concerning a Kleene-Schützenberger Theorem [22], shows the robustness of the notion of WERA-recognizable timed series, as they can equivalently be characterized in terms of automata, logics and rational operations. The same applies to timed series recognizable by weighted timed automata, for which we were successful in adapting the proofs presented in this paper using the relative distance logic $\mathcal{L}\overleftarrow{\mathrm{d}}$ introduced by Wilke and his results concerning timed languages with bounded variability [28,24]. Notice that our result generalizes corresponding results on ERA-recognizable languages as well as formal power series [14,10]. Also, we have stated conditions for closure of recognizable timed series under the Hadamard product, which corresponds to the intersection operation in the unweighted setting.

Acknowledgement

I would like to thank Manfred Droste and Christian Mathissen for their helpful discussions and comments.

References

1. Alur, R., Fix, L., Henzinger, T.A.: Event-clock automata: A determinizable class of timed automata. Theor. Comput. Sci. 211(1-2), 253–273 (1999)
2. Alur, R., La Torre, S., Pappas, G.J.: Optimal paths in weighted timed automata. Theoretical Computer Science 318, 297–322 (2004)
3. Behrmann, G., Fehnker, A., Hune, T., Larsen, K., Pettersson, P., Romijn, J., Vaandrager, F.: Minimum-cost reachability for priced timed automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 147–161. Springer, Heidelberg (2001)
4. Bouyer, P., Brihaye, T., Bruyère, V., Raskin, J.-F.: On the optimal reachability problem on weighted timed automata. Formal Methods in System Design 31(2), 135–175 (2007)
5. Bouyer, P., Brihaye, T., Markey, N.: Improved undecidability results on weighted timed automata. Inf. Process. Lett. 98(5), 188–194 (2006)
6. Büchi, J.R.: On a decision method in restricted second order arithmetics. In: Nagel, E., et al. (eds.) Proc. Intern. Congress on Logic, Methodology and Philosophy of Sciences, pp. 1–11. Stanford University Press, Stanford (1960)
7. Dima, C.: Kleene theorems for event-clock automata. In: Ciobanu, G., Păun, G. (eds.) FCT 1999. LNCS, vol. 1684, pp. 215–225. Springer, Heidelberg (1999)
8. Droste, M., Gastin, P.: Weighted automata and weighted logics. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 513–525. Springer, Heidelberg (2005)
9. Droste, M., Gastin, P.: Weighted automata and weighted logics. Theor. Comput. Sci. 380(1-2), 69–86 (2007)

10. Droste, M., Gastin, P.: Weighted automata and weighted logics. In: Droste, M., Kuich, W., Vogler, H. (eds.) Handbook of Weighted Automata, pp. 173–206. Springer, Heidelberg (to appear, 2009)
11. Droste, M., Quaas, K.: A Kleene-Schützenberger Theorem for Weighted Timed Automata. In: Amadio, R. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 142–156. Springer, Heidelberg (2008)
12. Droste, M., Rahonis, G.: Weighted automata and weighted logics on infinite words. In: H. Ibarra, O., Dang, Z. (eds.) DLT 2006. LNCS, vol. 4036, pp. 49–58. Springer, Heidelberg (2006)
13. Droste, M., Vogler, H.: Weighted tree automata and weighted logics. Theor. Comput. Sci. 366(3), 228–247 (2006)
14. D'Souza, D.: A logical characterisation of event recording automata. In: Joseph, M. (ed.) FTRTFT 2000. LNCS, vol. 1926, pp. 240–251. Springer, Heidelberg (2000)
15. Elgot, C.C.: Decision problems of finite automata design and related arithmetics. Trans. Am. Math. Soc. 98, 21–51 (1961)
16. Grinchtein, O., Jonsson, B., Pettersson, P.: Inference of event-recording automata using timed decision trees. In: Baier, C., Hermanns, H. (eds.) CONCUR 2006. LNCS, vol. 4137, pp. 435–449. Springer, Heidelberg (2006)
17. Henzinger, T.A., Raskin, J.-F., Schobbens, P.-Y.: The regular real-time languages. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 580–591. Springer, Heidelberg (1998)
18. Mathissen, C.: Definable transductions and weighted logics for texts. In: Harju, T., Karhumäki, J., Lepistö, A. (eds.) DLT 2007. LNCS, vol. 4588, pp. 324–336. Springer, Heidelberg (2007)
19. Mathissen, C.: Weighted logics for nested words and algebraic formal power series. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 221–232. Springer, Heidelberg (2008)
20. Mäurer, I.: Weighted picture automata and weighted logics. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 313–324. Springer, Heidelberg (2006)
21. Meinecke, I.: Weighted logics for traces. In: Grigoriev, D., Harrison, J., Hirsch, E.A. (eds.) CSR 2006. LNCS, vol. 3967, pp. 235–246. Springer, Heidelberg (2006)
22. Quaas, K.: A Kleene-Schützenberger-Theorem for Weighted Event-Clock-Automata (2008), http://www.informatik.uni-leipzig.de/~quaas/weca_ks.pdf
23. Quaas, K.: A logical characterization for weighted event-recording automata (2009), http://www.informatik.uni-leipzig.de/~quaas/wera_logic.pdf
24. Quaas, K.: A logical characterization for weighted timed automata (2009), http://www.informatik.uni-leipzig.de/~quaas/wta_logic.pdf
25. Raskin, J.-F., Schobbens, P.-Y.: State clock logic: A decidable real-time logic. In: Maler, O. (ed.) HART 1997. LNCS, vol. 1201, pp. 33–47. Springer, Heidelberg (1997)
26. Thomas, W.: Languages, automata and logic. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 389–485. Springer, Heidelberg (1997)
27. Thomas, W.: Automata on infinite objects. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, Formal Models and Sematics (B), vol. B, pp. 133–192. Elsevier, Amsterdam (1990)
28. Wilke, T.: Specifying Timed State Sequences in Powerful Decidable Logics and Timed Automata. In: Langmaack, H., de Roever, W.-P., Vytopil, J. (eds.) FTRTFT 1994 and ProCoS 1994. LNCS, vol. 863, pp. 694–715. Springer, Heidelberg (1994)

Balanced Words Having Simple Burrows-Wheeler Transform

Antonio Restivo and Giovanna Rosone

University of Palermo, Dipartimento di Matematica ed Applicazioni,
Via Archirafi 34, 90123 Palermo, Italy
{restivo,giovanna}@math.unipa.it

Abstract. The investigation of the "clustering effect" of the Burrows-Wheeler transform (BWT) leads to study the words having *simple BWT*, i.e. words w over an ordered alphabet $A = \{a_1, a_2, \ldots, a_k\}$, with $a_1 < a_2 < \ldots < a_k$, such that $bwt(w)$ is of the form $a_k^{n_k} a_{k-1}^{n_{k-1}} \cdots a_1^{n_1}$, for some non-negative integers $n_1, n_2, \ldots, n_k$. We remark that, in the case of binary alphabets, there is an equivalence between words having simple BWT, the family of (circular) balanced words and the conjugates of standard words. In the case of alphabets of size greater than two, there is no more equivalence between these notions. As a main result of this paper we prove that, under assumption of balancing, the following three conditions on a word w are equivalent: *i)* w has simple BWT, *ii)* w is a circularly rich word, and *iii)* w is a conjugate of a finite epistandard word.

1 Introduction

Michael Burrows and David Wheeler introduced in 1994 (cf. [5]) a reversible transformation on words that turns out to be an extremely useful tool for textual data compression. Compression algorithms based on the Burrows-Wheeler Transform (BWT) take advantage of the fact that the word output of BWT shows a local similarity (occurrences of a given symbol tend to occur in clusters) and then turns out to be highly compressible. In order to investigate such a "clustering effect" of BWT it is interesting to consider the extremal case when all occurrences of each letter make up a factor of the transform, i.e. the transform produces a perfect clustering. Perfect clustering corresponds indeed to optimal performances of some BWT-based compression algorithms.

So we consider the set $\mathcal{S}$ of the words w for which the BWT produces a perfect clustering, i.e., the set of all words w over an ordered alphabet $A = \{a_1, a_2, \ldots, a_k\}$, with $a_1 < a_2 < \ldots < a_k$, such that $bwt(w)$ is of the form $a_k^{n_k} a_{k-1}^{n_{k-1}} \cdots a_2^{n_2} a_1^{n_1}$, for some non-negative integers $n_1, n_2, \ldots, n_k$. In the sequel we refer to the elements of $\mathcal{S}$ as the words having *simple Burrows-Wheeler transform*. Since two words that are conjugates have the same BWT, the set $\mathcal{S}$ is invariant under conjugation, and then the notions and the results considered in this paper could be perhaps better described in terms of *circular words*: a circular word corresponds indeed to a conjugacy class of a word. Remark further that

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 431–442, 2009.

the study of circular words is closely related to that of infinite periodic words. In the case of a binary alphabet a complete characterization of the set $\mathcal{S}$ has been given in [17], where it is proved (cf. also [11]) that the following three conditions on a word w are equivalent: *i)* w is in $\mathcal{S}$, *ii)* w is a conjugate of a standard word, *iii)* w is (circularly) balanced. In the case of a three letters alphabet a constructive characterization of the set $\mathcal{S}$ has been given in [23].

The set $\mathcal{S}$ and the notion of circularly balanced word (related to that of balanced periodic infinite word) are well defined in an arbitrary finite alphabet. We note that the notion of balanced periodic words over an alphabet of size larger than two also appears in the statement of the Fraenkel conjecture (cf. [8]). As a direct consequence of a result of Graham, one can prove that balanced sequences on a set of letters having different frequencies must be periodic (cf. [24]). The problem of characterizing balanced words over any alphabet has been developed by Altman, Gaujal and Hordijk in [1] in the field of optimal routing in queuing networks. Some combinatorial properties of balanced circular words have been recently investigated in [16].

The notion of standard word is closely related to that of Sturmian sequence. Numerous generalizations of Sturmian sequences have been introduced for an alphabet with more than 2 letters. Among them, one natural generalization are the *episturmian sequences* that are defined by using the palindromic closure property of Sturmian sequences (cf. [7]). Here we consider some special prefixes of episturmian sequences, that we call *finite epistandard* words: in the case of a binary alphabet they correspond to the finite standard words.

Another notion, related to the previous ones, that has been recently introduced is that of *rich word*: a word is rich if it contains the maximal number of distinct palindromic factor (cf. [10]). Rich words appear in many different contexts: they include sturmian and episturmian words, and also other families of words known in literature.

We remark that, in the case of alphabets of size greater than two, there is no more equivalence between the set $\mathcal{S}$, the family of (circular) balanced words and the conjugates of epistandard words. However, as a main result of this paper we prove that, under assumption of balancing, the following three conditions on a word w are equivalent *i)* w is in $\mathcal{S}$, *ii)* w is a circularly rich word, and *iii)* w is a conjugate of a finite epistandard word. Apart its interest for the study of optimal performances of BWT-based compression algorithms, our result is a contribution to combinatorics of episturmian sequences and could provide new insights on Fraenkel conjecture.

2 Preliminaries

Let $A = \{a_1, a_2, \ldots, a_k\}$ be a finite ordered alphabet (with $a_1 < a_2 < \ldots < a_k$). We denote by A^* the set of words over A. Given a finite word $w = b_1 b_2 \cdots b_n \in A^*$ with each $b_i \in A$, the length of w, denoted $|w|$, is equal to n. By convention, the empty word ε is the unique word of length 0. We denote by $\tilde{w}$ the reversal of w, given by $\tilde{w} = b_n \cdots b_2 b_1$. If w is a word that has the property of reading the same in either direction, i.e. if $w = \tilde{w}$, then w is called a *palindrome*. A word has the *two palindrome property* if it can be written as uv where u and v are palindromes or empty.

We say that two words $x, y \in A^*$ are *conjugate*, if $x = uv$ and $y = vu$ for some $u, v \in A^*$. Conjugacy between words is an equivalence relation over A^*. The *conjugacy class* $[x]$ of $x \in A^n$ is the set of all words $b_i b_{i+1} \cdots b_n b_1 \cdots b_{i-1}$, for $1 \leq i \leq n$. A conjugacy class can also be represented as a circular word. Hence in what follows we will use "circular word" and "conjugacy class" as synonyms.

A word $w \in A^*$ is *primitive* if $w = u^h$ implies $w = u$ and $h = 1$. Recall that (cf. [14]) every word $u \in A^*$ can be written in a unique way as a power of a primitive word, i.e. there exists a unique primitive word w, called the *root* of u, and a unique integer k such that $u = w^k$.

If u is a word in A^*, we denote by u^ω the infinite word obtained by infinitely iterating u, i.e. $u^\omega = uuuuu \ldots$. A word $w \in A^\omega$ is *ultimately periodic* of period $n \in \mathbb{N}$ if $w_i = w_{i+n} \forall i \geq l$ and $l \in \mathbb{N}$. If $l = 1$, then w is *purely periodic*. An infinite word that is not ultimately periodic is said to be *aperiodic*.

A word $v \in A^*$ is said to be a *factor* (resp. a *prefix*, resp. a *suffix*) of a word $w \in A^*$ if there exist words $x, y \in A^*$ such that $w = xvy$ (resp. $w = vy$, resp. $w = xv$). A factor (resp. the prefix, resp. the suffix) is *proper* if $xy \neq \varepsilon$ (resp. $y \neq \varepsilon$, resp. $x \neq \varepsilon$). A factor u of a finite or infinite word w is said to be *left special* (resp. *right special*) in w if there exist at least two distinct letters a, b such that au and bu (resp. ua, ub) are factors of w. For any finite or infinite word w, $F(w)$ denotes the set of all its factors. We say that $F(w)$ is closed under reversal if for any $u \in F(w)$, $\tilde{u} \in F(w)$. Moreover, if w is infinite, we denote by $Ult(w)$ the set of all letters occurring infinitely often in w. A factor of an infinite word x is *recurrent* in x if it occurs infinitely often in x, and x itself is said to be *recurrent* if all of its factors are recurrent in it. Given two palindromes w, v, we say that v is a *central factor* of w if $w = uv\tilde{u}$ for some $u \in A^*$. The *palindromic right-closure* $w^{(+)}$ of a finite word w is the (unique) shortest palindrome having w as a prefix (see [6]). The *iterated palindromic closure* function [12], denoted by Pal, is defined recursively as follows. Set $Pal(\varepsilon) = \varepsilon$ and, for any word w and letter x, define $Pal(wx) = (Pal(w)x)^{(+)}$.

A finite or infinite word is *balanced* if, for any two of its factors u, v with $|u| = |v|$, we have

$$||u|_a - |v|_a| \leq 1 \quad \text{for any letter } a,$$

i.e. the number of a's in each of u and v differs by at most 1. A finite word is *circularly balanced* if all its conjugates are balanced. Let denote by $\mathcal{B}$ the set of the circularly balanced finite words, so $u \in \mathcal{B}$ if and only if u^ω is balanced.

3 The Burrows-Wheeler Transform

The Burrows-Wheeler transform was introduced in 1994 by Burrows and Wheeler [5] and represents an extremely useful tool for textual lossless data compression. The idea is to apply a reversible transformation in order to produce a permutation $bwt(w)$ of an input sequence w, defined over an ordered alphabet A, so that the sequence becomes easier to compress. Actually the transformation tends to group characters together so that the probability of finding a character close to another instance of the same character is substantially increased. BWT transforms a word w by lexicographically sorting all the n conjugates of w and extracting the last character of each conjugate. The sequence $bwt(w)$ consists of the concatenation of these characters. We denote by M the matrix which consists of all conjugates of w lexicographically sorted. In the sequel we will refer to M as the “Burrows-Wheeler matrix” of w. Moreover the transformation computes the index I, that is the row of M containing the original sequence. For instance, suppose we want to compute $bwt(w)$ where $w = abraca$. Consider the Burrows-Wheeler matrix M in Figure 1. The last column L of the matrix M represents $bwt(w) = caraab$ and $I = 2$ since the original sequence w appears in row 2.

The transform is reversible in the sense that, given $bwt(w)$ and the index I, it is possible to recover the original string w. Notice that if we except the index, all the mutual conjugate words have the same Burrows-Wheeler Transform. Actually the index has the only aim of denoting one representative in the conjugacy class. However this index is not necessary for the construction of the matrix M from the last column L.

The following lemma shows the relation of the BWT of a word and the BWT of a power of the word itself (cf. [17]):

Lemma 1. *Let $u, v \in A^*$ and let $bwt(v) = b_1 b_2 \cdots b_n$, $b_i \in A$, for $i = 1, \ldots, n$.*

1. *If $u = v^d$ then $bwt(u) = b_1^d b_2^d \cdots b_n^d$.*
2. *If $bwt(u) = b_1^d b_2^d \cdots b_n^d$ then there exists a conjugate u' of u such that $u' = v^d$.*

Now, we define the set $\mathcal{S}$, as the set of the words w over a totally ordered alphabet $A = \{a_1, a_2, \ldots, a_k\}$, with $a_1 < a_2 < \ldots < a_k$, for which

$$bwt(w) = a_k^{n_k} a_{k-1}^{n_{k-1}} \cdots a_2^{n_2} a_1^{n_1}$$

for some non-negative integers $n_1, n_2, \ldots, n_k$.

Clearly, by Lemma 1 a word is in $\mathcal{S}$ if and only if its root is in $\mathcal{S}$.

We recall that in the case $|A| = 2$, the set $\mathcal{S}$ has been characterized in [17] where it is proved that the set $\mathcal{S}$ coincides with the set of power of conjugates of standard words. In the case $|A| = 3$ a constructive characterization of the set $\mathcal{S}$ has been given by Simpson and Puglisi in [23] and the case of a generic alphabet has been also studied in [22].

The following result, proved in [22], provides a characterization of the words in $\mathcal{S}$ in terms of the Burrows-Wheeler matrix M. We denote by R the matrix obtained from M by a rotation of 180°. Notice that the rows of R correspond to the conjugates of $\tilde{w}$.

Theorem 1. *A word $w \in \mathcal{S}$ if and only if $M = R$.*

For instance, in Fig. 1, M and R are distinct and the word $w = abraca$ does not belong to $\mathcal{S}$. We mention that a result equivalent of Theorem 1 has been obtained, with a different proof, by Simpson and Puglisi [23, Theorem 4.3]. They also derive the following corollary (cf. [23, Corollary 4.4]).

Corollary 1. *Each conjugate of $w \in \mathcal{S}$ has the two palindrome property.*

$$\begin{array}{ccccccccccccc}
 & & & M & & & \qquad & & & R & & & \\
a & a & b & r & a & c & & b & a & a & c & a & r \\
a & b & r & a & c & a & & a & r & b & a & a & c \\
a & c & a & a & b & r & & a & a & c & a & r & b \\
b & r & a & c & a & a & & r & b & a & a & c & a \\
c & a & a & b & r & a & & a & c & a & r & b & a \\
r & a & c & a & a & b & & c & a & r & b & a & a
\end{array}$$

Fig. 1. The matrix M and R of the sequence $w = abraca$

4 The Case of Two Letters Alphabets

In the case of two letters alphabet the elements of $\mathcal{S}$ are closely related to Sturmian words, that were introduced by Morse and Hedlund (cf. [18]). Sturmian words can be defined in several different but equivalent ways (cf. [15, Chapter 2]). Some definitions are "combinatorial" and others of "geometrical" nature. With regard to the first type of definition a Sturmian word is a binary infinite word which is not ultimately periodic, of minimal complexity. Alternatively Sturmian words can be defined as the balanced infinite binary sequences. From the "geometrical" point of view , the Sturmian words code discrete lines. In particular, a Sturmian word can be defined by considering the intersections with a squared-lattice of a semiline having a slope which is an irrational number. A vertical intersection is denoted by the letter a, a horizontal intersection by b and the intersection with a corner by ab or ba (cf. [15,17]). If the semiline starts from the origin the corresponding Sturmian words is called *characteristic*. Characteristic Sturmian words can be constructed by a family of finite words called *standard words*, in the sense that every characteristic word is the limit of a sequence of standard words (cf. [15]).

Let $d_1, d_2, \ldots, d_n, \ldots$ be a sequence of natural numbers, with $d_i \geq 0$ and $d_i > 0$ for $i = 2, \ldots, n, \ldots$. Consider the following sequence $\{s_n\}_{n\geq 0}$ of words over the binary alphabet $\{a, b\}$: $s_0 = b$, $s_1 = a$, and $s_{n+1} = s_n^{d_n} s_{n-1}$ for $n \geq 1$. The sequence $\{s_n\}_{n\geq 0}$ converges to a limit s that is a *characteristic Sturmian* word. Moreover any characteristic Sturmian word is obtained in this way. The sequence $\{s_n\}_{n\geq 0}$ is called the approximating sequence of s and $(d_1, d_2, \ldots, d_n, \ldots)$ is the *directive sequence* of s. Each finite word s_n in the sequence is called a *standard* word. It is univocally determined by the (finite) directive sequence $(d_1, d_2, \ldots, d_{n-1})$. The following theorem proved in [17] (cf. also [11]) gives a complete characterization of words over a binary alphabet having simple Burrows-Wheeler transform.

Theorem 2. *Let v a primitive word over the alphabet $A = \{a, b\}$. Then following conditions are equivalent:*

1. *$bwt(v) = b^p a^q$ with $gcd(p, q) = 1$;*
2. *v is a conjugate of a standard word;*
3. *v is circularly balanced.*

A characterization of words having simple Burrows-Wheeler transform in the case of three letters alphabet has been given in [23]. Remark that for alphabet of cardinality greater than two, the equivalence of 1 and 2 is no longer true. Indeed there exist circularly balanced words that do not have simple BWT, for instance $v = ababc$ (in fact $bwt(v) = cbaab$) and there exist unbalanced words having simple BWT, for instance $u = acacbbc$ (in fact $bwt(u) = cccbbaa$). Moreover, in order to study the relationship between the conditions 1, 2 and 3 in the case of larger alphabets, we need to extend the notion of (finite) standard word.

5 Episturmian Words

A first generalization of *Sturmian words* to an arbitrary alphabet is the family of *Arnoux-Rauzy sequences* (cf. [21,2]). Another generalization of Sturmian sequences, which also is a slight generalization of Arnoux-Rauzy sequences, is the set of infinite episturmian sequences. These sequences are not necessarily balanced, nor are they necessarily aperiodic (cf. [7]). An infinite word $t \in A^{\omega}$ is *episturmian* if $F(t)$ is closed under reversal and t has at most one right (or equivalently left) special factor of each length. Moreover, an episturmian word is *standard* if all of its left special factors are prefixes of it. Sturmian words are exactly the aperiodic episturmian words over a 2-letter alphabet. For a recent survey on the theory of episturmian words see [9].

Let us recall the definition given by Droubay, Justin and Pirillo (cf. [7]).

Definition 1. *An infinite sequence s is* standard episturmian *if it satisfies one of the following equivalent conditions:*

i) For every prefix u of s, $u^{(+)}$ is also prefix of s.
ii) Every leftmost occurrence of a palindrome in s is a central factor of a palindromic prefix of s.
iii) There exists an infinite sequence $u_1 = \varepsilon, u_2, u_3, \dots$ of palindromes and an infinite sequence $\Delta(s) = x_1 x_2 \cdots$, $x_i \in A$, such that $u_{n+1} = (u_n x_n)^{(+)}$ for all $n \geq 1$ and that all the u_n are prefixes of s.

$\Delta(s)$ is called the *directive sequence* of the standard episturmian sequence s.

Remark 1. An episturmian sequence s is periodic if and only if $|Ult(\Delta(s))| = 1$ (see [13, Proposition 2.9]).

Contrary to the case of two letters alphabet, a standard episturmian sequence over an alphabet of size greater than two is not in general balanced. In [20], Paquin and Vuillon characterized the balanced episturmian words by classifying these words into the three families, which we recall in the following theorem.

Theorem 3. *Any balanced standard episturmian sequence s over an alphabet $A = \{a_1, a_2, \dots, a_k\}$, $k \geq 3$, has a directive sequence, up to a letter permutation, in one of the three following families of sequences:*

i) $\Delta(s) = a_1^m a_k a_{k-1} \dots a_3 (a_2)^\omega$ *where* $k \geq 3$ *and* $m \geq 1$*;*
ii) $\Delta(s) = a_1 a_k a_{k-1} \dots a_{k-\ell} a_1 a_{k-\ell-1} a_{k-\ell-2} \dots a_3 (a_2)^\omega$*, where* $0 \leq \ell \leq k-4$ *and* $k \geq 4$*;*
iii) $\Delta(s) = a_1 a_k a_{k-1} \dots a_2 (a_1)^\omega$ *where* $k \geq 3$*.*

As a direct consequence of Remark 1 and Theorem 3, one can prove the following corollary.

Corollary 2. *Any balanced standard episturmian sequence on 3 or more letters is periodic.*

5.1 Finite Epistandard Words

A standard episturmian sequence s can also be obtained by the *Rauzy rules* (see [7, Theorem 8]), where if $\Delta(s) = a_{i_1} a_{i_2} a_{i_3} \cdots$ then the sequence of the labels of the applied Rauzy rules is $i_1, i_2, i_3, \dots$. We recall that a sequence $(R_n)_{n \in \mathbb{N}}$ of Rauzy k-tuples $R_n = (A_n^{(1)}, A_n^{(2)}, \dots, A_n^{(k)})$ is defined as follows: $R_0 = (a_1, a_2, \dots, a_k)$, R_{n+1} is obtained from R_n by applying one of the Rauzy rules, labelled $1, 2, \dots, k$, with the rule $i \in [1,k]$ defined by

$$A_{n+1}^{(i)} = A_n^{(i)}$$
$$A_{n+1}^{(j)} = A_n^{(i)} A_n^{(j)} \text{ for } j \in [1,k] \backslash \{i\}.$$

There exists a unique (infinite) sequence u such that every prefix of u is a prefix of infinitely many of the $A_n^{(q)}$, $n \in \mathbb{N}$, $q \in [1,k]$. So any Rauzy sequence $(R_n)_{n \in \mathbb{N}}$ defines an infinite standard episturmian sequence.

Definition 2. *A word* $w \in A^*$ *is called* finite epistandard *if it is the element of a k-tuples* R_n*, for some* $n \geq 1$*.*

It is easy to see that, in the case of binary alphabets, the notion of finite epistandard word corresponds to the notion of (finite) standard word given in Section 4. We observe that the notion of finite epistandard word is also connected to the notion of epichristoffel word defined in [19].

Let us remark that a finite epistandard word is primitive. In the sequel we will denote by $\mathcal{EP}$ the set of words that are a power of a conjugate of a finite epistandard word. From previous construction and Remark 1 follows lemma stated below.

Lemma 2. *The sequence* t^ω *is a standard episturmian sequence if and only if* t *is a finite epistandard word.*

Theorem 3 can be restated in terms of finite epistandard words as follows:

Theorem 4. *Any balanced standard episturmian sequence* s *over an alphabet with 3 or more letters is of the form* $s = t^\omega$*, where* t *is a finite epistandard word that belongs to one of the following three families (up to letter permutation):*

i) $t = pa_2$*, with* $p = Pal(a_1^m a_k a_{k-1} \cdots a_3)$*, where* $k \geq 3$ *and* $m \geq 1$*;*

ii) $t = pa_2$, *with* $p = Pal(a_1 a_k a_{k-1} \cdots a_{k-\ell} a_1 a_{k-\ell-1} a_{k-\ell-2} \cdots a_3)$, *where* $0 \leq \ell \leq k-4$ *and* $k \geq 4$*;*
iii) $t = Pal(a_1 a_k a_{k-1} \cdots a_2)$, *where* $k \geq 3$.

Proof. The three families of balanced finite epistandard word t are obtained by applying one of the following Rauzy rules, corresponding to the directive sequences in Theorem 3:

i) $1^m k(k-1)\ldots 32$, where $k \geq 3$ and $m \geq 1$;
ii) $1k(k-1)\ldots(k-\ell)1(k-\ell-1)(k-\ell-2)\ldots 32$, where $0 \leq \ell \leq k-4$ and $k \geq 4$;
iii) $1k(k-1)\ldots 21$, where $k \geq 3$.

t is the word $A_r^{(j)}$, where in the case *i)* r is $m+k-1$ and j is equals to 2, in the case *ii)* r is $k+1$ and j is equals to 2 and in the case *iii)* r is $k+1$ and j is equals to 1. The infinite standard episturmian sequence $s = t^\omega$ is obtained by applying infinitely often the last rules. □

We observe that the sequences of the last family of Theorem 4 correspond to the *Fraenkel's sequence. Fraenkel's conjecture* [8] is a well-known problem related to balance that arose in a number-theoretic context. Fraenkel conjectured that, for a fixed $k \geq 3$, there is only one covering of $\mathbb{Z}$ by k *Beatty sequences* of the form $(\lfloor \alpha n + \beta \rfloor)_{n \geq 1}$, where α, β are real numbers. A combinatorial interpretation of this conjecture may be stated as follows (taken from [20]). Over a k-letter alphabet with $k \geq 3$, there is only one recurrent balanced infinite word, up to letter permutation and shifts, that has mutually distinct letter frequencies. This supposedly unique infinite word is called *Fraenkel's sequence* and is given by $(F_k)^\omega$ where the *Fraenkel words* $(F_i)_{i \geq 1}$ are defined recursively by $F_1 = a_1$ and $F_i = F_{i-1} a_i F_{i-1}$ for all $i \geq 2$. For further details see [10,20].

Recall that a word t is circularly balanced if the infinite word t^ω is balanced. Thus the following corollary follows from Theorem 4 and Lemma 2.

Corollary 3. *A finite epistandard word is circularly balanced if and only if it belongs to one of the three families described in Theorem 4.*

6 Rich Words

In [7], it was proved that any word w of length $|w|$ contains at most $|w|+1$ distinct palindromic factors (including the empty word). The episturmian sequences, which include the Sturmian sequences, are "rich" in palindromes, in the sense that they contain the maximum number of different palindromic factors. Specifically, in [7], it was proved that if an infinite word w is episturmian, then any factor u of w contains exactly $|u|+1$ distinct palindromic factors.

Glen et al. in [10] introduced and studied rich words, that constitute a new class of finite and infinite words characterized by containing the maximal number of distinct palindromes. More precisely, a finite word w is *rich* if it has exactly $|w|+1$ distinct palindromic factors. A word is rich if all of its factors are rich. Rich words have been recently investigated in several papers (cf. [10,3,4]). In particular, we mention some results from [10], that will be useful in the sequel.

Theorem 5. *Recurrent balanced rich infinite words are precisely the balanced episturmian words.*

Corollary 4. *Recurrent balanced rich infinite words with mutually distinct letter frequencies are Sturmian words or have the form given by Fraenkel's conjecture.*

Proposition 1. *For a finite word w, the following properties are equivalent:*

1. *w^ω is rich;*
2. *w^2 is rich;*
3. *w is a product of two palindromes and all of the conjugates of w (including itself) are rich.*

We say that a finite word w is *circularly rich* if the infinite word w^ω is rich. We denote by $\mathcal{R}$ the set of the circularly rich words. The following key result relates rich words and words having simple Burrows-Wheeler transform.

Theorem 6. *If the word w belongs to $\mathcal{S}$ then w is circularly rich.*

The proof makes use of the Theorem 1 and Corollary 1 (cf. [22] and [23]) and of several properties of rich words proved in [10,3,4]. In fact by Corollary 1 each $w \in \mathcal{S}$ has the two palindrome property. Then, by the Property 3 of Proposition 1, it suffices to prove that all the conjugates of w are rich.

In the proof, we use the characterization in [10], where a word x is rich if and only if every suffix of x has a unioccurrent palindromic prefix (upp for short). If x has a upp, say p, then p is the longest palindromic prefix of x.

So we prove, by induction on h (where $1 \leq h \leq |w|$), that each factor of length h of words in $[w]$, or analogously each suffix of length h of a conjugate of w, is rich. The result is clearly true if $h \leq 3$, in fact it is easy to verify that all words of length 3 or less are rich. Now suppose the statement is true for all factors of length less than or equal to h, i.e. each factor u of $[w]$ of length h is rich, and we prove that each factor v of $[w]$ of length $h+1$ is rich. We can write $v = bu$, with $b \in A$. If a is the last letter of u, we can write $v = bu = bu'a$, with $a \in A$ and u' a factor of $[w]$ of length $h-1$. In [22], it was proved that if the words of the form $u'a$ and bu' are rich (so $u'a$ has a upp, say p, and bu' has a upp, say q) then $|q| \leq |p| + 2$. Moreover, from the properties of rich words, it was proved that, if either $|q| > 1$ and $|p| \leq |q| \leq |p| + 2$ or $|q| < |p|$ and $h - 1 \leq |p| \leq h$, then v is rich. Thus, it remains to prove the case where $|q| < |p| < h - 1$ and $|q| > 1$ (case 1) and the case where $|q| \leq |p|$ and $|q| = 1$ (case 2). The proof of such cases is long and complex, and cannot be reported here. It is obtained by contradiction. Indeed, we show that if $v = bu'a$ is not rich, then v contains two occurrences of q and, in particular, q appears as suffix of v. So $v = bu'b$ and u' is not a palindrome (otherwise v is a palindrome too and the upp of itself). In [22], it was proved that the condition that u' is not a palindrome contradicts the properties of the BW matrix M of a word w in S. For the complete proof of Theorem 6 we refer to [22]. The following example shows that the converse of Theorem 6 is false.

Example 1. The word $w = ccaaccb$ is circularly rich, but $bwt(w) = cacccba$, hence $w \notin \mathcal{S}$.

7 Simple BWT over Alphabets of Size Greater Than Two

In this section, we prove the following relationship:

$$\mathcal{B} \cap \mathcal{R} = \mathcal{B} \cap \mathcal{EP} = \mathcal{B} \cap \mathcal{S}.$$

We observe that the notions of epistandard, circularly balanced and circularly rich word are invariant under letter permutation. On the contrary the property that the word has simple BWT depends on the order of the alphabet. Hence the equivalence that we state in the sequel between some of these notions holds true up to letter permutation.

Theorem 7. *Any conjugate of a word in one of the three families defined in Theorem 4 belongs (up to letter permutation) to the set $\mathcal{S}$.*

Proof. We consider the words t of the three families in the form given in Theorem 4, and the alphabet order $a_1 < a_2 < \ldots < a_k$. In three cases, by the structure of t, we can determine the factor that follows each occurrence of letters of A in each conjugate of t. Then we prove that the letters of the last column L of Burrows-Wheeler matrix M of t are non-increasing.

Type *i)* : $t = pa_2$, with $p = Pal(a_1^m a_k a_{k-1} \cdots a_3)$.
Each occurrence of letter a_k is followed by factor $a_1^m a_j$ with $1 < j < k$. Each occurrence of letter a_i, with $2 < i < k$, is followed by factor $Pal(a_1^m a_k \cdots a_{i+1})a_j$, with $1 < j < i$. The unique occurrence of letter a_2 is followed by factor $Pal(a_1^m a_k \cdots a_3)$. Finally, each occurrence of letter a_1 is followed either by factor $a_1^h a_j$ (only in the case $m > 1$), with $1 \leq h \leq m-1$, or by the letter a_j, with $2 \leq j \leq k$.

Type *ii)*: $t = pa_2$, with $p = Pal(a_1 a_k a_{k-1} \cdots a_{k-\ell} a_1 a_{k-\ell-1} a_{k-\ell-2} \cdots a_3)$.
Each occurrence of letter a_k is followed by factor $a_1 a_j$, with $1 \leq j < k$. Each occurrence of letter a_i, with $k-\ell \leq i \leq k-1$, is followed by factor $Pal(a_1 a_k \cdots a_{i+1})a_j$, with $1 \leq j < i$. Each occurrence of letter $a_{k-\ell-1}$ is followed by factor $Pal(a_1 a_k \cdots a_{k-\ell} a_1)a_j$, with $1 < j < k-\ell-1$. Each occurrence of letter a_i, with $2 < i < k-\ell-1$, is followed by factor $Pal(a_1 a_k \cdots a_{k-\ell} a_1 a_{k-\ell-1} \cdots a_{i+1})a_j$, with $1 < j < i$. The unique occurrence of letter a_2 is followed by factor $Pal(a_1 a_k a_{k-1} \cdots a_{k-\ell} a_1 a_{k-\ell-1} \cdots a_3)$. Finally, each occurrence of letter a_1 is followed either by factor $Pal(a_1 a_k \cdots a_{k-\ell})a_j$, with $2 \leq j \leq k-\ell-1$, or by letter a_j, with $2 \leq j \leq k$.

Type *iii)*: $t = Pal(a_1 a_k a_{k-1} \cdots a_2)$.
Each occurrence of letter a_k is followed by factor $a_1 a_j$, with $1 \leq j < i$, each occurrence of letter a_i, with $1 < i < k$, is followed by factor $Pal(a_1 a_k \cdots a_{i+1})a_j$, with $1 \leq j < i$. Finally, each occurrence of letter a_1 is followed either by factor $Pal(a_1 a_k \cdots a_3)a_2$ or by letter a_j with $2 \leq j \leq k$.

Clearly, in all cases, the smallest rows (in the lexicographic order) of M end with a_k; moreover the greatest rows end with a_1. Hence the intermediate rows end with a_i for $1 < i < k$. Now we consider two conjugates t' and t'' of t, where the last letter of t', say a_i, is greater than the last letter of t'', say a_j, where $1 < j < i < k$. By the relation $a_j < a_i$ we derive that $t' < t''$. Indeed, by structure of t, the longest common prefix of t' and t'' is a palindromic factor with a_{i+1} as central letter. Moreover such a factor is followed by a letter $a_h < a_i$ in t' and by the letter a_i in t''. Since $a_h < a_i$, we obtain $t' < t''$. □

We are now ready to prove the main result of the paper.

Theorem 8. *Let $A = \{a_1, a_2, \ldots, a_k\}$ be a totally ordered alphabet and let $w \in A^*$ be a primitive circularly balanced word over A. The following statements are equivalent up to a letter permutation:*

i) w belongs to $\mathcal{S}$;
ii) w is a circularly rich word;
iii) w is a conjugate of a finite epistandard word.

Proof. $i) \Rightarrow ii)$: it follows from the Theorem 6.
$ii) \Leftrightarrow iii)$: it follows from Theorem 5 and Lemma 2.
$iii) \Rightarrow i)$: it follows from Theorem 7 and Lemma 1. □

Example 2. The circularly balanced word $w = adacadabadacada$ belongs to $\mathcal{S}$, is a finite epistandard word and is circularly rich. □

The following examples show that the notions coincide only under assumption of balancing.

Example 3. The non-circularly balanced word $w = bbbbbacaca$ belongs to $\mathcal{S}$ (clearly it is circularly rich), but it is not a finite epistandard word. The non-circularly balanced word $w = (adac)^2adab(adac)^2ada(adac)^2adab(adac) \notin \mathcal{S}$ and it is a finite epistandard word. □

The following example shows that there exist non-circularly balanced words which belong to $\mathcal{EP} \cap \mathcal{S}$.

Example 4. The non-circularly balanced word $w = acabac$ is a finite epistandard word and belongs to $\mathcal{S}$. □

References

1. Altman, E., Gaujal, B., Hordijk, A.: Balanced sequences and optimal routing. Journal of the ACM 47(4), 752–775 (2000)
2. Arnoux, P., Rauzy, G.: Représentation géométrique de suites de complexité 2n + 1. Bull. Soc. Math. France 119, 199–215 (1991)
3. Bucci, M., De Luca, A., Glen, A., Zamboni, L.Q.: A connection between palindromic and factor complexity using return words. Advance. Advances in Applied Mathematics 42, 60–74 (2009)
4. Bucci, M., De Luca, A., Glen, A., Zamboni, L.Q.: A new characteristic property of rich words. Theoretical Computer Science (in press) (2008) doi: 10.1016/j.tcs.2008.11.001

5. Burrows, M., Wheeler, D.J.: A block sorting data compression algorithm. Technical report, DIGITAL System Research Center (1994)
6. de Luca, A.: Sturmian words: structure, combinatorics, and their arithmetics. Theoretical Computer Science 183(1), 45–82 (1997)
7. Droubay, X., Justin, J., Pirillo, G.: Episturmian words and some constructions of de Luca and Rauzy. Theoretical Computer Science 255(1-2), 539–553 (2001)
8. Aviezri, S.F.: Complementing and exactly covering sequences. J. Combinatorial Theory Ser. A 14(1), 8–20 (1973)
9. Glen, A., Justin, J.: Episturmian words: a survey. To appear in RAIRO Theoretical Informatics and Applications (2009)
10. Glen, A., Justin, J., Widmer, S., Zamboni, L.Q.: Palindromic richness. European Journal of Combinatorics 30(2), 510–531 (2009)
11. Jenkinson, O., Zamboni, L.Q.: Characterisations of balanced words via orderings. Theoretical Computer Science 310(1-3), 247–271 (2004)
12. Justin, J.: Episturmian morphisms and a Galois theorem on continued fractions. Theoretical Informatics and Applications 39(1), 207–215 (2005)
13. Justin, J., Pirillo, G.: Episturmian words and episturmian morphisms. Theoretical Computer Science 276(1-2), 281–313 (2002)
14. Lothaire, M.: Combinatorics on Words. Encyclopedia of Mathematics, vol. 17. Addison-Wesley, Reading (1983); Reprinted in the Cambridge Mathematical Library, Cambridge University Press (1997)
15. Lothaire, M.: Algebraic Combinatorics on Words. Cambridge University Press, Cambridge (2002)
16. Mantaci, R., Mantaci, S., Restivo, A.: Balance properties and distribution of squares in circular words. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 504–515. Springer, Heidelberg (2008)
17. Mantaci, S., Restivo, A., Sciortino, M.: Burrows-Wheeler transform and Sturmian words. Information Processing Letters 86, 241–246 (2003)
18. Morse, M., Hedlund, G.A.: Symbolic dynamics II. Sturmian trajectories. American Journal of Mathematics 62(1), 1–42 (1940)
19. Paquin, G.: On a generalization of Christoffel words: epichristoffel words. In: Proceedings of Journées Montoises D'Informatique Théorique (2008)
20. Paquin, G., Vuillon, L.: A characterization of balanced episturmian sequences. Electronic Journal of Combinatorics 14, 12 (2006)
21. Rauzy, G.: Suites à termes dans un alphabet fini. In: Séminaire de théorie des Nombres de Bordeaux. Exposé, vol. 25, pp. 1–16 (1982/1983)
22. Restivo, A., Rosone, G.: Burrows-Wheeler transform and palindromic richness. Theoretical Computer Science (in press) (accepted manuscript) (2009) doi: 10.1016/j.tcs.2009.03.008
23. Simpson, J., Puglisi, S.J.: Words with simple Burrows-Wheeler transforms. Electronic Journal of Combinatorics 15, article R83 (2008)
24. Vuillon, L.: Balanced words. Bull. Belg. Math.Soc. 10(5), 787–805 (2003)

On the Complexity of Hmelevskii's Theorem and Satisfiability of Three Unknown Equations

Aleksi Saarela*

Department of Mathematics and Turku Centre for Computer Science TUCS,
University of Turku, 20014 Turku, Finland
amsaar@utu.fi

Abstract. We analyze Hmelevskii's theorem, which states that the general solutions of constant-free equations on three unknowns are expressible by a finite collection of formulas of word and numerical parameters. We prove that the size of the finite representation is bounded by an exponential function on the size of the equation. We also prove that the shortest nontrivial solution of the equation, if it exists, is exponential, and that its existence can be solved in nondeterministic polynomial time.

1 Introduction

This work concerns the theory of word equations, which is a fundamental part of combinatorics on words. It has connections to many other areas including representation results of algebra, theory of algorithms and pattern matching.

Some remarkable results of this topic proved during the last few decades are the decidability of the satisfiability problem for word equations, see [11], and the compactness result of systems of word equations, see [1] and [6]. The first result was improved to a PSPACE algorithm in [12]. The satisfiability problem has been conjectured to be in NP [13].

In the case of constant-free word equations with only three unknowns important results have also been achieved. Hmelevskii [8] proved in 1970 that the general solution of any such equation can be expressed as a finite formula on word and numerical parameters. On the other direction Spehner [15,16] classified all sets of relations a given solution can satisfy. Both of these results have only very complicated proofs. Another example of a challenging nature of word problems is that the question of finding any upper bound for the maximal size of independent system of word equations on three unknowns is still open, see [7] and [3].

The result of Hmelevskii is well known, see e.g. [10], but the original presentation is very hard to read. A simplified proof using modern tools of combinatorics on words has been given, together with a double exponential upper bound of the size of the formula giving the general solution, see [9]. A complete write-up of the results in [9] is in [14].

* Supported by the Academy of Finland under grant 8121419.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 443–453, 2009.

In this paper we continue the work of analyzing Hmelevskii's result. Based on [9] and [14] we improve the bound of the size of the parametric solution to single exponential, as well as prove that the length of the shortest nontrivial solution is also exponential (if such a solution exists). This connects our work to the satisfiability problem mentioned above, because Plandowski and Rytter proved in [13] that there is a nondeterministic algorithm solving the problem in time polynomial in $n \log N$, where n is the length of the equation and N is the length of the shortest solution. From this and our result it follows that the problem of deciding if a constant-free equation on three unknowns has a nontrivial solution is in NP.

2 Definitions

We begin by giving some definitions needed in this paper. A basic reference of the subject is [2].

We assume that all word equations are constant-free unless otherwise stated. Thus we consider word equations $U = V$, where $U, V \in \Xi^*$ and Ξ is the alphabet of unknowns. A morphism $h : \Xi^* \to \Sigma^*$ is a solution of this equation, if $h(U) = h(V)$. We also consider *one-sided* equations $xU \rightrightarrows yV$. A morphism $h : \Xi^* \to \Sigma^*$ is a solution of this equation, if $h(xU) = h(yV)$ and $|h(x)| \geq |h(y)|$.

A solution h is *periodic*, if there exists a $t \in \Sigma^*$ such that every $h(x)$, where $x \in \Xi$, is a power of t. Otherwise h is *nonperiodic*. Periodic solutions are easy to find and represent, so in many cases it is enough to consider nonperiodic solutions.

If a word u is a *prefix* of a word v, that is $v = uw$ for some w, the notation $u \leq v$ is used. If also $u \neq v$, then u is a *proper prefix* and the notation $u < v$ is used.

Let $w = a_1 \dots a_n$. Its *reverse* is $w^R = a_n \dots a_1$, and its *length* is $|w| = n$. The number of occurrences of a letter a in w is denoted by $|w|_a$.

If $\Sigma = \{a_1, \dots, a_n\}$, then $U \in \Sigma^*$ can be denoted $U(a_1, \dots, a_n)$, and its image under a morphism h can be denoted $h(U) = U(h(a_1), \dots, h(a_n))$. If $u \in \Sigma^*$, then the morphism $a_1 \mapsto u$ means the morphism, which maps $a_1 \mapsto u$ and $a_i \mapsto a_i$, when $i = 2, \dots, n$.

Next we define the central notions of this paper: parametric words and parametric solutions.

We fix the alphabet of *word parameters* Δ and the set of *numerical parameters* Λ. Now *parametric words* are defined inductively as follows:

(i) if $a \in \Delta \cup \{1\}$, then (a) is a parametric word,
(ii) if α and β are parametric words, then so is $(\alpha\beta)$,
(iii) if α is a parametric word and $i \in \Lambda$, then (α^i) is a parametric word.

The set of parametric words is denoted by $\mathcal{P}(\Delta, \Lambda)$. The sets of parameters are always denoted by Δ and Λ.

When there is no danger of confusion, unnecessary parentheses can be omitted and notations like $\alpha^i\alpha^j = \alpha^{i+j}$ and $(\alpha^i)^j = \alpha^{ij}$ can be used. Then parametric words form a monoid, if the product of α and β is defined to be $\alpha\beta$.

If f is a function $\Lambda \to \mathbb{N}_0 = \{0, 1, 2, \dots\}$, we can abuse the notation and use the same symbol for the function, which maps parametric words by giving values for the numerical parameters with f: if $a \in \Delta \cup \{1\}$, then $f((a)) = a$; if $\alpha, \beta \in \mathcal{P}(\Delta, \Lambda)$, then $f((\alpha\beta)) = f(\alpha)f(\beta)$; if $\alpha \in \mathcal{P}(\Delta, \Lambda)$ and $i \in \Lambda$, then $f((\alpha^i)) = f(\alpha)^{f(i)}$. A parametric word is thus mapped by f to a word of Δ^*. This can be further mapped by a morphism $h : \Delta^* \to \Sigma^*$ to a word of Σ^*. The mapping $h \circ f$ is a *valuation* of a parametric word into Σ^*, and f is its valuation to the set Δ^*.

We define the *length* of a parametric word: the length of 1 is zero; if $a \in \Delta$, then the length of a is one; if $\alpha, \beta \in \mathcal{P}(\Delta, \Lambda)$, then the length of $\alpha\beta$ is the sum of the lengths of α and β; if $\alpha \in \mathcal{P}(\Delta, \Lambda) \setminus \{1\}$ and $i \in \Lambda$, then the length of α^i is the length of α plus one.

Next we define the *height* of a parametric word: if $a \in \Delta \cup \{1\}$, then the height of a is zero; if $\alpha, \beta \in \mathcal{P}(\Delta, \Lambda)$, then the height of $\alpha\beta$ is the maximum of the heights of α and β; if $\alpha \in \mathcal{P}(\Delta, \Lambda) \setminus \{1\}$ and $i \in \Lambda$, then the height of α^i is the height of α plus one. Parametric words of height zero can be considered to be words of Δ^*.

A *linear Diophantine relation* R is a disjunction of systems of linear Diophantine equations with lower bounds for the unknowns. For example,

$$((x + y - z = 0) \wedge (x \geq 2)) \vee ((x + y = 3) \wedge (x + z = 4))$$

is a linear Diophantine relation over the unknowns x, y and z. We are only interested in the nonnegative values of the unknowns. If $\Lambda = \{i_1, \dots, i_k\}$, f is a function $\Lambda \to \mathbb{N}_0$, and $f(i_1), \dots, f(i_k)$ satisfy R, then the notation $f \in R$ can be used.

Let S be a set of morphisms $\Xi^* \to \Sigma^*$, $\Lambda = \{i_1, \dots, i_k\}$, h_j a morphism from the monoid Ξ^* to parametric words and R_j a linear Diophantine relation, when $j = 1, \dots, m$. The set $\{(h_j, R_j) : 1 \leq j \leq m\}$ is a *parametric representation* of S, if

$$S = \{h \circ f \circ h_j : 1 \leq j \leq m, f \in R_j\},$$

where $h \circ f$ runs over all valuations to Σ^*. The linear Diophantine relations are not strictly necessary, but they make some proofs easier. A set can be *parameterized*, if it has a parametric representation. The *length* of the parametric representation is the sum of the lengths of all $h_j(x)$, where $j = 1, \dots, m$ and $x \in \Xi$.

We conclude these definitions by saying that solutions of an equation can be *parameterized*, if the set of its all solutions can be parameterized. A parametric representation of this set is a *parametric solution* of the equation. These definitions can be generalized in an obvious way for systems of equations.

Example 2.1. The equation $xz = zy$ has a parametric solution $\{(h_1, R), (h_2, R)\}$, where $\Delta = \{p, q\}$, $\Lambda = \{i\}$, $h_1(x) = pq$, $h_1(y) = qp$, $h_1(z) = p(qp)^i$, $h_2(x) = h_2(y) = 1$, $h_2(z) = p$ and R is the trivial relation satisfied by all functions $f : \Lambda \to \mathbb{N}_0$.

3 Remarks about Parametric Solutions

Next we make some remarks about parametric solutions to increase our understanding of them. The various claims made in this section are not needed in this paper. The proofs can be found in [14].

A parametric solution was defined as a set $\{(h_j, R_j) : 1 \leq j \leq m\}$. This solution can be written less formally as

$$\begin{aligned}&x = h_1(x),\ \ y = h_1(y),\ \ z = h_1(z),\ \ R_1\ \text{ or}\\ &\ \vdots\\ &x = h_m(x),\ y = h_m(y),\ z = h_m(z),\ R_m,\end{aligned}$$

if the unknowns are x, y, z. Actually, only one pair (h, R) is needed. For example, if we have a parametric solution

$$x = \alpha_1,\ y = \beta_1,\ z = \gamma_1 \qquad \text{or} \qquad x = \alpha_2,\ y = \beta_2,\ z = \gamma_2,$$

we can replace it with

$$x = \alpha_1^i \alpha_2^j,\ y = \beta_1^i \beta_2^j,\ z = \gamma_1^i \gamma_2^j, \quad i + j = 1,$$

where i and j are new parameters.

On the other hand, the linear Diophantine relations are not necessary either, if we again allow many morphisms. We can get rid of the relations by replacing every pair (h, R) with several morphisms h. This follows from article [4].

Example 3.1. Consider the periodic solutions of the equation $x^n = yz$. They are

$$x = t^i,\ y = t^j,\ z = t^k, \quad ni = j + k.$$

We can replace j with $nj' + b$ and k with $nk' + c$, where $0 \leq b, c < n$. Then $i = j' + k' + (b + c)/n$. Only those pairs (b, c) for which $b + c$ is divisible by n are possible. Thus we get a representation

$$\begin{aligned}&x = t^{j'+k'}, && y = t^{nj'}, && z = t^{nk'} && \text{or}\\ &x = t^{j'+k'+1}, && y = t^{nj'+1}, && z = t^{nk'+n-1} && \text{or}\\ &x = t^{j'+k'+1}, && y = t^{nj'+2}, && z = t^{nk'+n-2} && \text{or}\\ &\ \vdots\\ &x = t^{j'+k'+1}, && y = t^{nj'+n-1}, && z = t^{nk'+1},\end{aligned}$$

where the parameters j', k' can now have any nonnegative values.

The periodic solutions of an equation on three unknowns can be represented with just one morphism and without any Diophantine relations. This does not hold, if instead of periodic solutions we consider all solutions. Indeed, a parametric solution for the equation $xyxzzyz = zxzyxy$ consists of at least three morphisms, if linear Diophantine relations are not allowed. The next example gives the solutions of this equation. We are not aware of any better lower bounds for the maximal required number of morphisms in these kinds of parametric solutions.

Example 3.2. The solutions of the equation $xyxzzyz = zxzyxy$ are

$$x = p,\ y = q,\ z = 1 \quad \text{or} \quad x = p,\ y = q,\ z = pq \quad \text{or} \quad x = p^i,\ y = p^j,\ z = p^k,$$

where $p, q \in \Sigma^*$ and $i, j, k \geq 0$.

4 Basic Equations

Hmelevskii proved that every equation on three unknowns has a parametric solution, and the size of this solution was estimated to be at most double exponential in [9]. Our first goal is to improve this bound to single exponential, and our second goal is to prove that the shortest nontrivial solution is also of exponential size. We need to refer to the theorems and proofs in [14]. Often these theorems claim the existence of some object, while we need to know also something about the size or structure of that object. Typically this information can be obtained simply by examining the old proof, but this is not trivial. In these cases we state the more precise form of the theorem, but do not repeat the proof.

In this section the new information is about the coefficients of some linear Diophantine relations. This is necessary for our second goal.

Let α and β be parametric words. The pair (α, β) can be viewed as an equation, referred to as an *exponential equation*. The *height* of this equation is the height of $\alpha\beta$. The solutions of this equation are the functions $f : \Lambda \to \mathbb{N}_0$ that satisfy $f(\alpha) = f(\beta)$.

The following three theorems were proved in [14, Theorems 5.1, 5.2, 5.3] except for the upper bounds of the sizes of the coefficients in the relation R. These bounds, however, are easily obtained by examining the proofs. The latter two theorems (especially the last one) are technical variations of the first one.

Theorem 4.1. *Let $E : \alpha = \beta$ be an exponential equation of height one. There exists a linear Diophantine relation R such that a function $f : \Lambda \to \mathbb{N}_0$ is a solution of E if and only if $f \in R$. The sizes of the coefficients in R are of the same order as the length of $\alpha\beta$.*

Theorem 4.2. *Let $\Lambda = \{i, j\}$ and let $s_0, \ldots, s_m$, $t_1, \ldots, t_m$, $u_0, \ldots, u_n$ and $v_1, \ldots, v_n$ be parametric words of height at most one, with no occurrences of parameter j. Assume that i occurs at least in the words $t_1, \ldots, t_m$ and $v_1, \ldots, v_n$. Let $\alpha = s_0 t_1^j s_1 \ldots t_m^j s_m$ and $\beta = u_0 v_1^j u_1 \ldots v_n^j u_n$. Now there exists a linear Diophantine relation R such that a function $f : \Lambda \to \mathbb{N}_0$ is a solution of the exponential equations $E : \alpha = \beta$ if and only if $f \in R$. The sizes of the coefficients in R are of the same order as the length of $\alpha\beta$.*

Theorem 4.3. *Let $\Delta = \{p, q\}$, $\Lambda = \{i, j, k\}$ and $a \geq 2$. Let $\alpha = (pq^a)^i p$, $\beta = q$, $\gamma = (pq^a)^j p$, or*

$$\begin{cases} \alpha &= qp((pq)^{k+1}p)^{a-2}pq(((pq)^{k+1}p)^{a-1}pq)^i, \\ \beta &= (pq)^{k+1}p, \\ \gamma &= qp((pq)^{k+1}p)^{a-2}pq(((pq)^{k+1}p)^{a-1}pq)^j. \end{cases}$$

Let $A, B \in \{x, y, z\}^*$ and let h be the morphism mapping $x \mapsto \alpha, y \mapsto \beta, z \mapsto \gamma$. Now there exists a linear Diophantine relation R such that a function $f : \Lambda \to \mathbb{N}_0$ is a solution of the exponential equation $E : h(A) = h(B)$ if and only if $f \in R$. The sizes of the coefficients in R are of the same order as the length of $h(A)h(B)$.

From now on we only consider equations with three unknowns. The alphabet of unknowns is $\Xi = \{x, y, z\}$. The left-hand side of an equation can be assumed to begin with x. We can also assume that x occurs on the right-hand side, but not as the first letter.

Periodic solutions and solutions, where some unknown has the value 1, are called *trivial*. These are easy to parameterize.

An equation is a *basic equation*, if it is a trivial equation $U = U$, where $U \in \Xi^*$, if it has only trivial solutions, or if it is of one of the following forms, where $a, b \geq 1$, $c \geq 2$ and $t \in \{x, z\}$:

B1. $x^a y \ldots = y^b x \ldots$

B2. $x^2 \ldots \rightrightarrows y^a x \ldots$

B3. $xyt \ldots \rightrightarrows zxy \ldots$

B4. $xyt \ldots \rightrightarrows zyx \ldots$

B5. $xyz \ldots = zxy \ldots$

B6. $xyz \ldots = zyx \ldots$

B7. $xy^c z \ldots = zy^c x \ldots$

B8. $xyt \ldots \rightrightarrows z^a xy \ldots$

B9. $xyxz \ldots \rightrightarrows zx^2 y \ldots$

The parameterizability of basic equations is quite easy to prove and was done in [14, Theorem 6.2]. The $O(n)$ bound for the coefficients in the linear Diophantine relations follows from the bounds in Theorems 4.1, 4.2 and 4.3.

Theorem 4.4. *Every basic equation has a parametric solution. The solution is of length $O(1)$ and the coefficients in the linear Diophantine relations are of size $O(n)$, where n is the length of the equation.*

5 Length of the Parametric Solution

In this section we prove that the size of the parametric solution is exponential. At the same time we improve some of the theorems in [14] so that they can be used later to prove the existence of a nontrivial solution of exponential size.

First we define images of equations and some other related concepts. These definitions are very important in the proof of Hmelevskii's theorem.

An *image* of an equation $xU(x, y, z) \rightrightarrows V(y, z)xW(x, y, z)$ under the morphism $x \mapsto V^k P x$, where $k \geq 0$, $V = PQ$ and $Q \neq 1$, is

$$xU(V^k Px, y, z) \leftleftarrows QPxW(V^k Px, y, z).$$

If V contains only one of y, z or if $P = 1$, the image is *degenerated.*

Equation E is *reduced to the equations $E_1, \dots, E_n$ by an n-tuple of substitutions*, if E is of the form $xU(x, y, z) \rightrightarrows t_1 \dots t_k x V(x, y, z)$, where $1 \leq n \leq k$ and $t_1, \dots, t_k \in \{y, z\}$, equation E_i is

$$xU(t_1 \dots t_i x, y, z) \leftleftarrows t_{i+1} \dots t_k t_1 \dots t_i x V(t_1 \dots t_i x, y, z),$$

when $1 \leq i < n$, and equation E_n is

$$xU(t_1 \dots t_n x, y, z) = t_{n+1} \dots t_k t_1 \dots t_n x V(t_1 \dots t_n x, y, z).$$

A sequence of equations $E_0, \dots, E_n$ is a *chain*, if E_i is an image of E_{i-1} for all i, $1 \leq i \leq n$. Then E_n is an *image of order n* of E_0. If every E_i is a degenerated image, then the chain is degenerated and E_n is a degenerated image of order n.

The following lemma is the same as [14, Lemma 8.1].

Lemma 5.1. *Let $u, v, w \in \Sigma^*$, $0 < |w| \leq |u|$ and $c \geq 1$. If*

$$wu^{c+1}v \dots = u^{c+1}vu \dots \qquad \textit{or} \qquad w(uv)^c u^2 \dots = (uv)^c u^2 \dots,$$

then $uv = vu$.

The next lemma is a seemingly minor but essential improvement of [14, Lemma 8.2]: the number k in the lemma can be selected to be logarithmic instead of linear with respect to the number $|p - q|$. This is what ultimately leads to an exponential bound for the length of the parametric solution.

Lemma 5.2. *Let E_0 be the equation $xy^a zy^p s \dots \rightrightarrows zy^b xy^q t \dots$, where $s, t \in \{x, z\}$ and $a + p \neq b + q$. Let k be an even number such that $2^{(k-4)/2} \geq 1 + |p - q|$. Let E_k be the equation $xP \rightrightarrows zQ$ and $E_0, \dots, E_k$ be a degenerated chain. Now the solutions of E_k satisfying $y \neq 1$ are also solutions of the equation $xy^a zy^b \rightrightarrows zy^b xy^a$.*

Proof. Assume that E_{i+1} is the image of E_i under the morphism $f_i : x \mapsto (zy^b)^{c_i} x$, when i is even, and under the morphism $f_i : z \mapsto (xy^a)^{c_i} z$, when i is odd. Because $f_0(x)$ and $f_0(z)$ and thus $f_0(s)$ and $f_0(t)$ begin with z, the equation E_k is of the form

$$xy^a zy^p r \dots \rightrightarrows zy^b xy^q r \dots, \tag{1}$$

where

$$r = (f_k \circ \dots \circ f_1)(z) = (f_k \circ \dots \circ f_4)((((xy^a)^{c_3} zy^b)^{c_2} xy^a)^{c_1} (xy^a)^{c_3}).$$

Let $F_m = f_m \circ \dots \circ f_4$. The words xy^a and zy^b occur as factors of $F_4(xy^a)$ at least once, and if they occur as factors of $F_m(xy^a)$ at least $2^{(m-4)/2}$ times, they occur as factors of $F_{m+2}(xy^a)$ at least $2^{(m-2)/2}$ times. Thus, by induction, they occur as factors of $F_k(xy^a)$ at least $2^{(k-4)/2}$ times. If h is a solution of E_k, then

$$\begin{aligned} ||h(xy^a zy^p)| - |h(zy^b xy^q)|| &= |a + p - b - q||h(y)| \\ \leq (a + b)|h(y)| + |p - q||h(y)| &\leq (1 + |p - q|)|h(xy^a zy^b)| \\ \leq 2^{(k-4)/2}|h(xy^a zy^b)| \qquad &\leq |h(F_k(xy^a))|. \end{aligned}$$

Thus, by (1),

$$w((u^{c_3}v)^{c_2}u)^{c_1}u^{c_3}\ldots = ((u^{c_3}v)^{c_2}u)^{c_1}u^{c_3}\ldots,$$

where $u = h(F_k(xy^a))$, $v = h(F_k(zy^b))$ and $|w| \leq |u|$. If $w = 1$, then $h(xy^a zy^p) = h(zy^b xy^q)$, which is not possible by the assumptions $h(y) \neq 1$ and $a + p \neq b + q$. Thus it follows from Lemma 5.1 that $uv = vu$. It can be seen that $u, v \in \{h(xy^a), h(zy^b)\}^*$, u ends with $h(xy^a)$ and v ends with $h(zy^b)$. This means that $h(xy^a)$ and $h(zy^b)$ satisfy a nontrivial relation. It follows that they commute, that is $h(xy^a zy^b) = h(zy^b xy^a)$. □

The equations $E_1, \ldots, E_n$ form a *neighborhood* of an equation E, if one of the following conditions holds:

N1. $E_1, \ldots, E_n$ form a complete set of θ-images of E (see [14]),
N2. E reduces to $E_1, \ldots, E_n$ with an n-tuple of substitutions,
N3. E is the equation $U = V$, U and V begin with different letters, $n = 2$, and E_1 and E_2 are equations $U \rightrightarrows V$ and $V \rightrightarrows U$,
N4. $n = 1$ and E is the equation $U = V$ and E_1 is the equation $U^R = V^R$,
N5. E is the equation $SU = TV$, $|S|_t = |T|_t$ for all $t \in \Xi$, $n = 1$ and E_1 is the equation $US = VT$,
N6. $n = 1$ and E_1 is E reduced from the left or multiplied from the right,
N7. $n = 1$ and, with the assumptions of Lemma 5.2, E is the equation $xP \rightrightarrows zQ$ and E_1 the equation $xy^a zy^b xP \rightrightarrows zy^b xy^a zQ$.

The first paragraph of the next theorem, proved in [14, Theorem 8.3], justifies the definition of a neighborhood. The second paragraph can be deduced by examining the rules in the definition of a neighborhood and, most importantly, the definition of a complete set of θ-images.

Theorem 5.3. *Let E be a word equation of length n and let $E_1, \ldots, E_m$ be its neighborhood. If each E_i has a parametric solution of length at most c, then E has a parametric solution of length $O(mn^{26})c$.*

Compared to the parametric solutions of the equations E_i, the parametric words in the parametric solution of E contain $O(1)$ new numerical parameters, the height of the parametric words can increase by $O(1)$, and the coefficients of the linear Diophantine relations are of the same size.

A directed acyclic graph, whose vertices are equations, is a *tree* of E, if the following conditions hold:

(i) only vertex with no incoming edges is E,
(ii) all other vertices have exactly one incoming edge,
(iii) if there are edges from E_0 to exactly $E_1, \ldots, E_n$, then these equations form a neighborhood of E_0.

The first paragraph of the next theorem is from [14, Theorem 8.4], and the second paragraph follows from the second paragraph of Theorem 5.3 and from Theorem 4.4.

Theorem 5.4. *Let E be a word equation of length n. If E has a tree of height k, then all equations in the tree are of length $O(n)^{27^k}$. If each leaf equation in this tree has a parametric solution of length at most c, then E has a parametric solution of length $O(n)^{52 \cdot 27^k} c$.*

If the leaf equations are basic equations, the parametric words in the parametric solution of E contain $O(k)$ numerical parameters, their height is $O(k)$, and the coefficients of the linear Diophantine relations are of size $O(n)^{27^k}$.

A tree in which all leaves are basic equations is a *basic tree.*

The old version of Lemma 5.2 was used in [14, Lemmas 9.3, 10.2]. By using the improved version and making the corresponding small changes in the proof of [14, Theorem 10.5] gives the following theorem.

Theorem 5.5. *Every equation of length n with three unknowns has a basic tree of height $O(\log n)$.*

Now we can prove one of our main results. We note that it seems unlikely that Hmelevskii's methods would give a sub-exponential bound.

Theorem 5.6. *Every equation of length n with three unknowns has a parametric solution of length $\exp(n^{O(1)})$.*

Proof. By Theorem 5.5, every equation has a basic tree of height $O(\log n)$. By Theorem 4.4, the leaf equations have parametric solutions of bounded length. Now from Theorem 5.4 it follows that E has a parametric solution of length $O(n)^{52 \cdot 27^k}$, where $k = O(\log n)$, that is of length $\exp(n^{O(1)})$. □

6 Shortest Nontrivial Solution

Based on Theorem 5.6 we can prove that the shortest nontrivial solution is of exponential length. However, this is not trivial. For example, if we have a parametric word $(p^i q)^j$, then by giving the value 1 for the numerical parameters we get a short word, but the problem is that $i = j = 1$ does not necessarily satisfy the linear Diophantine relation. Thus we need to estimate the size of the minimal solution of the relation. We also need to make sure that the solution of the word equation is indeed nontrivial.

Theorem 6.1. *If an equation of length n with three unknowns has a nontrivial solution, it has a nontrivial solution of length $\exp(n^{O(1)})$.*

Proof. Consider an equation $E : x \ldots = y \ldots$ and its parametric solution

$$\{(h_j, R_j) : 1 \leq j \leq m\}$$

of length $\exp(n^{O(1)})$. If E has a nontrivial solution, it has a solution where x and y begin with the same letter but z begins with a different letter. Let $h \circ f \circ h_j$ be such a solution, where $h \circ f$ is a valuation. Now also $f \circ h_j$ is such a solution, and so is $g \circ h_j$, if $g \in R_j$ maps exactly the same numerical parameters to zero as f. Thus $g \circ h_j$ is a nontrivial solution. We must select g so that this solution is sufficiently short.

The lengths of the parametric words $h_j(t)$, where $t \in \{x, y, z\}$, are $\exp(n^{O(1)})$. By Theorems 5.4 and 5.5, every occurrence of a word parameter in $h_j(t)$ appears at most $g(i_1) \dots g(i_k)$ times in $g(h_j(t))$, where $i_1, \dots, i_k$ are numerical parameters and $k = O(\log n)$. Thus the length of $g(h_j(t))$ is $g(i_1) \dots g(i_k) \exp(n^{O(1)})$.

The conditions for g are that it must be in R_j and it must map exactly the same numerical parameters to zero as f. The latter condition can be handled by adding either the equation $i = 0$ (if $f(i) = 0$) or the inequality $i > 0$ (if $f(i) > 0$) to R_j for every $i \in \Lambda$. Inequalities $i > c$ can be replaced with $i = c + 1 + i'$, where i' is a new variable. In this way we get a linear Diophantine relation R'_j, which is a disjunction of linear systems of equations. Because $f \in R'_j$, at least one of these systems has a nonnegative integer solution.

According to [5], if a system of linear equations has a nonnegative integer solution, then it has one of size $O(lM)$, where l is the number of unknowns, M is an upper bound for the $r \times r$ subdeterminants of the augmented matrix of the system, and r is the rank of the system. Now r is at most $l = O(\log n)$. The coefficients in the system are of exponential size by Theorems 5.4 and 5.5, so $M = \exp(n^{O(1)})$. Thus there is a nonnegative integer solution of size $\exp(n^{O(1)})$. This solution gives us a function g such that $g(i_1) \dots g(i_k) \exp(n^{O(1)}) = \exp(n^{O(1)})$. This proves the theorem. □

Now we consider the satisfiability problem. Constant-free equations have always the solution, where every unknown gets the value 1, and usually they have also other periodic solutions. The natural question is thus whether a constant-free equation has a nontrivial solution. This can be easily reduced to the satisfiability problem of equations with constants. In this way we get the result that the above-mentioned question is in NP for equations on three unknowns.

Theorem 6.2. *The existence of a nontrivial solution of a constant-free equation on three unknowns can be decided in nondeterministic polynomial time.*

Proof. The equation $xU = yV$, where $U, V \in \Xi^*$, has a nontrivial solution if and only if it has a solution $x = ax', y = ay', z = bz'$, where a and b are different letters and $x', y', z' \in \Sigma^*$. So we are interested in the existence of a solution for the equation obtained from $xU = yV$ by replacing x with ax', y with ay' and z with bz', where x', y', z' are now new unknowns. The length of this equation is twice the length of the original equation.

There is a nondeterministic algorithm (see [13]) that solves the existence of a solution for the last equation in time polynomial in $n \log N$, where n is the length of the equation and N is the length of the shortest solution. The claim now follows from Theorem 6.1. □

References

1. Albert, M.H., Lawrence, J.: A proof of Ehrenfeucht's conjecture. Theoret. Comput. Sci. 41, 121–123 (1985)
2. Choffrut, C., Karhumäki, J.: Combinatorics of words. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages. Springer, Heidelberg (1997)
3. Czeizler, E., Karhumäki, J.: On non-periodic solutions of independent systems of word equations over three unknowns. Internat. J. Found. Comput. Sci. 18, 873–897 (2007)
4. Eilenberg, S., Schützenberger, M.P.: Rational sets in commutative monoids. J. Algebra 13, 173–191 (1969)
5. von zur Gathen, J., Sieveking, M.: A bound on solutions of linear integer equalities and inequalities. Proc. Amer. Math. Soc. 72, 155–158 (1978)
6. Guba, V.S.: Equivalence of infinite systems of equations in free groups and semigroups to finite subsystems. Mat. Zametki 40, 321–324 (1986)
7. Harju, T., Karhumäki, J., Plandowski, W.: Independent systems of equations. In: Lothaire, M. (ed.) Algebraic Combinatorics on Words. Cambridge University Press, Cambridge (2002)
8. Hmelevskii, Y.I.: Equations in free semigroups. Proc. Steklov Inst. of Math. 107 (1971); Amer. Math. Soc. Translations (1976)
9. Karhumäki, J., Saarela, A.: An analysis and a reproof of Hmelevskii's theorem. In: Proc. of 12th International Conference on Developments in Language Theory, pp. 467–478 (2008)
10. Lothaire, M.: Combinatorics on words. Addison-Wesley, Reading (1983)
11. Makanin, G.S.: The problem of solvability of equations in a free semigroup. Mat. Sb. 103, 147–236; English transl. in Math. USSR Sb. 32, 129–198 (1977)
12. Plandowski, W.: Satisfiability of word equations with constants is in PSPACE. J. ACM 51, 483–496 (2004)
13. Plandowski, W., Rytter, W.: Application of Lempel-Ziv encodings to the solution of word equations. In: Proc. of 25th International Colloquium on Automata, Languages, and Programming, pp. 731–742 (1998)
14. Saarela, A.: A new proof of Hmelevskii's theorem. Licentiate thesis, Univ. Turku (2009), http://users.utu.fi/amsaar/en/licthesis.pdf
15. Spehner, J.-C.: Quelques problemes d'extension, de conjugaison et de presentation des sous-monoides d'un monoide libre. Ph.D. Thesis, Univ. Paris (1976)
16. Spehner, J.-C.: Les presentations des sous-monoides de rang 3 d'un monoide libre. Semigroups. In: Proc. Conf. Math. Res. Inst., pp. 116–155 (1978)

Definability in the Infix Order on Words

Oleg V. Kudinov[1,⋆] and Victor L. Selivanov[2,⋆⋆]

[1] S.L. Sobolev Institute of Mathematics
Siberian Division Russian Academy of Sciences
kud@math.nsc.ru
[2] A.P. Ershov Institute of Informatics Systems
Siberian Division Russian Academy of Sciences
vseliv@ngs.ru

Abstract. We develop a theory of (first-order) definability in the infix partial order on words in parallel with a similar theory for the h-quasiorder of finite k-labeled forests. In particular, any element is definable (provided that words of length 1 or 2 are taken as parameters), the first-order theory of the structure is atomic and computably isomorphic to the first-order arithmetic. We also characterize the automorphism group of the structure and show that any arithmetical predicate invariant under the automorphisms of the structure is definable in the structure.

Keywords: Infix order, definability, automorphism, least fixed point, first-order theory, biinterpretability.

1 Introduction

Starting with the classical works of A. Tarski and A.I. Mal'cev, the study of definability in natural structures became a central issue of logic and computation theory. For computation theory, the study of structures on words and trees is the most relevant. In particular, many deep facts on definability and (un)decidability are known for finitely generated free semigroups and groups (see e.g. [Qu46, Ma77, KM06] and references therein).

The study of natural quasiorders on words and trees is also a traditional subject (see e.g. [Th90, Lo97, Ku06] and references therein) with several deep and interesting definability and (un)decidability results.

In [KS07, KS07a, KSZ08, KS09] many results on definability in the quotient structure $(G\mathcal{F}_k; \leq)$ of finite k-labeled forests with the h-quasiorder were established. This line of research is parallel to the popular ongoing study of definability in the degree structures of computability theory, because $(G\mathcal{F}_k; \leq)$ is isomorphic to a natural initial segment of the Wadge degrees of k-partitions of the Baire space [He93, Se07].

[⋆] Supported by DFG-RFBR Grant 06-01-04002 and by RFBR grants 08-01-00336, 07-01-00543a.

[⋆⋆] Supported by DFG-RFBR Grant 06-01-04002 and by RFBR grant 07-01-00543a.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 454–465, 2009.

In this paper we develop a similar complete definability theory for the structure $(A^*;\leq)$ where A^* is the set of words over an arbitrary finite alphabet A with at least two letters and $\leq$ is the infix partial order on A^* defined as follows: $u \leq v$ iff u is an infix of v, i.e. $v = xuy$ for some $x, y \in A^*$. If x (resp. y) in $v = xuy$ may be chosen empty, u is called a prefix (resp. a suffix) of v.

We use some standard notation on words, in particular ε denotes the empty word, $xu = x \cdot u$ denotes the concatenation of words x and u, $|u|$ denotes the length of a word u, $|u|_a$ denotes the number of occurrences of the letter $a \in A$ in the word u, $A^{[m,n]}$ denotes the set $\{u \in A^* \mid m \leq |u| \leq n\}$. We identify letters with the corresponding one-letter words. W.l.o.g. we may assume that $A = A_k = \{0, \ldots, k-1\}$ for some $k \geq 2$.

For a given structure $\mathbf{A}$ of signature σ, a predicate on A is *definable* if it is defined by a first-order formula of signature σ (in fact, this definition is not completely precise; to get the well-known precise definition, one has to fix also a suitable list of variables, as in the definition of the operator Γ in the next section). A function on $\mathbf{A}$ is definable if its graph is definable. An element is definable if the corresponding singleton set is definable. A structure is definable if its universe and all signature predicates are definable.

Section 2 recalls some necessary facts from [KS09]. In Section 3 we show that any element of A^* is definable in the $A^{[1,2]}$-expansion of $(A^*;\leq)$ (i.e. in the expansion obtained by adding to the signature $\{\leq\}$ the constant symbols denoting words of lengths 1 or 2) and characterize the automorphism group of $(A^*;\leq)$. In Section 4 we establish an important technical coding result for the $A^{[1,2]}$-expansion of $(A^*;\leq)$. In Section 5 we establish our main definability results for the infix order, in particular we show that any arithmetical predicate on A^* which is invariant under the automorphisms of $(A^*;\leq)$ is definable in $(A^*;\leq)$.

The last result implies that the structure $(\omega;+,\cdot)$ is definable in $(A^*;\leq)$ (which improves the result in [Qu46] that $(\omega;+,\cdot)$ is definable in $(A^*;\cdot)$) and that the first-order theory $FO(A^*;\leq)$ of this structure is computably isomorphic to $FO(\omega;+,\cdot)$ (which improves the result in [Ku06] that $FO(A^*;\leq)$ is undecidable). These results show that the infix partial order has the maximal possible complexity with respect to definability and first-order theory which is in contrast with the classical theorem of M. Rabin [Ra69] stating that the prefix (and hence also the suffix) partial order has a decidable first-order (even monadic second-order) theory.

In computability theory, people actively discuss several versions of the so called biinterpretability conjecture stating that some structures of degrees of unsolvability are biinterpretable (in parameters) with $(\omega;+,\cdot)$ (see e.g. [Ni00] and references therein). The conjecture (which seems still open for the most important cases) is considered as in a sense the best possible definability result about degree structures. This paper and the paper [KS09] show that some natural structures on words and forests are biinterpretable (even without parameters) with $(\omega;+,\cdot)$. We believe that our methods could be applied for proving such kind of results for many other structures including those considered in [Ku06].

2 Preliminaries on Gandy Theorem and Definability

Here we recall notions and results from [KS09] which are frequently used below.

Along with first-order definability, we are interested in definability by formulas of special kind related to admissible set theory [Ba75, Er96]. Let σ be a finite signature containing a binary relation symbol $\leq$ and possibly some other relational or constant symbols. *RQ-Formulas* of σ are constructed from the atomic formulas according to the usual rules of first-order logic concerning $\wedge, \vee, \neg, \rightarrow$, the usual rules for the (unbounded) quantification with $\forall, \exists$ and the following additional formation rules for the bounded quantification: if φ is an *RQ*-formula and x, y are variables then the expressions $\forall x \leq y\varphi$ and $\exists x \leq y\varphi$ are *RQ*-formulas. As for the usual first-order formulas, any *RQ*-formula is equivalent to a *special* *RQ*-formula (i.e., a formula without implications that has negations only on the atomic formulas).

Δ_0-*Formulas* of signature σ are constructed inductively according to the following rules: any atomic formula of signature σ is a Δ_0-formula; if φ and ψ are Δ_0-formulas then $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ and $(\varphi \rightarrow \psi)$ are Δ_0-formulas; if x, y are variables and φ is a Δ_0-formula then $\forall x \leq y\varphi$ and $\exists x \leq y\varphi$ are Δ_0-formulas. Σ-*Formulas* of signature σ are constructed inductively according to the following rules: any Δ_0-formula is a Σ-formula; if φ and ψ are Σ-formulas then $(\varphi \wedge \psi)$ and $(\varphi \vee \psi)$ are Σ-formulas; if x, y are variables and φ is a Σ-formula then $\forall x \leq y\varphi$, $\exists x \leq y\varphi$ and $\exists x\varphi$ are Σ-formulas.

A predicate on a σ-structure A is Δ_0-*definable* (resp. Σ-*definable*) if it is defined by a Δ_0-formula (resp. by a Σ-formula). A predicate on A is Δ-*definable* if both the predicate and its negation are Σ-definable.

Let P be an n-ary predicate symbol not in σ, φ an *RQ*-formula of signature $\sigma \cup \{P\}$, and $\bar{x} = x_1, \ldots, x_n$ a list of variables that includes all free variables of φ. We say that P *occurs positively in* φ if φ is in negation normal form and has no subformulas $\neg P(y_1, \ldots, y_n)$. For any n-ary predicate Q on a σ-structure $\mathbf{A}$, we denote by $(\mathbf{A}, Q)$ the expansion of $\mathbf{A}$ to the $\sigma \cup \{P\}$-structure where P is interpreted as Q. Then we can define the operator $\Gamma = \Gamma_{\varphi,\bar{x}}$ on the n-ary predicates on A that sends any Q to the predicate

$$\Gamma_{\varphi,\bar{x}}(Q) = \{(a_1, \ldots, a_n) \mid (\mathbf{A}, Q) \models \varphi(a_1, \ldots, a_n)\}.$$

A σ-structure $\mathbf{A}$ is *bounded* iff $\leq$ is a transitive directed relation (directed means that for all $x, y \in A$ there is $z \in A$ with $x, y \leq z$) and, for any Δ_0-formula φ, $\mathbf{A} \models \forall x \leq t\exists y\varphi$ implies $\mathbf{A} \models \exists v\forall x \leq t\exists y \leq v\varphi$.

For a Σ-formula φ and a variable u not occurring in φ, let φ^u be the Δ_0-formula obtained from φ by substituting any occurrence $\exists x$ of unbounded existential quantifier in φ by the corresponding bounded existential quantifier $\exists x \leq u$. The next fact is essentially from [Er96].

Proposition 1. *Let* $\mathbf{A}$ *be a bounded* σ*-structure and* P *an* n*-ary predicate symbol that occurs positively in a* Σ*-formula* φ *with the free variables among* $\bar{x}$.

(i) The operator Γ *is monotone, i.e.,* $Q \subseteq R \subseteq A^n$ *implies* $\Gamma(Q) \subseteq \Gamma(R)$.

(ii) The operator Γ sends Σ-predicates to Σ-predicates.

(iii) The symbol P occurs positively in the Σ-formula $\exists u\varphi^u$ which is equivalent to φ in $\mathbf{A}$ and hence induces the same operator Γ.

By Proposition 1(i) and Tarski fixed-point theorem, the operator Γ has the least fixed point denoted by $LFP(\Gamma)$. In general, the least fixed points defined in this way may be complicated. But for some structures $\mathbf{A}$ it turns out that the least fixed point of any Σ-formula φ as above is a Σ-predicate (in this case we say that $\mathbf{A}$ has the *Gandy property*). Let us recall a sufficient condition for $\mathbf{A}$ to have the Gandy property established in [KS09].

We say that a σ-structure $\mathbf{A}$ *admits a Δ-coding of finite sets* if there is a binary Δ-predicate $E(x, y)$ on $\mathbf{A}$ such that $E(x, y)$ implies $x \leq y$ for all $x, y \in A$ and, for all $n < \omega$ and $x_1, \ldots, x_n \in A$, there is $y \in A$ with

$$\mathbf{A} \models \forall x(E(x, y) \leftrightarrow x = x_1 \vee \cdots \vee x = x_n).$$

Informally, y is considered as a code of the set $\{x \mid E(x, y)\}$. The definition requires the existence of at least one code for any finite subset of A, but does not require that the code is unique.

Observe that, by the axiom of choice, there is a sequence $\{set^n\}_{n<\omega}$ of functions $set^n : A^n \to A$ which codes the finite sets in the sense that, for all $n < \omega$ and $x, x_1, \ldots, x_n \in A$ we have: $x_1, \ldots, x_n \leq set^n(x_1, \ldots, x_n)$ and $E(x, set^n (x_1, \ldots, x_n)) \leftrightarrow x \in \{x_1, \ldots, x_n\}$ (in particular, set^0 is an element of A such that $E(x, set^0)$ is false for all $x \in A$).

We call a σ-structure $\mathbf{A}$ *locally finite* if $\{x \mid x \leq y\}$ is finite for each $y \in A$. The next result is useful for understanding definability in some structures.

Theorem 1 ([KS09]). *Let $\mathbf{A}$ be a bounded locally finite σ-structure that admits a Δ-coding of finite sets and a Δ-definable copy of $(\omega; \leq)$. Then $\mathbf{A}$ has the Gandy property.*

Recall that a structure $\mathbf{A}$ equipped with a numbering α (i.e., a surjection from ω onto A) is *arithmetical*, if the equality predicate and all signature predicates are arithmetical modulo α. Obviously, any definable predicate on an arithmetical structure $(\mathbf{A}; \alpha)$ is arithmetical (w.r.t. α) and invariant under the automorphisms of $\mathbf{A}$; we say that $(\mathbf{A}; \alpha)$ has the *maximal definability property* if the converse is also true, i.e., any arithmetical predicate invariant under the automorphisms of $\mathbf{A}$ is definable.

Let again $\mathbf{A}$ be a countable σ-structure and let α be a numbering of A. We say that *the elements of* $(\mathbf{A}; \alpha)$ *are uniformly Σ-definable* if there is an arithmetical sequence of unary Σ-formulas $\{\psi_n(v_0)\}$ such that ψ_n defines the element $\alpha(n)$ in $\mathbf{A}$ for each $n < \omega$.

Recall (cf. [Ho93, Ni00]) that a structure $\mathbf{B}$ of a finite relational signature τ is *biinterpretable* with a structure $\mathbf{C}$ of a finite relational signature ρ if $\mathbf{B}$ is definable in $\mathbf{C}$ (in particular, there is a bijection $f : B \to B_1$ on a definable set $B_1 \subseteq C^m$ for some $m \geq 1$ which induces an isomorphism on the τ-structure $\mathbf{B}_1$ definable in $\mathbf{C}$), $\mathbf{C}$ is definable in $\mathbf{B}$ (in particular, there is a similar bijection $g : C \to C_1$

on a definable set $C_1 \subseteq B^n$ for some $n \geq 1$), the function $g^m \circ f : B \to B^{nm}$ is definable in $\mathbf{B}$ and the function $f^n \circ g : C \to C^{mn}$ is definable in $\mathbf{C}$.

Theorem 2 ([KS09]). *Let $(\mathbf{A};\alpha)$ be an arithmetical σ-structure with the uniformly Σ-definable elements such that $\mathbf{A}$ is bounded, locally finite and admits a Δ-coding of finite sets and a Δ-definable copy $(N;\preceq)$ of $(\omega;\leq)$. Then $\mathbf{A}$ has the maximal definability property and is biinterpretable with $(\omega;+,\cdot)$.*

3 Defining Elements and Characterizing Automorphisms

The first main result of this section is the following

Theorem 3. *The elements of A^* are uniformly Σ-definable in the $A^{[1,2]}$-expansion of $(A^*;\leq)$.*

Proof.[1] For $x, y \in A^*$, let $S(x,y)$ mean that $x \in \{ay, ya\}$ for some $a \in A$. Since $S(x,y)$ is equivalent to $y < x \wedge \neg\exists z \leq x(y < z < x)$, S is a Δ_0-predicate.

We show by induction on $|w|$ that any word $w \in A^*$ is Σ-definable in the $A^{[1,2]}$-expansion of $(A^*;\leq)$. For $|w| \leq 2$ the assertion is obvious, so assume that $|w| \geq 3$ and consider the representation $w = aw_1b$ where $a, b \in A$ and $w_1 \in A^+$. By induction, the words $u = w_1b$ and $v = aw_1$ are Σ-definable. Since for any $c \in A$ the word aac is Σ-definable (the case $c = a$ is easy, and for $c \neq a$ we have: $x = aac$ iff $S(aa,x)$, $c \leq x$ and $ca \not\leq x$), it suffices to check that $x = w$ is equivalent to

$$S(x,u) \wedge S(x,v) \wedge \exists y(S(y,x) \wedge aac \leq y) \tag{1}$$

where c is the first letter of w_1. Obviously, (1) is true for $x = w$, so it remains to deduce $x = w$ from (1). Let x satisfy (1). By first and second conjuncts in (1), only the following four cases are possible: $x = a_1u = a_2v$ for some $a_1, a_2 \in A$, $x = ub_1 = vb_2$ for some $b_1, b_2 \in A$, $x = a_1u = vb_1$ for some $a_1, b_1 \in A$, $x = a_1v = ub_1$ for some $a_1, b_1 \in A$.

In the first case we have $a_1w_1b = a_2aw_1$, hence $w_1b = aw_1$. Then $a = b$ (otherwise, $w_1 = aw_2b$ for some $w_2 \in A^*$, hence $aw_2bb = aaw_2b$ and so $w_2b = aw_2$ which yields an infinite sequence $w_1 > w_2 > \cdots$, a contradiction) and therefore $x \in a^*$. This shows that first and second conjuncts in (1) determine x uniquely, hence $x = w$.

The second case is similar to the first one. In the third case we have $x = a_1w_1b = aw_1b_1$, hence $a_1 = a$, $b = b_1$ and $x = w$.

In the forth case we have $x = w_1bb_1 = a_1aw_1$. An easy induction on $|w_1|$ shows that $b = a, b_1 = a_1$, $w_1 = (a_1a)^na_1$ in case $|w_1| = 2n+1$ and $b = a_1, b_1 = a$, $w_1 = (a_1a)^{n+1}$ in case $|w_1| = 2n+2$. Hence, $x = (a_1a)^{n+1}a_1$ in case $|w_1| = 2n+1$ and $x = (a_1a)^{n+2}$ in case $|w_1| = 2n+2$. By third conjunct in (1), $aaa_1 \leq y$ for some y with $S(y,x)$. This implies that $a = a_1$, $x \in a^*$ and hence $x = w$.

[1] We thank an anonymous referee for sketching this proof which is shorter than our original proof.

Writing down (recursively) the explicit Σ-definitions of elements in the constructive proof above, we obtain a computable (hence arithmetical) sequence $\{\psi_n(v_0)\}$ of Σ-formulas witnessing that the elements of A^* are uniformly Σ-definable (w.r.t. to the natural numbering of A^*). □

Next we characterize the automorphism group $Aut(A^*;\leq)$ of the structure $(A^*;\leq)$. We start with the following immediate corollary of the last theorem.

Corollary 1. *Any automorphism of $(A^*;\leq)$ identical on $A^{[1,2]}$ is the identity automorphism.*

Let $\mathbf{S}_k$ denote the symmetric group on k elements $\{0,\dots,k-1\}$ and let $\mathbf{A}\simeq\mathbf{B}$ denote that structures $\mathbf{A}$ and $\mathbf{B}$ are isomorphic. Since for $k=1$ we have $(A_k^*;\leq)\simeq(\omega;\leq)$, $Aut(A_k^*;\leq)$ is the trivial one-element group. For $k\geq 2$, along with the identity automorphism e the group $Aut(A_k^*;\leq)$ has also some other elements, in particular the reverse automorphism r defined by $r(i_1\cdots i_n)=i_n\cdots i_1$ for all $n\geq 0$ and $i_1,\dots,i_n<k$ (note that $r\circ r=e$).

Lemma 1. *Let $k\geq 2$ and let f be an automorphism of $(A_k^*;\leq)$ such that $f(i)=i$ and $f(ij)=ji$ for all $i,j<k$. Then $f=r$.*

Proof. The function $f\circ r$ is an automorphism of $(A_k^*;\leq)$ identical on $A_k^{[1,2]}$. By Corollary 1, $f\circ r=e$, hence $f=f\circ r\circ r=r$. □

Theorem 4. *For any $k\geq 2$, $Aut(A_k^*;\leq)\simeq\mathbf{S}_k\times\mathbf{S}_2$.*

Proof. The restriction map $f\mapsto f|_{A_k}$ is easily seen to be a group homomorphism from $Aut(A_k^*;\leq)$ onto $\mathbf{S}_k$. We check that the kernel K of this homomorphism coincides with $\{e,r\}$. Obviously, $e,r\in K$. Conversely, let $f\in K$, then $f(i)=i$ for all $i<k$. Since $\{01,10\}$ is the set of minimal elements in $(\{u\in A_k^*\mid 0,1<u\};\leq)$, $f(01)\in\{01,10\}$. We distinguish two cases.

Case 1. $f(01)=01$. Then of course $f(10)=10$. We show that in fact $f(ij)=ij$ for all $i,j<k$. For $i=j$ this is obvious, so let $i\neq j$. Assume first that $0\in\{i,j\}$, say $0=i$ (the case $0=j$ is treated similarly). As above, $f(0j)\in\{0j,j0\}$. Toward a contradiction, suppose that $f(0j)=j0$. For the word $w=10j$ we have $10,0j\leq w$, hence $10=f(10)\leq f(w)$ and $j0=f(0j)\leq f(w)$. But $|f(w)|=3$ (because any automorphism of $(A_k^*;\leq)$ obviously preserves the length of words), hence $f(w)$ cannot be above both $10,j0$; a contradiction.

The case $1\in\{i,j\}$ is symmetric, so it remains to consider the case $0,1\notin\{i,j\}$. Since $f(0i)=0i$ and $i\in\{i,j\}$, $f(ij)=ij$ by taking i in place of 1 in the above argument. We have shown that f is identical on $A_k^{[1,2]}$. By Corollary 1, $f=e$.

Case 2. $f(01)=10$. By the argument of case 1 one can show that in fact $f(ij)=ji$ for all $i,j<k$. By Lemma 1, $f=r$. This completes the proof of equality $K=\{e,r\}$.

Let $g\mapsto\tilde{g}$ be the embedding of $\mathbf{S}_k$ into $Aut(A_k^*;\leq)$ defined by $\tilde{g}(i_1\cdots i_n)=g(i_1)\cdots g(i_n)$ for all $n\geq 0$ and $i_1,\dots,i_n<k$ (note that $\tilde{g}|_{A_k}=g$ for each $g\in\mathbf{S}_k$). Then $\tilde{\mathbf{S}}_k=\{\tilde{g}|g\in\mathbf{S}_k\}$ and K are subgroups of $Aut(A_k^*;\leq)$, $\tilde{\mathbf{S}}_k\cap K=\{e\}$, and r commutes with any element of $\tilde{\mathbf{S}}_k$. Then each element of $Aut(A_k^*;\leq)$ is

uniquely representable in the form $\tilde{g} \circ h$ where $g \in \mathbf{S}_k$ and $h \in K$. Therefore, $Aut(A_k^*; \leq) \simeq \tilde{\mathbf{S}}_k \times K$. This completes the proof because $\tilde{\mathbf{S}}_k \simeq \mathbf{S}_k$ and $K \simeq \mathbf{S}_2$. □

Remark. Note that $Aut(A_k^*; \leq)$ is slightly bigger than $Aut(A_k^*; \cdot) \simeq \mathbf{S}_k$.

4 Coding the Finite Sets

Here we establish the following important technical fact.

Theorem 5. *For any $k \geq 2$, the $A_k^{[1,2]}$-expansion of $(A_k^*; \leq)$ admits a Δ-coding of finite sets.*

We begin with defining some auxiliary predicates and functions on A_k^* and establishing necessary lemmas.

First note that for all distinct $i, j < k$ the sets i^*, i^*j, ji^*, i^*j^*, ji^*j, and similar sets with $*$ replaced by $+$, are Δ-definable (we use the notation in the style of regular expressions). E.g., $x \in ji^+j$ iff $x \in \{i,j\}^*$, $ji \leq x$, $ij \leq x$, $jj \not\leq x$ and $iji \not\leq x$; by Theorem 3, ji^+j is a Δ-set.

Define the binary Δ_0-predicates S, L on A_k^* as follows: $S(x,y)$ iff $y < x$ and $\neg\exists z < x(y < z)$, $L(x,y)$ iff $x \leq y$ and $[x,y] = \{z \mid x \leq z \leq y\}$ is linearly ordered under $\leq$. Note that $x \leq x_1 \leq y$ and $L(x,y)$ imply $L(x_1,y)$ and that $L(\varepsilon, x)$ iff $x \in i^*$ for some $i < k$.

Lemma 2. *Let $x < y$ and $y \in A_k^*$ has at least two distinct letters. Then $L(x,y)$ iff $y = uxv$ for unique $u, v \in A_k^*$, and exactly one of u, v is empty. In particular, if $L(x,y)$ then x is a prefix or a suffix of y.*

Proof. From right to left, the assertion is obvious. Conversely, let $L(x,y)$, in particular $y = uxv$ for some $u, v \in A_k^*$. First we check that one of u, v is necessarily empty. Suppose the contrary, then $ix = xj$ for some $i, j < k$. Then $i = j$ (otherwise, $x = ix_1j$ and $ix_1 = x_1j$, hence we obtain an infinite chain $x > x_1 > \cdots$ which is a contradiction). Moreover, by a similar argument $x \in i^+$. Since y has two distinct letters, there is x_1 having two distinct letters with $x < x_1 < y$. By the argument in parenthesis we obtain a contradiction. It remains to show that x cannot be simultaneously a prefix and a suffix of y. Suppose the contrary. If $y = xux$ for some nonempty $u \in A_k^*$, one easily finds incomparable elements in $[x,y]$. If there is no such u, x must contain distinct letters and xi is incomparable with jx (where $i, j < k$ satisfy $y = xiv$ and $y = wjx$ for some $v, w \in A_k^*$). □

Let d_l (resp. d_r) be the unary function on A_k^* that deletes the first (resp. the last) letter of a given nonempty word, and let $d_l(\varepsilon) = d_r(\varepsilon) = \varepsilon$.

Lemma 3. *The functions d_l and d_r are Δ-definable.*

Proof. By symmetry, it suffices to prove the assertion for d_l, i.e. to show that the graph $w = d_l(x)$ is a Σ-predicate. Simplifying notation, we stick to the crucial particular case of binary alphabet $A_k = \{0, 1\}$. It suffices to check that $w = d_l(x)$

is equivalent to $x = w = \varepsilon \vee \phi$ where ϕ is the conjunction of the Δ_0-predicate $S(x,w)$ and disjunction of the following Σ-predicates:

$$w \in 0^* \wedge x \in 0^* \cup 10^*,\ w \in 1^* \wedge x \in 1^* \cup 01^*,$$
$$w \in 0^+1 \wedge x \in 0^+1 \cup 10^+1,\ w \notin 0^* \cup 1^* \cup 0^+1 \wedge \psi$$

where ψ is the following formula:

$$\exists y, y_1, t, z(S(y_1, y) \wedge y_1 \in 0^* \wedge S(t, y) \wedge t \in 0^*1 \wedge$$
$$y \not\leq w \wedge y_1 \not\leq z \wedge L(y, z) \wedge L(t, z) \wedge L(w, z) \wedge L(x, z)).$$

In case $w \in 0^* \cup 1^* \cup 0^+1$ the equivalence is obvious, so it remains to assume $w \notin 0^* \cup 1^* \cup 0^+1$ and show that $w = d_l(x)$ is equivalent to ψ. Let n be the least number with $0^n \not\leq w$. From left to right, we distinguish two cases.

Case 1. $x = 0w$. Set $z = 0^n x$, then $z = 0^n 0w = 0^n 00^l 1w_1$ for some $l \geq 0$ and $w_1 \in k^*$. Then the values $y = 0^n 00^l$, $y_1 = y0$ and $t = y1$ make formula ψ true.

Case 2. $x = 1w$. Set $y = 0^n$, $y_1 = y0$ and $t = y1$. If $w \in 1A_k^*$, set $z = yx$. Then $z = 0^n 111^l 0w_1$ for some $l \geq 0$ and $w_1 \in k^*$, hence ψ is true. Let now $w \in 0A_k^*$. Since $w \notin 0^* \cup 1^* \cup 0^*1$, either $w = 00^l 10w_1$ or $w = 00^l 11w_1$. Setting $z = y1x = 0^n 1100^l 10w_1$ in the first alternative and $z = yx = 0^n 100^l 11w_1$ in the second alternative, we see that ψ is true.

Conversely, let ψ be true and let y, y_1, t, z be satisfying values, in particular $y = 0^m$ for some $m \geq 1$, $y_1 = y0$ and $t = y1$. Since $S(x,w)$, it suffices to show that w is a suffix of x. Suppose the contrary, then $x = wi$ for some $i < k$. Since $L(y,z)$ and $L(t,z)$, by Lemma 2 y, t cannot both be suffixes of z, hence y is a prefix of z. Since $L(w,z)$, $L(x,z)$ and neither w nor x can be a prefix of z, by Lemma 2 both w, x are suffixes of z. Therefore, w is a suffix of x. □

For any $i < k$, define the binary predicate P_i on A_k^* by

$$P_i(v, x) \leftrightarrow v \in i^+ \wedge v \leq x \wedge iv \not\leq x \wedge \neg L(v, x) \wedge L(v, d_l(x)) \wedge L(v, d_r(x)).$$

Define also the binary predicate Q on A_k^* by

$$Q(v,x) \leftrightarrow v \in 1^+0^+ \wedge v \leq x \wedge 1v \not\leq x \wedge v0 \not\leq x \wedge \neg L(v,x) \wedge L(v, d_l(x)) \wedge L(v, d_r(x)).$$

From Lemmas 2 and 3 it follows that P_i and Q are Δ-predicates with the following properties:

Lemma 4. *(i) Let $i < k$, $v, x \in A_k^*$ and x has at least two distinct letters. Then $P_i(v,x)$ iff $v \in i^+$ and $x = vx_1v$ for some $x_1 \in A_k^* \setminus (iA_k^* \cup A_k^* i)$ with $v \not\leq x_1$.*

(ii) $Q(v,x)$ iff $v \in 1^+0^+$ and $x = vx_1v$ for some $x_1 \in A_k^* \setminus (0A_k^* \cup A_k^* 1)$ with $v \not\leq x_1$.

For any $x \in A_k^*$ and $s \geq |x| + 4$, let $\tilde{x} = 01x0$ and let $p_s(x)$ be the word in $0^*\tilde{x}0^*$ that has exactly s zeros at the beginning and the end.

Lemma 5. *For any $x \in A_k^*$ and $s \geq |x| + 4$, $\tilde{x}$ is the greatest element (under $\leq$) in the set of words $z \in A_k^* \setminus (0^*1^* \cup 1^*0^*)$ that are below any of $1^s p_s(\tilde{x}) 1^s$, $1^s \tilde{x} 1^s$, $1^s 0\tilde{x} 1^s$, $1^s \tilde{x} 01^s$.*

Proof. Obviously, $\tilde{x}$ satisfies all the conditions on z. It remains to check that $z \leq \tilde{x}$ for any z as in the formulation. Simplifying notation, we stick to the case $k = 2$ and distinguish four cases depending on the first and the last letters of z. If $z \in 0A_k^*0$ then $z \leq 1^s\tilde{x}1^s$ implies $z \leq \tilde{x}$. Assume $z \in 1A_k^*1$ and fix an infix embedding of z into $1^s p_s(\tilde{x})1^s$. If the first 1 of z is in the prefix 1^s of $1^s p_s(\tilde{x})1^s$ then the last 1 of z has to be to the right of the prefix 0^s of $p_s(\tilde{x})$, hence $|z|_0 \geq s$. This contradicts to $z \leq 1^s\tilde{x}1^s$ because the last word has at most $|x| + 2$ zeros. A similar contradiction is obtained in case when the last 1 of z is in the suffix 1^s of $1^s p_s(\tilde{x})1^s$. Hence, $z \leq p_s(\tilde{x})$ and therefore $z \leq \tilde{x}$.

If $z \in 1A_k^*0$ then $z = 11^l00^m1w0$ for some $l, m \geq 0$ and $w \in A_k^*$. Fix infix embeddings of z in $1^s\tilde{x}1^s$ and $1^s0\tilde{x}1^s$. Suppose that in both embeddings the first 1 of z is in the prefixes 1^s, then it has the same position in both prefixes 1^s, hence the 1 before w in z also has the same position in any of $1^s\tilde{x}1^s$, $1^s0\tilde{x}1^s$. But this is impossible because of the distinct number of zeros in $1^s\tilde{x}1^s$ and $1^s0\tilde{x}1^s$ between those 1's. Hence, in at least one of the embeddings the first 1 of z is in $\tilde{x}$ and therefore $z \leq \tilde{x}$. The remaining case $z \in 0A_k^*1$ is symmetric to the previous case. □

Next we explain informally our coding of finite sets. We define $set^0() = \varepsilon$ and, for $n \geq 1$ and $x_1, \ldots, x_n \in A_k^*$,

$$y = set^n(x_1, \ldots, x_n) = 1^s c(x_1) \cdots c(x_n) 0^s \tag{2}$$

where $s = max\{|x_1|, \ldots, |x_n|\} + 4$ and $c(x) = p_s(\tilde{x})1^s\tilde{x}1^s0\tilde{x}1^s\tilde{x}01^s$. The minimal infixes of y of the form $1^s0^sw1^s0^s$ coincide with the words $1^sc(x_i)0^s$ for $1 \leq i \leq n$, and they may be recovered from y by Lemma 4(ii). The minimal infixes of $1^sc(x_i)0^s$ of the form 1^sw1^s coincide with $1^sp_s(\tilde{x}_i)1^s$, $1^s\tilde{x}_i1^s$, $1^s0\tilde{x}_i1^s$, $1^s\tilde{x}_i01^s$, and they may be recovered from $1^sc(x_i)0^s$ by Lemma 4(i). From the words $1^sp_s(\tilde{x}_i)1^s$, $1^s\tilde{x}_i1^s$, $1^s0\tilde{x}_i1^s$, $1^s\tilde{x}_i01^s$ we may recover $\tilde{x}_i$ by Lemma 5. Finally, $x_i = d_ld_ld_r(\tilde{x}_i)$.

To realize this idea, we need one more lemma. Let f_0, f_1, g be the unary functions on A_k^* defined as follows: $f_i(x)$ is the largest word in i^* below x, and $g(x) = f_1(x)f_0(x)$. Note that if y is as in (2) then $f_0(y) = 0^s$, $f_1(y) = 1^s$ and $g(y) = 1^s0^s$. Since $L(1^m, 1^m0^n)$ and $L(0^n, 1^m0^n)$ we easily obtain

Lemma 6. *The functions f_0, f_1, g are Δ-definable.*

Proof of Theorem 5. We have to find a Δ-predicate $E(x, y)$ with the properties stated in Section 2. Let $\exists^{max} t \leq uR(t, \ldots)$ abbreviate

$$\exists t \leq u(R(t, \ldots) \wedge \neg\exists t_1 \leq u(R(t_1, \ldots) \wedge t < t_1)).$$

Let $E(x, y)$ be the Δ-predicate

$$\exists u \leq y(Q(g(y), u) \wedge \exists^{max} t \leq u(t \notin (0^*1^* \cup 1^*0^*) \wedge$$
$$\forall z \leq u(P_1(f_1(y), z) \to t \leq z) \wedge x = d_ld_ld_r(t)).$$

From the remarks above it follows that if y is as in (2) then $E(x, y)$ iff $x \in \{x_1, \ldots, x_n\}$. □

5 Main Results

Now we are ready to establish the following main result of this paper.

Theorem 6. *The $A^{[1,2]}$-expansion of $(A^*; \leq)$ has the Gandy property, the maximal definability property and is biinterpretable with $(\omega; +, \cdot)$.*

Proof. The expansion is clearly bounded and locally finite. By Theorem 3, the elements of A^* are uniformly Σ-definable. The structure $(0^*; \leq)$ is a Δ-definable copy of $(\omega; \leq)$. By Theorem 5, the expansion admits a Δ-coding of finite sets. Thus all conditions of Theorems 1 and 2 are satisfied. Conclusions of these theorems give the desired properties. □

We formulate an immediate consequence of the main theorem:

Corollary 2. *The structure $(\omega; +, \cdot)$ is definable in the $A^{[1,2]}$-expansion of the structure $(A^*; \leq)$. Therefore, the first-order theory of the $A^{[1,2]}$-expansion of $(A^*; \leq)$ is computably isomorphic to $FO(\omega; +, \cdot)$.*

We conclude the paper with the complete characterization of the definable predicates on $(A^*; \leq)$.

Theorem 7. *For any $k \geq 2$, the structure $(A_k^*; \leq)$ has the maximal definability property.*

Proof. Let $S(x, y)$ denote the predicate "$y < x$ and $\neg\exists z < x(y < z)$" on A_k^*. Let $\{v_a\}_{a \in A_k^{[1,2]}}$ be distinct variables and let

$$\rho = \rho(v_0, \ldots, v_{k-1}, v_{00}, \ldots, v_{0(k-1)}, \ldots, v_{(k-1)0}, \ldots, v_{(k-1)(k-1)})$$

be a formula of signature $\{\leq\}$ equivalent to the conjunction of the following formulas:

$S(v_i, \varepsilon)$ for all $i < k$,
$\neg(v_i = v_j)$ for all distinct $i, j < k$,
$S(v_{ij}, v_i) \wedge S(v_{ij}, v_j)$ for all $i, j < k$,
$\neg(v_{ij} = v_{ji})$ for all distinct $i, j < k$,
$\exists x(S(x, v_{ij}) \wedge S(x, v_{jl}))$ for all pairwise distinct $i, j, l < k$.

Repeating the proof of Theorem 4 we see that ρ defines in $(A_k^*; \leq)$ the orbit $Orb(\bar{b})$ of the tuple

$$\bar{b} = (0, \ldots, k-1, 00, \ldots, 0(k-1), \ldots, (k-1)0, \ldots, (k-1)(k-1))$$

(recall that $Orb(\bar{b}) = \{f(\bar{b}) \mid f \in Aut(A^*; \leq)\}$ where $\bar{b} = (b_0, \ldots, b_n)$ and $f(\bar{b}) = (f(b_0), \ldots, f(b_n))$.

Let $P(\bar{x})$, $\bar{x} = (x_1, \ldots, x_n)$, be an n-ary arithmetical predicate on A_k^* which is invariant under the automorphisms of $(A_k^*; \leq)$. We have to show that P is definable in $(A_k^*; \leq)$. W.l.o.g. we assume that variables $x_1, \ldots, x_n$ are distinct from the variables v_a above. By Theorem 6, there is a formula $\phi(\bar{x})$ of signature $\{\leq, a\}_{a \in A_k^{[1,2]}}$ that defines P. Let ϕ_1 be the formula of signature $\{\leq\}$ obtained from ϕ by substituting the variable v_a in place of the constant symbol a for all $a \in A_k^{[1,2]}$. Finally, let θ be the formula obtained from $\rho \wedge \phi_1$ by existential quantification over the variables v_a for all $a \in A_k^{[1,2]}$. Then θ defines P in $(A_k^*; \leq)$. □

The last result implies definability in $(A_k^*; \leq)$ of many interesting predicates on words, in particular of the subword partial order [Ku06]. We think that it would be very complicated to write down an explicit "natural" first-order definition of the subword partial order in $(A_k^*; \leq)$.

A proof similar to the proof of the last theorem establishes the following strengthening of a result in [Ku06].

Corollary 3. *The theory $FO(A^*; \leq)$ is computably isomorphic to $FO(\omega; +, \cdot)$.*

From the last theorem and finiteness of any orbit $Orb(\bar{b})$ we immediately obtain the following important model-theoretic properties of $(A^*; \leq)$.

Corollary 4. *The structure $(A^*; \leq)$ is atomic and minimal.*

References

[Ba75] Barwise, J.: Admissible Sets and Structures. Springer, Berlin (1975)

[Er96] Ershov, Y.L.: Definability and Computability. Plenum, New-York (1996)

[He93] Hertling, P.: Topologische Komplexitätsgrade von Funktionen mit endlichem Bild. Informatik-Berichte 152, Fernuniversität Hagen (1993)

[Ho93] Hodges, W.: Model Theory. Cambridge Univ. Press, Cambidge (1993)

[KM06] Kharlampovich, O., Miasnikov, A.: Elementary theory of free nonabelian groups. Journal of Algebra 302, 451–552 (2006)

[KS07] Kudinov, O.V., Selivanov, V.L.: Definability in the homomorphic quasiorder of finite labeled forests. In: Cooper, S.B., Löwe, B., Sorbi, A. (eds.) CiE 2007. LNCS, vol. 4497, pp. 436–445. Springer, Heidelberg (2007)

[KS07a] Kudinov, O.V., Selivanov, V.L.: Undecidability in the homomorphic quasiorder of finite labeled forests. Journal of Logic and Computation 17, 1135–1151 (2007)

[KS09] Kudinov, O.V., Selivanov, V.L.: A Gandy theorem for abstract structures and applications to first-order definability. To appear in the LNCS volume of Computability in Europe 2009 Proceedings (2009)

[KSZ08] Kudinov, O.V., Selivanov, V.L., Zhukov, A.V.: Definability in the h-quasiorder of labeled forests. Annals of Pure and Applied Logic (2008) doi:10.1016/j.apal.2008.09.026

[Ku06] Kuske, D.: Theories of orders on the set of words. RAIRO Theoretical Informatics and Applications 40, 53–74 (2006)

[Lo97] Lothaire, M.: Combinatorics on words, Cambridge Mathematical Library. Cambridge University Press, Cambridge (1997)

[Ma77] Makanin, G.S.: The problem of solvability of equations in a free semigroup. Mat. Sbornik 103, 147–236 (1977)

[Ni00] Nies, A.: Definability in the c.e. degrees: questions and results. Contemporary Mathematics 257, 207–213 (2000)

[Qu46] Quine, W.: Concatenation as a basis for arithmetic. Journal of Symbolic Logic 11, 105–114 (1946)

[Ra69] Rabin, M.O.: Decidability of second-order theories and automata on infinite trees. Trans. Amer. Math. Soc. 141, 1–35 (1961)

[Se07] Selivanov, V.L.: Hierarchies of Δ_2^0-measurable k-partitions. Mathematical Logic Quarterly 53, 446–461 (2007)

[Th90] Thomas, W.: Automata on infinite objects. In: Handbook of Theoretical Computer Science, vol. B, pp. 133–191 (1990)

Two-Sided Bounds for the Growth Rates of Power-Free Languages

Arseny M. Shur

Ural State University

Abstract. The growth properties of power-free languages over finite alphabets are studied. A method to obtain sharp two-sided bounds of the growth rate of β-power-free languages for arbitrary rational number $\beta \geq 2$ is obtained. A table of the growth rates, calculated with the absolute error less than 10^{-5} for different particular power-free languages, is presented.

1 Introduction

The study of words and languages avoiding repetitions is one of the central topics in combinatorics of words since the pioneering work of Thue [17]. For a survey, see [3] and the references therein. A repetition is called *avoidable* on the given alphabet, if there exist infinitely many words over this alphabet having no repetition of this type. Thue proved that squares are avoidable on the ternary alphabet, while cubes and overlaps are avoidable on two letters as well. Integral powers, which are certainly the simplest repetitions, can be generalized in several ways. Among such generalizations we mention patterns, abelian powers, relational powers, and, of course, *fractional powers* (represented by their *exponents*), which are studied in this paper. An exponent of a word is the ratio between its length and its minimal period. If $\beta > 1$ is a rational number, then a word is called β-free (β^+-free) if all its factors have exponents less than β (respectively, at most β).

When a repetition is known to be avoidable, it is natural to calculate some quantitative measure of its avoidability. If ρ denotes an avoidable repetition (or, more generally, an avoidable property of words) over the alphabet Σ, then the "size" of the language $L(\rho) \subseteq \Sigma^*$ of all words avoiding ρ is an appropriate measure of avoidability of ρ. Such a size is given by *combinatorial complexity* (or *growth function*) of $L(\rho)$ which is defined as $C_{L(\rho)}(n) = |L(\rho) \cap \Sigma^n|$. Brandenburg [4] showed that the combinatorial complexities of the language of binary cube-free words and the language of ternary square-free words both have exponential growth. Restivo and Salemi [13] proved that the combinatorial complexity of the language of binary overlap-free words grows polynomially. Since then, many papers have appeared in this area, see, e.g., a survey [2] and also [10,12], where some bounds of the order of growth for particular repetition-free languages were obtained. But up to the very recent time there has been no universal method for obtaining such bounds and no efficient algorithms for computer-assisted studies.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 466–477, 2009.

In our previous work [15,16] we developed a computationally efficient universal algorithm for upper bounds of the order of growth of an arbitrary power-free language L. More precisely, we estimate the *growth rate* of L which is defined by $\alpha(L) = \limsup\limits_{n\to\infty}(C_L(n))^{1/n}$. In this paper we show that the upper bound given by this algorithm often implies a sharp lower bound. As a result, we obtain the growth rates of many β-power-free languages with a very good precision.

The idea to obtain the lower bounds for the growth rates from the upper bounds belongs to Kolpakov [10]. He was the first to obtain relatively good lower bounds for the numbers of binary cube-free words and ternary square-free words. But in Kolpakov's method the amount of computation needed to get the lower bound is much bigger than the one for the upper bound. As a consequence, really sharp upper bounds can not be used to get lower bounds. The main advantage of our method is that obtaining a lower bound from the upper one needs a computation of nearly constant size. Thus, we can use better upper bounds. As a result, we give, for example, bounds which are at least 100 times more precise than the bounds of [10] for the numbers of binary cube-free words and ternary square-free words. The second advantage of our method is its universality: we do not even need to know what language we are studying. To obtain the lower bound it is enough to know the upper bound and some parameter for which this upper bound was obtained.

We also mention the paper [1], where some lower bounds for the growth rates of *languages avoiding patterns* are given. These bounds are derived from a purely algebraic result by Golod. However, Golod's result cannot be applied to the growth rates of power-free languages.

After preliminaries we briefly recall our method for upper bounds and prove the main result (Theorem 6) about the lower bound. The numerical results are given in the last section.

2 Preliminaries

We recall necessary notation and definitions. For more background, see [6,7,11].

1. Words, languages, and automata. An *alphabet* Σ is a nonempty finite set, the elements of which are called *letters*. *Words* are finite sequences of letters. As usual, we write Σ^n for the set of all n-letter words and Σ^* for the set of all words over Σ, including the *empty word* λ. A word u is a *factor* (respectively *prefix*, *suffix*) of the word w if w can be represented as $\bar{v}u\hat{v}$ (respectively $u\hat{v}$, $\bar{v}u$) for some (possibly empty) words $\bar{v}$ and $\hat{v}$. A *factor* (respectively *prefix*, *suffix*) of w is called *proper* if it does not coincide with w. The subsets of Σ^* are called *languages* (over Σ). A language is *factorial* if it is closed under taking factors of its words.

A word w is *forbidden* for a language L if it is a factor of no word from L. The set of all minimal (with respect to the factorization order) forbidden words for a language is called the *antidictionary* of this language. A factorial language is uniquely determined by its antidictionary, and is regular if the antidictionary is also regular (in particular, finite).

A word $w \in \Sigma^*$ can be viewed as a function $\{1, \ldots, n\} \to \Sigma$. Then a *period* of w is any period of this function. The *exponent* of w is the ratio between its length and its minimal period and is denoted by $\exp(w)$. The prefix of w whose length is the minimal period of w is called the *root* of w. If $\beta > 1$ is a rational number, then w is called β-free (β^+-free) if all its factors have exponents less than β (respectively, at most β). By β-free (β^+-free) languages we mean the languages of *all* β-free (respectively β^+-free) words over a given alphabet. These languages are obviously factorial and are also called *power-free* languages. Following [4], we use only the term β-free, assuming that β belongs to the set of "extended rationals". This set consists of all rational numbers and all such numbers with a plus; the number x^+ covers x in the usual $\leq$ order. For the arithmetic operations it will be enough to view x^+ as the number $(x+\frac{1}{n})$, where n is extremely large. For example, the condition $r \leq k/2^+$ means just that $r < k/2$.

We use two basic properties of periodic words (see, e.g., [11]).

Theorem 1 (Lyndon, Schützenberger). *If the equation $xy = yz$ holds for some words x, y, and z, then there exist words $p \neq \lambda$ and q such that $x = pq$, $z = qp$, and $y = (pq)^n p$ for some nonnegative integer n.*

Theorem 2 (Fine, Wilf). *If i and j are periods of a word u and $|u| \geq i + j - \gcd(i, j)$, then $\gcd(i, j)$ is also a period of u.*

We consider *deterministic finite automata* (dfa's) with *partial* transition function. The language recognized by a dfa $\mathcal{A}$ is denoted by $L(\mathcal{A})$. We view a dfa as a digraph, sometimes even omitting the labels of edges. A *trie* is a dfa which is a tree such that the initial vertex is its root and the set of terminal vertices is the set of all its leaves. Only *directed* walks in digraphs are considered. For a dfa, the number of words of length n in the language it recognizes obviously equals the number of *accepting walks* of length n in the automaton. A dfa is *consistent*, if each vertex is contained in some accepting walk.

2. Growth rates, digraphs, and linear algebra. For an arbitrary language L, we are interested in the asymptotic behaviour of its combinatorial complexity $C_L(n)$, more precisely, in the growth rate $\alpha(L) = \limsup\limits_{n\to\infty}(C_L(n))^{1/n}$. For factorial languages, lim sup can be replaced by lim [9].

A word w is *right extendable* (resp., *extendable*) in the language L if for any n L contains a word wu such that $|u| > n$ (resp., a word uwv such that $|u|, |v| > n$). The set of all right extendable (resp., extendable) words is denoted by $\mathrm{re}(L)$ (resp., $\mathrm{e}(L)$). We use the following result of [14].

Theorem 3. *For any factorial language L, $\alpha(\mathrm{e}(L)) = \alpha(\mathrm{re}(L)) = \alpha(L)$.*

strongly connected component (scc) of a digraph G is a subgraph G' maximal with respect to inclusion such that there exists a walk from any vertex of G' to any other vertex of G'. A digraph is *strongly connected*, if it consists of a unique scc. A *trivial* scc has one vertex and no edges.

The *index* of a digraph is the maximum absolute value of the eigenvalues of its adjacency matrix. Recall some statements of the classical Perron-Frobenius

Theorem. Note that an *irreducible* adjacency matrix is exactly the adjacency matrix of a strongly connected digraph.

Theorem 4 (Perron, Frobenius). *Let M be a nonnegative matrix, and α be the maximum absolute value of an eigenvalue of M. Then*
1) α is itself an eigenvalue of M and is simple if M is irreducible;
2) M has a nonnegative eigenvector corresponding to α (called the principal eigenvector*).*

Growth rates of regular languages and indices of digraphs are closely connected. Namely, if $\mathcal{A}$ is a consistent dfa, $L = L(\mathcal{A})$, then $\alpha(L)$ equals the index of $\mathcal{A}$. A short proof of this fact can be found in [14].

3 Approximation of Power-Free Languages

3.1 Upper Bounds

An efficient universal method for obtaining upper bounds for the growth rates of power-free languages was briefly described in [16]. The details are given in [15]. We recall the main steps of this method.

First, the target power-free language L is replaced by its *m-approximation* L_m which is a regular language satisfying $L \subset L_m$. Namely, the antidictionary of L_m is the finite set of all minimal forbidden words for L with roots of length at most m. This finite antidictionary is built from the known parameters of L by some optimized search algorithm and is stored as a trie.

Second, a consistent dfa recognizing the language L_m is constructed from the obtained trie by a fast algorithm given in [5]. This algorithm is based on the textbook Aho-Corasick algorithm for pattern matching and can be represented as follows (the details of the linear-time and linear-space implementation are omitted):

Algorithm CMR.

```
Input: trie $\mathcal{T}$ recognizing the finite antidictionary $M$.
Output: dfa $\mathcal{A}$ recognizing the factorial language $L$ with the antidictionary $M$.
0. Associate each vertex in $\mathcal{T}$ with the word labeling the accepting walk ending
at this vertex.
1. Add all possible edges to $\mathcal{T}$, following the rule:
the edge $(u, v)$ with the label $c$ should be added if
    $u$ is not terminal, and
    $u$ has no outgoing edge labeled by $c$, and
    $v$ is the longest suffix of the word $uc$ which is a vertex of $\mathcal{T}$.
2. Remove all terminal vertices and mark all remaining vertices as terminal to
get $\mathcal{A}$.
```

On the final step of our method the growth rate of the language L_m is calculated. The efficiency of this calculation is guaranteed by the following theorem.

Theorem 5 **([15,16]).** *There is an algorithm which, given a consistent dfa $\mathcal{A}$ with N vertices and a number δ, $0 < \delta < 1$, calculates the growth rate of the language $L(\mathcal{A}) \subseteq \Sigma^*$ with the absolute error at most δ in time $\Theta(-\log\delta\cdot|\Sigma|\cdot N)$ using $\Theta(-\log\delta\cdot N)$ additional space.*

The details of the calculating algorithm (we call it Algorithm S) can be found in [15]. In practice, Algorithm S implemented on a modest PC processes automata with about 10^8 edges in a few minutes. So, the possibilities to get sharp upper bounds with the described method are very high. In the next subsection we show that in many cases these upper bounds can be turned into two-sided ones.

3.2 Lower Bounds

The following theorem is the main theoretical result of this paper. It explains how to use the number $\alpha(L_m)$ obtained by the method of previous subsection to get the lower bound for $\alpha(L)$.

Theorem 6. *Suppose that $\beta \geq 2$, Σ is a finite alphabet, and L is the β-free language over Σ. Further, let $m \geq 1$ be an integer, L_m be the m-approximation of L, $\mathcal{A}_m$ be the dfa constructed by Algorithm CMR and recognizing L_m. If the digraph $\mathcal{A}_m$ has a unique nontrivial scc, then $\alpha(L) > \gamma$ for any number γ such that $\gamma + \frac{1}{\gamma^{m-1}(\gamma-1)} \leq \alpha(L_m)$.*

Remark 1. The calculation of $\alpha(L_m)$ by Algorithm S includes the partition of $\mathcal{A}_m$ into scc's. Thus, we need no additional calculation to verify that $\mathcal{A}_m$ satisfies the required condition. We have no example of such a dfa with more than one nontrivial scc and believe that the nontrivial scc is unique for all these dfa's. Conjecture 1 below seems likely to hold and yields the uniqueness of the nontrivial scc immediately.

Conjecture 1. Let L be a power-free language, $u, v \in \mathrm{e}(L)$. Then $uwv \in \mathrm{e}(L)$ for some word w.

The proof of Theorem 6 requires a detailed study of the dfa $\mathcal{A}_m$. Recall that we consider the vertices of $\mathcal{A}_m$ as words. The notation $u.w = v$ below indicates that in $\mathcal{A}_m$ the walk with the label w and the beginning u ends in the vertex v. The first lemma collects some general properties of the dfa built by Algorithm CMR. Then the main combinatorial lemma and an auxiliary algebraic result follow. After this, we prove Theorem 6.

Lemma 1 **([5]).** *Let $\mathcal{A}$ be the automaton built by Algorithm CMR from a finite antidictionary M. Then*
1) each vertex of $\mathcal{A}$ is a proper prefix of a word from M;
2) if a word w labels a walk in $\mathcal{A}$ which ends in some vertex u and $|w| \geq |u|$, then u is a suffix of w;
3) if $\lambda.w = u$ in $\mathcal{A}$, then u is the longest suffix of w which is a vertex of $\mathcal{A}$.

Lemma 2. *Suppose that a word $w \in (L_m{-}L)$ has a suffix v which is a minimal forbidden word with the period j. Then the vertex $u = \lambda.w$ of the automaton $\mathcal{A}_m$ satisfies $|u| < (\beta{-}1)j$.*

Proof. The condition $w \in L_m$ implies $j > m$. Further, each vertex u of the automaton $\mathcal{A}_m$ is a proper prefix of a minimal forbidden word with the root of length at most m by definition and Lemma 1(1). Then u has period $i \leq m$, yielding $|u| < \beta i < \beta j$. By Lemma 1(2), u is a suffix of w. Hence, u is a suffix of v by the last inequality. So, u has a period j also. Therefore, if $|u| > j$, then u has two proper periods and Theorem 2 is applicable. Namely, if $|u| \geq i+j-\gcd(i,j)$, then $\gcd(i,j)$ is a period of u. Hence, the suffix u of length j is a proper power of some word. Thus, j is not the minimal period of v and v is not a minimal forbidden word. This contradiction yields

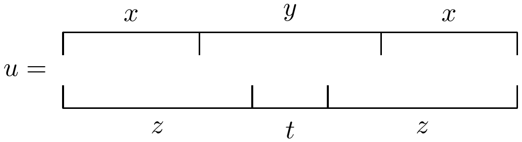
$$|u| < i + j - \gcd(i,j). \tag{1}$$

In particular, (1) implies the statement of the lemma for $\beta \geq 3$.

In the case $\beta = 2$ one has $v = rr$ for some word r. Hence, if $|u| \geq j$ then r has a proper period i. Thus, the prefix and the suffix of r of length $j{-}i$ coincide. So, the word v contains a square in the middle and is not minimal. Therefore, we have $|u| < j$, as required.

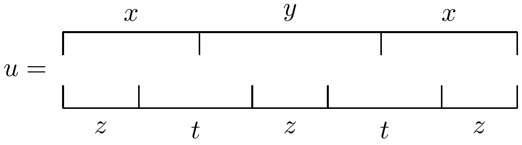
It remains to consider the case $2 < \beta < 3$. We give the proof for the case $\beta \in \mathbb{Q}$. If β is a rational number with $^+$, the argument is essentially the same. We represent v in the form $v = r'rr$, where $|r| = j$ and r' is a suffix of r. Assume that $|u| \geq (\beta{-}1)j$ and derive a contradiction. By (1), u is a proper suffix of rr. Since j is a period of u, we put $u = xyx$, where $yx = r$, $y \neq \lambda$. By assumption, $|xyx| \geq (\beta{-}1)|yx|$, yielding $|x| \geq (\beta{-}2)j$ and $|x| \geq \frac{\beta-2}{3-\beta}|y|$. Two cases for the period i are possible.

Case 1: $|u| \leq 2i$. Then $u = ztz$ and $|zt| = i$ (the word t may be empty). Since $i < j$, the mutual location of the parts of the word u looks as follows:



Since x is a prefix and a suffix of the word z, we apply Theorem 1 to get the words $p \neq \lambda$ and q such that $x = (pq)^n p$, $z = (pq)^{n+1}p$ for some integer $n \geq 0$. Then $r = yx = qpt(pq)^{n+1}p$ and hence the word rr contains the factor $(pq)^{n+2}p$. Since $\beta < 3$, we get $n = 0$, $x = p$, and the considered factor is $xqxqx$. From the inequalities $|q| < |z| - |x| < |y|$ and $|x| \geq \frac{\beta-2}{3-\beta}|y|$ we immediately have $\exp(xqxqx) > \beta$, contradicting the minimality of v.

Case 2: $|u| > 2i$. Then $u = ztztz$, $|zt| = i$, and $t \neq \lambda$, because $|u| < \beta i < 3i$. By (1), $|x| < |zt|$. On the other hand, $|z| < (\beta{-}2)i < (\beta{-}2)j \leq |x|$. Thus, the mutual location of the parts of the word u looks as follows:



Since z is a prefix and a suffix of the word x, we apply Theorem 1 to get the words $p \neq \lambda$ and q such that $z = (pq)^n p$, $x = (pq)^{n+1}p$ for some integer $n \geq 0$. Moreover,

$$t = qpt' = t''pq \tag{2}$$

for some words t' and t''. Then the word tzt has a factor $(pq)^{n+2}p$, yielding $n = 0$, $x = pqp$. Hence, $u = pqpt'pqpt'p = pt''pqpt''pqp$, $r = yx = t'pqpt'p$, and the word rr has a factor $s = pqpt'pt'p$. By (2), one of the words pq, t' is a suffix of another one. If t' is a suffix of pq, then the word s contains $(t'p)^3$, a contradiction with the minimality of v. Let $t' = \bar{t}pq$. Then $s = x\bar{t}x\bar{t}x$. Similar to case 1, from the inequalities $|y| > |\bar{t}|$ and $|x| \geq \frac{\beta-2}{3-\beta}|y|$ we deduce that $\exp(s) > \beta$. A contradiction with the minimality of v is obtained again.

So, the assumption $|u| \geq (\beta-1)j$ leads to a contradiction. The lemma is proved.

Lemma 3. *Let G be a digraph with a unique nontrivial scc, and let $\mathbf{x}$ be the principal right eigenvector of its adjacency matrix. Then the component $[\mathbf{x}]_u$ of this vector equals zero if and only if the nontrivial scc of G can not be attained from the vertex u.*

Proof. Suppose that A is the adjacency matrix of G, $\alpha \geq 1$ is its index, i.e. $A\mathbf{x} = \alpha\mathbf{x}$. From Theorem 4(1) we conclude that α is a simple eigenvalue of A, because the singleton graph has index 0. Then, by Theorem 4(2) we can assume that $\mathbf{x}$ is a unique nonnegative right principal eigenvector of A.

Let u be a vertex of G, from which the nontrivial scc is unattainable. Then the lengths of walks from u are bounded by a constant. Furthermore, if the longest walk from u has length n, then the longest walks from the successors of u in the digraph are of length at most $n-1$. Let us prove the equality $[\mathbf{x}]_u = 0$ by induction on n. If $n = 0$, then a zero row corresponds to u in the matrix A. Hence, $[A\mathbf{x}]_u = 0$ and the inductive base holds. Now move to the inductive step and take $n > 0$. Then $[A\mathbf{x}]_u = \sum[\mathbf{x}]_v$, where v runs over the set of successors of u (taking the multiplicities of edges into account). By inductive hypothesis, $[\mathbf{x}]_v = 0$. The statement is proved.

If $[\mathbf{x}]_u = 0$ for all vertices u of the nontrivial scc, then we can use the inductive argument from the previous paragraph to show that $[\mathbf{x}]_u = 0$ for all vertices u of G. But this is impossible by the definition of eigenvector. Hence, there is a vertex u of the nontrivial scc such that $[\mathbf{x}]_u > 0$. Then $[A\mathbf{x}]_v > 0$ for each predecessor v of u. Since u can be attained from every vertex, from which the nontrivial scc is attainable, we have $[A\mathbf{x}]_v > 0$ for all such vertices by induction. The lemma is proved.

Proof (of Theorem 6). Obviously, $C_L(n) = C_{L_m}(n) - C_{L_m-L}(n)$. Note that the language L_m-L consists of the words containing minimal forbidden factors with the roots of length strictly greater than m. Also note that $C_{L_m-L}(n) = 0$ for all $n < \lceil (m+1)\beta \rceil$. We put $L_u(n) = \{w \in L \cap \Sigma^n \,|\, \lambda.w = u\}$. Let $P_u(n) = |L_u(n)|$ and denote the row vector with the components $P_u(n)$ by $\boldsymbol{P}(n)$. Further, let $F_u(n) = \{wa \in (L_m - L) \cap \Sigma^n \,|\, w \in L, a \in \Sigma, \lambda.wa = u\}$, $Q_u(n) = |F_u(n)|$, and let $\boldsymbol{Q}(n)$ be the row vector with the components $Q_u(n)$.

Let us clarify the meaning of the sets $F_u(n)$. Let $w \in L_u(n)$. If $u.a = v$, then the word wa belongs either to $L_v(n{+}1)$, or to $F_v(n{+}1)$. Hence, for any vertex u one has

$$P_u(n{+}1) = \sum_{(v,u) \text{ is an edge}} P_v(n) - Q_u(n{+}1). \tag{3}$$

For vectors, (3) is replaced by

$$\boldsymbol{P}(n{+}1) = \boldsymbol{P}(n)A - \boldsymbol{Q}(n{+}1), \tag{4}$$

where A is the adjacency matrix of the automaton $\mathcal{A}_m$. The function $C_L(n)$ with the growth rate $\alpha(L)$ equals the sum of all the components of the vector $\boldsymbol{P}(n)$ or, in other words, the scalar product of $\boldsymbol{P}(n)$ and the vector $\mathbf{1} = (1, \ldots, 1)$. Hence, the growth rate of the scalar product of $\boldsymbol{P}(n)$ and any positive vector $\mathbf{x}$ is also $\alpha(L)$. Moreover, if the sum of the components of $\boldsymbol{P}(n)$ over a subset X of vertices has the growth rate $\alpha(L)$ as well, then we can take any nonnegative vector, which is positive on X, as $\mathbf{x}$.

By Theorem 3, $\alpha(L) = \alpha(\mathrm{re}(L))$. Obviously, $\mathrm{re}(L) \subseteq \mathrm{re}(L_m)$. It is also easy to see that $w \in \mathrm{re}(L_m)$ if and only if the nontrivial scc of the automaton $\mathcal{A}_m$ is attainable from the vertex $u = \lambda.w$. Hence, to obtain a function with the growth rate $\alpha(L)$ we can sum up the components of $\boldsymbol{P}(n)$ over the set of all such vertices u. Using Lemma 3, we can take the right principal eigenvector of the matrix A as $\mathbf{x}$. The obtained function $S(n) = \boldsymbol{P}(n)\mathbf{x}$ has the growth rate $\alpha(L)$, and satisfies the following equality yielding from (4):

$$S(n{+}1) = \boldsymbol{P}(n{+}1)\mathbf{x} = \boldsymbol{P}(n)A\mathbf{x} - \boldsymbol{Q}(n{+}1)\mathbf{x} = \alpha(L_m)S(n) - \boldsymbol{Q}(n{+}1)\mathbf{x}. \tag{5}$$

Now we give an upper bound for $Q_u(n{+}1)$. Consider a word $w \in F_u(n{+}1)$ which has a minimal forbidden suffix v with the period j, $m < j \leq \frac{n+1}{\beta}$. Since $\beta \geq 2$, the word w ends with two equal blocks of length j. Hence, w is uniquely determined by its own prefix w' of length $(n{+}1{-}j)$. By Lemma 2, the word u occurs in v at least twice: as a suffix and also a period (i.e., j symbols) left from this suffix. Note that both these occurrences are preceded by the same letter, because the inequality $|u| < (\beta{-}1)j$ is strict. Then the condition $\lambda.w = u$ implies $\lambda.w' = u$ by Lemma 1(3). Thus, $w' \in L_u(n{+}1{-}j)$. We proved that each element of the set $F_u(n{+}1)$ is uniquely determined by an element of exactly one of the sets $L_u(n{+}1{-}j)$, where $m < j \leq \frac{n+1}{\beta}$. So we get

$$Q_u(n{+}1) \leq \sum_{j=m+1}^{\lfloor \frac{n+1}{\beta} \rfloor} P_u(n{+}1{-}j).$$

Suppose that the number γ satisfies the condition

$$\gamma^{j-1} S(n{+}1{-}j) \leq S(n) \tag{6}$$

for all $j = m{+}1, \ldots, \lfloor \frac{n+1}{\beta} \rfloor$. Then

$$\boldsymbol{Q}(n{+}1)\mathbf{x} \leq \sum_{j=m+1}^{\lfloor \frac{n+1}{\beta} \rfloor} S(n{+}1{-}j) \leq S(n) \cdot \sum_{j=m+1}^{\lfloor \frac{n+1}{\beta} \rfloor} \frac{1}{\gamma^{j-1}} < S(n) \frac{1}{\gamma^{m-1}(\gamma - 1)}. \tag{7}$$

From (5) and (7) we get

$$S(n{+}1) > S(n)\left(\alpha(L_m) - \frac{1}{\gamma^{m-1}(\gamma-1)}\right). \tag{8}$$

Now suppose that γ satisfies the conditions of the theorem (thus, the expression in parenthesis in (8) is greater than γ). For all $n < \lceil (m{+}1)\beta \rceil - 1$ we have $\boldsymbol{Q}(n{+}1) = \mathbf{0}$, thus yielding the equation $S(n{+}1) = \alpha(L_m)S(n)$ and inequality (6). Therefore, we get $S(n{+}1) > \gamma S(n)$ for all n by induction. Indeed, for the inductive base we have $S(n{+}1) = \alpha(L_m)S(n) > \gamma S(n)$ for all $n < \lceil (m{+}1)\beta \rceil - 1$, while (6) implies (8) for the inductive step. The result now follows.

Now let us study the behaviour of the function $f(\gamma) = \gamma + \frac{1}{\gamma^{m-1}(\gamma-1)}$.

Proposition 1. *In the interval* $(1, +\infty)$*, the function* $f(\gamma)$ *has a unique minimum and no maximum.*

Proof. Since the derivative $f'(\gamma) = 1 - \frac{m\gamma - m + 1}{\gamma^m(\gamma-1)^2}$ exists in every point of $(1, +\infty)$, the points of extremum of f are among zeroes of f'. To find these zeroes we should solve the equation

$$g(\gamma) = \gamma^m(\gamma{-}1)^2 - m(\gamma{-}1) - 1 = 0. \tag{9}$$

From (9) we obtain $(\gamma{-}1) = \frac{m \pm \sqrt{m^2 + 4\gamma^m}}{2\gamma^m}$. Since $\gamma > 1$, only the plus sign in the numerator is relevant. Hence, $(\gamma{-}1) = \frac{m + \sqrt{m^2 + 4\gamma^m}}{2\gamma^m} > \frac{m}{\gamma^m}$. So, all zeroes of g in $(1, +\infty)$ satisfy the inequality $(\gamma{-}1) > \frac{m}{\gamma^m}$, while this inequality implies that g' is strictly positive:

$$g'(\gamma) = m\gamma^{m-1}(\gamma{-}1)^2 + 2\gamma^m(\gamma{-}1) - m > m\gamma^{m-1}(\frac{m}{\gamma^m})^2 + 2\gamma^m(\frac{m}{\gamma^m}) - m > m > 0.$$

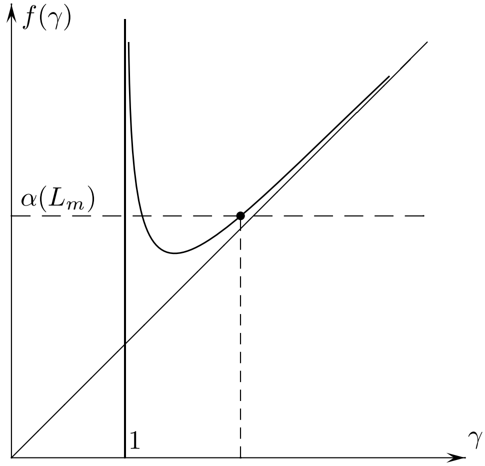


Fig. 1. *The graph of* $f(\gamma)$*. The* γ*-coordinate of the marked point is the best lower bound for* $\alpha(L)$*.*

On the other hand, $g''(\gamma) > 0$ in $(1, +\infty)$, and then $g'(\gamma)$ is increasing in this interval. Hence, the set $\{\gamma > 1 \,|\, g'(\gamma) > 0\}$ is an interval itself. Since g is increasing in the latter interval, g has a unique zero in the whole interval $(1, +\infty)$. This zero is the point of minimum of the function f, because $\lim\limits_{\gamma\to+1} f'(\gamma) < 0$ and $\lim\limits_{\gamma\to+\infty} f'(\gamma) > 0$.

By Proposition 1 and a couple of trivial observations, the graph of the function $f(\gamma)$ in the interval $(1, +\infty)$ looks like in Fig. 1. So, if the set $\{\gamma{>}1 \,|\, f(\gamma) \le \alpha(L_m)\}$ is nonempty, it is a closed interval. The rightmost point of this interval can be easily found with any precision. This point is the best lower bound which can be obtained by our method.

4 Numerical Results

In this section we present Table 1, which contains some of our results on the growth rates of power-free languages. In all cases the growth rate is found with the absolute error less than 10^{-5}. For the binary alphabet we begin with the $(7/3)^+$-free language, because it is the minimal binary power-free language of exponential growth [8]. (That is, for $2^+ \le \beta \le 7/3$ the growth rate of any binary β-free language is 1.)

We also note that the considered lower bound is reasonably good in all cases where it exists. If we take the ternary square-free language for example, then we get no lower bound for $m \le 20$, and the bound $\gamma = 1{,}2727$, which is less than

Table 1. Two-sided bounds for the growth rates of power-free languages

$\lvert\Sigma\rvert$	Exp	m	lower bound	upper bound	difference
2	$(7/3)^+$	64	1,22062891	1,2206448175	$2{\cdot}10^{-5}$
2	$(5/2)^+$	43	1,36629557	1,3663011100	$6{\cdot}10^{-6}$
2	3	36	1,45757319	1,45757728693	$4{\cdot}10^{-6}$
2	3^+	24	1,79512460	1,79512640867	$2{\cdot}10^{-6}$
2	4	21	1,82109999323	1,82109999324	$1{\cdot}10^{-11}$
2	4^+	20	1,92080153974	1,92080153975	$1{\cdot}10^{-11}$
3	2	54	1,30175907	1,3017618923	$3{\cdot}10^{-6}$
3	2^+	17	2,60587894	2,6058790806	$2{\cdot}10^{-7}$
3	3	16	2,70156143	2,7015616285	$2{\cdot}10^{-7}$
3	3^+	14	2,91192371	2,9119241945	$5{\cdot}10^{-7}$
4	2	18	2,62150791	2,6215080173	$1{\cdot}10^{-7}$
4	2^+	13	3,72849437	3,7284944228	$5{\cdot}10^{-8}$
5	2	14	3,73253856	3,7325385740	$2{\cdot}10^{-8}$
5	2^+	12	4,789850728	4,7898507379	$1{\cdot}10^{-8}$
6	2	13	4,791406948	4,7914069495	$2{\cdot}10^{-9}$
6	2^+	12	5,827732840	5,8277328410	$1{\cdot}10^{-9}$
7	2	13	5,8284660592	5,8284660593	$1{\cdot}10^{-10}$
7	2^+	12	6,8537249944	6,8537249945	$1{\cdot}10^{-10}$

0,03 away from the actual value of $\alpha(L)$, for $m = 21$. With the further growth of m the difference between the upper and lower bounds multiplicatively shrinks down to $3{\cdot}10^{-6}$ for $m = 54$ (see Table 1).

Remark 2. Slightly modifying the argument in the proof of Theorem 6 one can get even sharper lower bound for the case $\beta \geq 4$. Since $|u| < 2j$ by Theorem 2, the word w in this case is uniquely determined by its prefix of length $n{+}1{-}2j$. Then the lower bound can be ontained from the inequality $\gamma + \frac{1}{\gamma^{2m-1}(\gamma^2-1)} \leq \alpha(L_m)$. The corresponding numerical results are given in Table 1.

Remark 3. Table 1 suggests an interesting law: the growth rates of the k-ary 2^+-free language is quite close to the growth rate of the $(k{+}1)$-ary 2-free language, and the difference between these growth rates tends to 0 very quickly as k grows.

Acknowledgements

The author heartly thanks I. Gorbunova and A. Samsonov for their assistance in computational aspects of this work.

References

1. Bell, J.P., Goh, T.L.: Exponential lower bounds for the number of words of uniform length avoiding a pattern. Information and Computation 205, 1295–1306 (2007)
2. Berstel, J.: Growth of repetition-free words – a review. Theor. Comput. Sci. 340, 280–290 (2005)
3. Berstel, J., Karhumäki, J.: Combinatorics on words: A tutorial. Bull. Eur. Assoc. Theor. Comput. Sci. 79, 178–228 (2003)
4. Brandenburg, F.-J.: Uniformly growing k-th power free homomorphisms. Theor. Comput. Sci. 23, 69–82 (1983)
5. Crochemore, M., Mignosi, F., Restivo, A.: Automata and forbidden words. Inform. Processing Letters 67(3), 111–117 (1998)
6. Cvetković, D.M., Doob, M., Sachs, H.: Spectra of graphs, 3rd edn. Theory and applications. Johann Ambrosius Barth, Heidelberg (1995)
7. Gantmacher, F.R.: Application of the theory of matrices. Interscience, New York (1959)
8. Karhumäki, J., Shallit, J.: Polynomial versus exponential growth in repetition-free binary words. J. Combin. Theory. Ser. A 105, 335–347 (2004)
9. Kobayashi, Y.: Repetition-free words. Theor. Comp. Sci. 44, 175–197 (1986)
10. Kolpakov, R.: On the number of repetition-free words. In: Electronic proceedings of Workshop on words and automata (WOWA 2006), S.-Petersburg, #6 (2006)
11. Lothaire, M.: Combinatorics on words. Addison-Wesley, Reading (1983)
12. Ochem, P., Reix, T.: Upper bound on the number of ternary square-free words. In: Electronic proceedings of Workshop on words and automata (WOWA'06), S.-Petersburg, #8 (2006)
13. Restivo, A., Salemi, S.: Overlap-free words on two symbols. In: Perrin, D., Nivat, M. (eds.) Automata on Infinite Words. LNCS, vol. 192, pp. 196–206. Springer, Heidelberg (1985)

14. Shur, A.M.: Comparing complexity functions of a language and its extendable part. RAIRO Theor. Inf. Appl. 42, 647–655 (2008)
15. Shur, A.M.: Growth rates of complexity of power-free languages. Submitted to Theor. Comp. Sci (2008)
16. Shur, A.M.: Combinatorial complexity of regular languages. In: Hirsch, E.A., Razborov, A.A., Semenov, A., Slissenko, A. (eds.) Computer Science – Theory and Applications. LNCS, vol. 5010, pp. 289–301. Springer, Heidelberg (2008)
17. Thue, A.: Über unendliche Zeichenreihen, Kra. Vidensk. Selsk. Skrifter. I. Mat.-Nat. Kl., Christiana 7, 1–22 (1906)

On the Decidability of the Equivalence for a Certain Class of Transducers
(Extended Abstract)

Rodrigo de Souza

DEINFO Universidade Federal Rural de Pernambuco
Dois Irmãos, 52171-900, Recife PE, Brazil
rsouza@deinfo.ufrpe.br

Abstract. We give a new proof for the decidability of the equivalence for bounded length-degree transducers, a result established by Weber in 1992. Our proof relies on structural constructions that we have recently introduced to decompose such transducers. The complexity of our procedure is of triple exponential order, whereas the known bound is a tower of exponentials of exponential height. Furthermore, our proof deals with the more general family of transducers whose morphic image by some nonerasing morphism is a k-valued transducer.

1 Introduction

In this communication we generalise a decidability result for transducers obtained by Weber in [1] by using a structural construction we have introduced in [2] to decompose a k-valued transducer into a union of k functional ones.

A transducer is an automaton with input and output which realises a relation between words; the image of a word is the set of outputs of the successful computations reading this word. The supremum of the cardinalities of these images is called the *valuedness* of the transducer. That this parameter is interesting is evidenced by its remarkable connections with decidability questions. For instance, the equivalence is undecidable for transducers [3] (and even for the generalised sequential machines over a unary input or output alphabet [4,5]); but Schützenberger showed in [6] that the equivalence is decidable for the transducers where every image contains at most one word — the *functional transducers*.

We shall consider here the equivalence problem for a family of transducers which contains more general transducers than the k-valued ones (and, in particular, the functional ones). A transducer is of length-degree k, where k is a positive integer, if for every input word u, the *number of distinct lengths* in the image of u is at most k. Clearly, a *k-valued transducer*, where every image has *at most k words*, is of length-degree k. In [7], Weber showed that every bounded-valued transducer can be decomposed into a union of unambiguous *and* functional ones, and applied this decomposition to show that the equivalence for the bounded-valued transducers is decidable in double exponential time. Next, these two results have been generalised to the bounded length-degree transducers [8]:

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 478–489, 2009.

Theorem 1 (Weber 1992). *Every transducer $\mathcal{T}$ of bounded length-degree can be effectively decomposed into an exponential number of transducers*[1] *of length-degree* 1 *with size bounded by a double exponential on the size of $\mathcal{T}$.*

Theorem 2 (Weber 1992). *Let $\mathcal{T}$ and $\mathcal{U}$ be transducers of bounded length-degree, which realise the relations τ and γ, respectively. It is decidable whether $\tau \subseteq \gamma$ in complexity of order of a tower of exponentials of exponential height.*

The aim of the constructions we are going to describe is to extend Theorem 2 to a broader class of transducers with a better complexity.

Let us explain briefly Weber's proof for Theorem 2. It relies on the decomposition result stated in Theorem 1. The general idea is to establish an upper bound for the length of a witness for the *non-inclusion* based on properties of a decomposition of the transducers being compared. Then, the inclusion (thus, the equivalence) can be tested by checking the outputs of the words with length bounded by this witness. This upper bound is a tower of exponentials of exponential height (on the size of the transducers); it is affected by the number of transducers constructed in the decomposition (which yields the exponential height of this expression) and the size of these transducers. As observed in [8], a first attempt to improve this complexity would be to obtain a decomposition of transducers of length-degree k into exactly k transducers (instead of exponentially many ones) of length-degree at most 1. The complexity in this case would become a tower of k exponentials, that is, the height of this expression would be a function of the valuedness instead of the size of the considered transducers.

In [2,9,10], we develop a complete reworking of the theory of k-valued rational relations based entirely on structural constructions with automata. Concerning decomposition, we present in [2] a construction to decompose a k-valued transducer into a union of k functional and unambiguous ones of *single exponential size*, and explain how the same construction can be used to improve Theorem 1 and answer a more general question than the one posed by Weber in [8]:

Theorem 3 (Sakarovitch–de Souza 2008). *Let $\tau : A^* \to B^*$ be a finite image rational relation and $\theta : B^* \to C^*$ a morphism such that the composition $\tau\theta$ is k-valued.*[2] *Every transducer $\mathcal{S}$ realising τ can be effectively decomposed into k transducers of single exponential size (on the number of states of $\mathcal{S}$) whose compositions with θ are functions.*

Let us call a pair $(\mathcal{T}, \sigma)$ a *morphic transducer*. The bounded length-degree transducers are the morphic transducers such that σ is the length morphism and the composition of (the behaviour of) $\mathcal{T}$ and σ is a bounded-valued relation.

Later, we showed in [10] that the equivalence for the k-valued transducers is decidable in single exponential time by using our decomposition construction.

[1] By "decomposed" we mean that the relation realised by the transducer is equal to the union of the relations realised by the constructed ones.

[2] We write functions and relations in postfix notation: $x\tau$ is the image of x by the relation τ and thus the composition of relations is written by left-to-right concatenation. Let us recall that the rational relations are closed under composition [11].

Here we improve Theorem 2 by showing that the equivalence for the morphic bounded-valued transducers is decidable with a much smaller complexity:

Theorem 4. *Let $\mathcal{X}$ and $\mathcal{Y}$ be transducers which realise the relations τ and γ, respectively, and σ a nonerasing morphism*[3] *such that the compositions $\tau\sigma$ and $\gamma\sigma$ are k-valued relations. It is decidable in triple*[4] *exponential time on the number of states of $\mathcal{X}$ and $\mathcal{Y}$ whether $\tau \subseteq \gamma$.*

Remark that in this statement σ cannot be any morphism. Otherwise, τ and γ could be any rational relation: take for example the morphism which sends every word to the empty word; $\tau\sigma$ and $\gamma\sigma$ are of course functional. But, as we said, the equivalence is undecidable for the entire family of transducers.

Part of our proof for Theorem 4 bears clear similarity with the decidability procedure for k-valued transducers explained in [10]. In the latter, both transducers under inspection are at first decomposed into a union of functional and unambiguous transducers by using the construction we have presented in [2]: the *lexicographic decomposition*.[5] Such decomposition allow us to tackle a simpler situation. Here, our proof relies on the decomposition for the morphic k-valued transducers stated in Theorem 3, the *morphic decomposition*. Both decompositions are based on *covering of automata*. Roughly speaking, a covering is an expansion of the automaton where one can distinguish more easily between the computations (see [12] for the formal definition); the decompositions are formed by subautomata of a special covering we described in [2], the *lexicographic covering*. These coverings can be effectively constructed, and the number of states in their useful parts is bounded by a single exponential (on the number of states of the considered transducers). We recall both decompositions in Section 3.1.

The remaining of the proof is completely different from what we have done for the k-valued transducers. In [10], we show how our construction to decide the k-valuedness in polynomial time — the *Lead or Delay Valuation* — can be used together with the decomposition we have just sketched to "see" whether the transducers are equivalent. Such method does not hold anymore for the more general case of morphic transducers. The main reason is that the *morphic* 1-valued transducers built in the morphic decomposition are not necessarily functional (in spite of the fact that the compositions with θ are functional), whereas our previous algorithm relies heavily on the fact that the transducers yielded by the lexicographic decomposition are functional. Our approach is to show that, for every successful computation of each of the transducers being compared, one may find a *compatible* one (successful and with the same label) in the other transducer such that the *lag* between them is bounded by some expression on the size of the transducers. This is stated in Lemma 3; this bound is already of double exponential order, and is established with two constructions of one of the coverings described in [2]. Next, a new construction of the same covering allows to test the compatibility property. The complete proof is discussed in Section 4.

[3] Notions such as nonerasing morphism will be defined in the body of the paper.

[4] A more precise expression is given in Theorem 6 at the end of the paper.

[5] As the construction is coined in the journal version of [2].

Maybe the complexity stated in our result is not the best one could hope, and it is indeed matter for further development whether a more careful use of our constructions may lead to a double or even single exponential time complexity. On the other hand, Theorem 4 goes sufficiently far to show that these constructions allow a considerable gain with respect to Theorem 2 — this is in fact what we propose to illustrate here — in a more general setting.

Due to space constraints, the details of the proofs are omitted here, but are planned to appear in a full paper together with a complete discussion of the morphic decomposition theorem.

2 Preliminaries

We follow the definitions and notation in [11,13,14].

The set of words over a finite alphabet A (the free monoid over A) is denoted by A^*, and the empty word by 1. For free monoids A^* and B^*, a morphism from A^* into B^* is a function $\sigma : A^* \to B^*$ compatible with the operation of A^*: $1\sigma = 1$ and for every $u, v \in A^*$, $(uv)\sigma = (u\sigma)(v\sigma)$. We say that σ is *nonerasing* if $A\sigma \subseteq B^+$, that is, for every letter $a \in A$, $a\sigma$ is different from the empty word. Every morphism $\sigma : A^* \to B^*$ is completely defined by the images of the letters in A, thus in order to present a morphism we can simply exhibit these images.

Let M be a monoid. An automaton $\mathcal{A} = \langle\, Q, M, E, I, T\,\rangle$ is a labelled directed graph given by sets Q of states, I, $T \subseteq Q$ of initial and final states, respectively, and $E \subseteq Q \times M \times Q$ of transitions labelled by M. It is finite if Q and E are finite.

A *computation* in $\mathcal{A}$ is a sequence of transitions $c : p_0 \xrightarrow{m_1} p_1 \xrightarrow{m_2} \ldots \xrightarrow{m_l} p_l$, also written $c : p_0 \xrightarrow{m_1 \ldots m_l} p_l$. The *label* of c is $m_1 \ldots m_l \in M$, and c is a *successful computation* if $p_0 \in I$ and $p_l \in T$. The *behaviour* of $\mathcal{A}$ is the set $|\mathcal{A}| \subseteq M$ of labels of successful computations. The behaviour of finite automata over M coincide with the family $\operatorname{Rat} M$ of the *rational subsets* of M [11].

If M is a free monoid A^* and the labels of transitions are letters, then $\mathcal{A}$ is a (boolean) automaton over A. If M is a product $A^* \times B^*$, then every transition is labelled by an input word $u \in A^*$ and an output one $x \in B^*$ and $\mathcal{A}$ is a *transducer* realising a *rational relation* from A^* to B^*. A transducer is *unambiguous* if distinct successful computations have distinct labels.

The *image* of a word $u \in A^*$ by (the behaviour of) a transducer is the set of outputs of successful computations which *read* u. The transducer is called k*-valued*, where k is a positive integer, if, for every input word, the image has at most k words. It is *bounded-valued* if there exists such an integer k.

We shall only consider transducers which are *real-time*: every transition is labelled by a pair (a, K) formed by a letter $a \in A$ and a set $K \in \operatorname{Rat} B^*$, and I, T are functions from Q to $\operatorname{Rat} B^*$. By using classical constructions on automata, every transducer can be transformed into a real-time one. For bounded-valued transducers, we may suppose that every transition outputs a single word, and that the image of every initial or final state is the empty word.[6]

[6] A *nondeterministic generalised sequential machine* in some references.

For computations c and d of a (real-time) transducer with the same label, their *lag*, represented by $\langle c, d\rangle$, is the the maximal difference between the lengths of outputs of prefixes of c and d with the same input (see [2] for details).

We shall make systematic use of product of (real-time) transducers, see [15] for instance for details. Let us recall that in a product $\mathcal{T}_1\times\ldots\times\mathcal{T}_\ell$ of ℓ transducers over $A^*\times B^*$, every computation is labelled by a word on $A\times {B^*}^l$, and projects on ℓ computations, one for each $\mathcal{T}_i$, *with the same input.*

A *morphism* from an automaton $\mathcal{B} = \langle\, R, M, F, J, U\,\rangle$ to $\mathcal{A} = \langle\, Q, M, E, I, T\,\rangle$ is a pair of mappings, $R \to Q$ and $F \to E$ (both denoted by φ) which respect adjacency of transitions and $J\varphi \subseteq I$, $U\varphi \subseteq T$. The image by φ of every successful computation of $\mathcal{B}$ is a successful one in $\mathcal{A}$ with the same label, hence $|\mathcal{B}| \subseteq |\mathcal{A}|$. The morphism is a *covering* if it is *locally bijective.* This implies a bijection between the successful computations and thus $|\mathcal{B}| = |\mathcal{A}|$. See [12] for details.

3 Morphic Transducers

Let us start our discussion of Theorem 4 with some examples of morphic transducers and an overview of the morphic decomposition theorem (Theorem 3):

Definition 1. *A* morphic transducer *is a pair $\mathcal{M} = (\mathcal{S}, \sigma)$ where $\mathcal{S}$ is a (real-time) transducer over $A\times B^*$ and σ is a* nonerasing *morphism from B^* to some free monoid C^*. The* morphic image *of $\mathcal{S}$ is the transducer obtained by replacing the output of every transition of $\mathcal{S}$ by its image by σ. The* morphic behaviour *of $\mathcal{M}$ is the relation $|\mathcal{S}|\sigma$ — the composition of $|\mathcal{S}|$ and σ —, that is, the behaviour of the morphic image of $\mathcal{S}$ (thus a relation from A^* to C^*).*

*We say that $\mathcal{M}$ is k-*valued *if so does its morphic behaviour. We say that a transducer $\mathcal{S}$ from A^* to B^* (a rational relation τ from A^* to B^*) is* morphic k-valued *if there exists a nonerasing morphism σ from B^* to C^* such that the morphic transducer $(\mathcal{S}, \sigma)$ (the composition $\tau\sigma$) is k-valued.*

As we shall see in the examples below, not every transducer $\mathcal{S}$ admits a nonerasing morphism σ such that $|\mathcal{S}|\sigma$ is k-valued. That is,

Proposition 1. *The morphic k-valued rational relations form a proper subclass of the rational relations.*

Let us stress that given k-valued morphic transducers $(\mathcal{X}, \sigma)$ and $(\mathcal{Y}, \gamma)$, our aim is to decide the equality of the behaviour of $\mathcal{X}$ and $\mathcal{Y}$ (which are not necessarily k-valued) and not of the k-valued compositions $|\mathcal{X}|\sigma$ and $|\mathcal{Y}|\gamma$. It should also be clear that the morphisms σ and γ need to be known beforehand: we do not deal with the question whether such morphisms can be constructed starting from the transducers $\mathcal{X}$ and $\mathcal{Y}$, or whether the morphic k-valuedness is decidable. Finally, note that the equality of the relations $|\mathcal{X}|\sigma$ and $|\mathcal{Y}|\gamma$ does not imply that the transducers $\mathcal{X}$ and $\mathcal{Y}$ are equivalent.

Example 1. As said in the introduction, every bounded-valued rational relation is morphic bounded-valued (take for instance σ as the identity morphism), and the same is true for the rational relations of bounded length-degree: a bounded-valued morphic transducer is obtained by taking the length morphism.

Example 2. The synchronous transducers and the α-synchronous ones introduced in [16] are of bounded length-degree, and thus morphic bounded-valued.

Example 3. Every inverse of a nonerasing morphism is a morphic rational function (1-valued). Indeed, for a nonerasing morphism σ, the composition $\sigma^{-1}\sigma$ is the identity function (on the domain of σ^{-1}). Such inverses include relations which are not of bounded length-degree. For instance, consider the nonerasing morphism $\sigma_1 : \{a,b\}^* \to \{a\}^*$ given by $a\sigma_1 = a$ and $b\sigma_1 = aa$. For every a^n, $n \geq 0$, the lengths of the words in $a^n\sigma_1^{-1}$ range from n to $\lceil n/2 \rceil$. Thus, the number of distinct lengths in the images of σ_1^{-1} is not bounded.

Example 4. Let us exhibit a rational relation which is not morphic k-valued and thus show Proposition 1. Incidentally, we shall establish that the morphic bounded-valued rational relations are not closed by union. Consider σ_1 (the morphism defined in Example 3) and let σ_2 be the length morphism: for every u in $\{a,b\}^*$, $u\sigma_2 = a^{|u|}$. We claim that the finite image relation $\gamma = \sigma_1^{-1} \cup \sigma_2^{-1}$ is not morphic bounded-valued, that is, for every nonerasing morphism θ from $\{a,b\}^*$ to some C^*, the composition $\gamma\theta$ cannot be bounded-valued. Indeed, if $|a\theta| = |b\theta|$, then, for every a^n, the lengths of the words in $a^n(\gamma\theta)$ range from n to $\lceil n/2 \rceil$, for the image by θ of words with distinct lengths in $a^n\sigma_1^{-1}$ have distinct lengths. Otherwise (if $|a\theta| \neq |b\theta|$), the images of words in $a^n\sigma_2^{-1}$ with distinct numbers of occurrences of a have distinct lengths, and thus $a^n(\gamma\theta)$ has at least n words. We have shown that $\gamma\theta$ cannot be bounded-valued.

3.1 The Morphic Decomposition Theorem

The morphic decomposition theorem we are going to outline (Theorem 3 in the introduction) has been introduced in [2]. See also [17] for more details of the proof. With the notation of morphic transducers, it can be restated as follows:

Theorem 5 (Sakarovitch–de Souza 2008). *For every k-valued morphic transducer $\mathcal{M} = (\mathcal{S}, \sigma)$ with n states, one can construct effectively k functional (1-valued) morphic transducers $\mathcal{M} = (\mathcal{X}_0, \sigma), \ldots, (\mathcal{X}_{k-1}, \sigma)$ and morphisms (of automata) $\varphi^{(i)} : \mathcal{X}_i \to \mathcal{S}$, $1 \leq i \leq k$, where $|\mathcal{S}| = |\mathcal{X}_1| \cup \cdots \cup |\mathcal{X}_k|$. Each $\mathcal{X}_i$ has $2^{\mathcal{O}(h\ell k^4 n^{k+4})}$ states, where ℓ is the maximal length of the outputs of transitions of $\mathcal{S}$ and h is the size of the output alphabet of $\mathcal{S}$.*

The proof starts with a *lexicographic decomposition* of the morphic image of $\mathcal{S}$ (which is a k-valued transducer) into a union of k functional and unambiguous transducers. The whole process is depicted in Figure 2. Before describing it, let us say a word on the lexicographic decomposition of a k-valued transducer, which is represented in Figure 1. It is based on *covering of automata.* Roughly speaking, a covering of an automaton[7] $\mathcal{A}$ is an "expansion" of $\mathcal{A}$, a new and

[7] Recall that transducers are automata of a certain kind.

$$\begin{tikzcd}
\mathcal{T} & \mathcal{V}_N \arrow[l, dotted] \arrow[d, "\pi"'] & \bigcup_i \mathcal{Z}^{(i)} \arrow[l, Rightarrow] \arrow[d, "\pi"] \\
& \mathcal{A} & \bigcup_i \mathcal{B}_k^{(i)} \arrow[l, Rightarrow]
\end{tikzcd}$$

Fig. 1. Lexicographic decomposition of $\mathcal{T}$: $\mathcal{V}_N$ is an equivalent and input-k-ambiguous transducer yielded by the lead or delay covering; $\mathcal{A}$ is the underlying input automaton of $\mathcal{V}_N$; $\bigcup_i \mathcal{B}_k^{(i)}$ is a decomposition of $\mathcal{A}$ given by multi-skimming covering; $\bigcup_i \mathcal{Z}^{(i)}$ is the decomposition, which one obtains by "lifting" the outputs of $\mathcal{T}$ to $\bigcup_i \mathcal{B}_k^{(i)}$

larger automaton, whose successful computations are in bijection with those of $\mathcal{A}$ (details can be found in [12], where coverings are used systematically to produce new proofs for some classical properties of transducers). In [2], we have presented a general method for building coverings, which we call *lexicographic coverings*. The idea is to put a lexicographic ordering on the computations (viewed as words on the set of transitions). Next, the decomposition of a k-valued transducer amounts to an adequate choice of k subautomata in the constructed coverings. Two lexicographic coverings are used in our proof, as indicated in Figure 1: the multi-skimming covering (for $\mathbb{N}$-automata) and the lead or delay covering[8] (for transducers). A crucial step is the construction, with the latter covering, of the transducer $\mathcal{V}_N$ which is equivalent to $\mathcal{T}$ and input-k-ambiguous.

The first step of the morphic decomposition theorem is, as said, the construction of a lexicographic decomposition of the morphic image of $\mathcal{S}$ — the k-valued transducer $\mathcal{T}$ in Figure 2. Next, we show that it is possible to "stick" the computations of $\mathcal{T}$ into the decomposition in order to obtain k new functional transducers, not necessarily unambiguous, whose successful computations project onto the successful computations of $\mathcal{T}$. This operation of "sticking" is based on Property 4.5 in [2]. Roughly speaking, it states that there exists an integer K such that for every computation c in $\mathcal{T}$, one can find a computation d in at least one transducer of the decomposition with the same label and such that their *lag*, $\langle c, d \rangle$, is bounded by K. This property is established by means of a pumping argument which will also be useful in the next section to prove Theorem 4. The details can be found in [17]. Finally, the output of each transition of these new k functional transducers is replaced by the output of the projected one in $\mathcal{S}$ (by the automata morphisms maintained along the construction). The relabelled transducers form a morphic decomposition of $\mathcal{S}$.

4 The Decidability Result

Our proof of Theorem 4 and the one we have given in [10] for the equivalence of k-valued transducers share a common scheme: decompose both transducers and test some condition on the successful computations of the obtained functional

[8] "Lag separation covering" in [2].

$$\begin{tikzcd}
\mathcal{S} \arrow[r, "\sigma"] & \mathcal{T} & \bigcup_{0\leq i<k} \mathcal{Z}^{(i)} \arrow[l] \\
\bigcup_{0\leq i<k} \mathcal{X}^{(i)} \arrow[u] & & \bigcup_{0\leq i<k} \mathcal{W}^{(i)} \arrow[ll, "\sigma^{-1}"'] \arrow[ul] \arrow[u]
\end{tikzcd}$$

Fig. 2. Decomposition of a morphic k-valued transducer $\mathcal{S}$: $\mathcal{T}$ is the morphic image of $\mathcal{S}$ obtained by relabelling the outputs of the transitions, $\bigcup_{0\leq i<k} \mathcal{Z}^{(i)}$ is a lexicographic decomposition of $\mathcal{T}$, $\bigcup_{0\leq i<k} \mathcal{W}^{(i)}$ is the transducers obtained by "sticking" the computations of $\mathcal{T}$ into $\mathcal{Z}^{(0)}, \ldots, \mathcal{Z}^{(k-1)}$, $\bigcup_{0\leq i<k} \mathcal{X}^{(i)}$ is the morphic decomposition obtained from the $\mathcal{W}^{(i)}$ by relabelling back the outputs of transitions

ones. But distinct constructions are used to perform the second step. In order to explain our proof of Theorem 4, we shall at first discuss briefly the procedure presented in [10], and move progressively towards the constructions needed here.

The Equivalence for the *k*-Valued Transducers. Our proof for the equivalence of k-valued transducers combines the lexicographic decomposition (Section 3.1) and the construction we have described in [9] to decide the k-valuedness for transducers in polynomial time, the *Lead or Delay Valuation*. The decomposition allows to reduce the problem to the simpler one of deciding the inclusion of a functional transducer into a union of functional transducers. That is, the equivalence (in single exponential time) is a consequence of the following lemma:

Lemma 1. *Let $\mathcal{Z}^{(0)}, \ldots, \mathcal{Z}^{(k-1)}$ be a lexicographic decomposition of a k-valued transducer. Let $\mathcal{R}$ be a functional transducer in the lexicographic decomposition of another k-valued transducer. It is decidable in polynomial time (*on the size of $\mathcal{R}, \mathcal{Z}^{(0)}, \ldots, \mathcal{Z}^{(k-1)}$*) whether $|\mathcal{R}| \subseteq |\mathcal{Z}^{(0)}| \cup \cdots \cup |\mathcal{Z}^{(k-1)}|$.*

This lemma is established in [10] with the Lead or Delay Valuation (LDV). The LDV of a product $\mathcal{T}^{k+1}$ is a generalisation of the construction introduced in [15] to characterise the functionality, the product of $\mathcal{T}^2$ by the Lead or Delay Action. Roughly speaking, it attributes to every state of $\mathcal{T}^{k+1}$ a set of *differences*, words of the free group (over the output alphabet) which allows to compare pairs of words. For instance, two words are equal if, and only if, their difference is the empty word; see [9] for details. Then, it can be "read" in these differences whether for every successful computation of $\mathcal{T}^{k+1}$ one can find at least two projections with the same output — this is equivalent to say that $\mathcal{T}$ is k-valued.

Although some details of the Lead or Delay Valuation are delicate,[9] the general idea of our proof for Lemma 1 is quite natural. The algorithm tests *compatibility of computations*. We say that two computations (in the transducers being

[9] A complete explanation can be found in [18] and [17].

compared) are *compatible* if both are successful and have the same label (the same input and the same output). Clearly, the inclusion in Lemma 1 holds if, and only if, for every successful computation of $\mathcal{R}$, one can find a compatible one in some $\mathcal{Z}^{(i)}$. This test is performed on the cartesian product $\mathcal{R}\times\mathcal{Z}^{(0)}\times\ldots\times\mathcal{Z}^{(k-1)}$. Every computation c in this product projects on $k+1$ computation with the same input, a $\mathcal{R}$-*projection* (the projection on the first component, so, on $\mathcal{R}$) and the $\mathcal{T}$-*projections* (the ones on some transducer in the decomposition). We say that c is a *full computation* if its $\mathcal{R}$-projection is a successful computation and its $\mathcal{T}$-projections contain all the successful computations in $\mathcal{Z}^{(0)},\ldots,\mathcal{Z}^{(k-1)}$ reading the same input. It is not difficult to see that the aforementioned compatibility condition is equivalent to say that in every full computation, one can find at least one successful $\mathcal{T}$-projection whose output is equal to the output of the $\mathcal{R}$-projection (that is, the difference of these two outputs is the empty word). This can be "read" in the LDV of the product of $\mathcal{R}\times\mathcal{Z}^{(0)}\times\ldots\times\mathcal{Z}^{(k-1)}$ by the action $\circ_\mu : \mathbb{N}^Q\times A^*\to\mathbb{N}^Q$ defined in Section 5 of [10]. Roughly speaking, the accessible $\mathbb{N}$-vectors in $\circ_\mu$ "counts", for every input word u, the number of successful computations in $\mathcal{Z}^{(0)},\ldots,\mathcal{Z}^{(k-1)}$ which reads u. This information allows to distinguish the full computations of $\mathcal{R}\times\mathcal{Z}^{(0)}\times\ldots\times\mathcal{Z}^{(k-1)}$.

Another Way to Test the Equivalence. In order to introduce our proof for Theorem 4, let us describe a construction which allows to test the equivalence of k-valued transducers without the use of the LDV. The idea can be put in parallel with a method to decide the equivalence of boolean automata $\mathcal{A}$ and $\mathcal{B}$: $|\mathcal{A}|\subseteq|\mathcal{B}|$ if, and only if, for every accessible state (p,P) of the cartesian product of $\mathcal{A}$ by the determinised of $\mathcal{B}$ by the subset construction, $\mathcal{A}\times\mathcal{B}_{\mathsf{det}}$, if p is final in $\mathcal{A}$, then P contains at least one final state of $\mathcal{B}$. For k-valued transducers $\mathcal{T}$ and $\mathcal{T}'$, as the outputs have to be taken into account, we need an additional property of the computations: if $|\mathcal{T}|\subseteq|\mathcal{T}'|$, then there exists a constant K such that for every successful computation c of $\mathcal{T}$, there exists a compatible successful computation d in $\mathcal{T}'$ such that $\langle c,\, d\rangle < K$. This is made more precise in the lemma below:[10]

Lemma 2. *Let* $(\mathcal{X},\sigma)$ *and* $(\mathcal{Y},\sigma)$ *be* k*-valued morphic transducers with at most* n *states, where* $\mathcal{X}$ *and* $\mathcal{Y}$ *are transducers over* $A^*\times B^*$ *and* σ *is a morphism from* B^* *to* C^**. Let* ℓ *be the maximal length of the outputs of the transitions of* $\mathcal{X}$ *and* $\mathcal{Y}$*,* h *the size of* B*, and* $s=\max\{|b\sigma| \mid b\in B\}$*. Let* $\mathcal{R}$ *be the morphic image of some transducer in a morphic decomposition of* $\mathcal{X}$*, and* $\mathcal{W}^{(0)},\ldots,\mathcal{W}^{(k-1)}$ *the morphic images of a morphic decomposition of* $\mathcal{Y}$ *(see Figure 2). If* $|\mathcal{R}|\subseteq|\mathcal{W}^{(0)}|\cup\cdots\cup|\mathcal{W}^{(k-1)}|$*, then, for every successful computation* c *of* $\mathcal{R}$*, there exists a compatible computation* d *in some* $\mathcal{W}^{(i)}$ *such that*$\langle c,\, d\rangle\leq 2^{\mathcal{O}(hs\ell k^5 n^{k+4})}$.

The proof of Lemma 2 is based on a pumping argument on the computations of the product $\mathcal{R}\times\mathcal{W}^{(0)}\times\ldots\times\mathcal{W}^{(k-1)}\times\circ_\mu$. The same argument has also been used in [2] to establish properties of the lexicographic decomposition and the morphic decomposition (Lemma 4.3 and Property 4.5, respectively). We say that a computation in $\mathcal{R}\times\mathcal{W}^{(0)}\times\ldots\times\mathcal{W}^{(k-1)}\times\circ_\mu$ reading u is full if its $\mathcal{R}$-projection is successful, and for every i, if u belongs to the domain of $\mathcal{W}^{(i)}$,

[10] If one replaces $\mathcal{X}$ and $\mathcal{Y}$ by $\mathcal{T}$ and $\mathcal{T}'$, respectively, and σ by the identity morphism.

then the projection in $\mathcal{W}^{(i)}$ is successful (note that, as $\mathcal{W}^{(i)}$ is not necessarily unambiguous, it can contain several successful computations reading u; but as it is functional, all have the same output). The lemma states that every full computation has a pair of compatible projections (one in $\mathcal{R}$, other in some $\mathcal{W}^{(i)}$) whose lag is bounded. The pumping argument reads as follows. If one could find a full computation f where, for every pair of compatible projections, the lag is greater than the claimed bound, then there would exist a factorisation $f = f_1 f_2 f_3$ and an integer r such that f_2 is a circuit and the computation $f_1 f_2^r f_3$ contains fewer pairs of compatible projections. Such factorisations would lead to a computation having no pair of compatible projections, which contradicts the hypothesis that $|\mathcal{R}| \subseteq |\mathcal{W}^{(0)}| \cup \cdots \cup |\mathcal{W}^{(k-1)}|$. The use of the action $\circ_\mu$ is crucial: it guarantees that $f_1 f_2^r f_3$ is also full, that is, no successful computation is lost.

Lemma 2 allows to show that one can "see" in a lead or delay covering of the transducer $\mathcal{U} = \mathcal{R} \cup \mathcal{W}^{(0)} \cup \cdots \cup \mathcal{W}^{(k-1)}$ whether $|\mathcal{R}| \subseteq |\mathcal{W}^{(0)}| \cup \cdots \cup |\mathcal{W}^{(k-1)}|$, in the same way as the inclusion of boolean automata can be seen in $\mathcal{A} \times \mathcal{B}_{\mathsf{det}}$. Let $M = 2^{\mathcal{O}(hs\ell k^5 n^{k+4})}$. Adopt any lexicographic ordering of the transitions of $\mathcal{U}$ where every transition of $\mathcal{W}^{(0)} \cup \cdots \cup \mathcal{W}^{(k-1)}$ is smaller than those of $\mathcal{R}$. Let $\mathcal{W}$ be the restriction of the lead or delay covering of index M of $\mathcal{U}$ to the states whose first component is a state of $\mathcal{R}$. In $\mathcal{W}$, the second component plays the role of $\mathcal{B}_{\mathsf{det}}$: every computation C of $\mathcal{W}$ ends in some state of form (r, v), where r is a state of $\mathcal{R}$, and v is a vector indexed by the states of $\mathcal{U}$; v stores the differences (bounded by M) between the output of the projection of C on $\mathcal{R}$ and the outputs of the computations in $\mathcal{W}^{(0)} \cup \cdots \cup \mathcal{W}^{(k-1)}$ with the same input. Then, it follows from Lemma 2 that $|\mathcal{R}| \subseteq |\mathcal{W}^{(0)}| \cup \cdots \cup |\mathcal{W}^{(k-1)}|$ if, and only if, in every final state of $\mathcal{W}$, there exists at least one final state t of $\mathcal{W}^{(0)} \cup \cdots \cup \mathcal{W}^{(k-1)}$ such that v_t (the position of index t in this vector) contains the empty word.

The number of states of a lead or delay covering of index N is exponential in N and the number of states of the considered transducer (Property 3.10 in [2]). Thus, the size of $\mathcal{W}$ (and the complexity of the algorithm we have just sketched) is of double exponential order on the number of states of $\mathcal{X}$ and $\mathcal{Y}$. For the equivalence of morphic transducers we are going to explain, the triple exponential complexity comes from the construction of three lead or delay coverings: one for the morphic decomposition, two to test compatibility of computations.

The Equivalence for the Morphic Transducers. The first step of our proof for Theorem 4 is the construction of morphic decompositions for $\mathcal{X}$ and $\mathcal{Y}$, say $\mathcal{X}^{(0)}, \ldots, \mathcal{X}^{(k-1)}$ and $\mathcal{Y}^{(0)}, \ldots, \mathcal{Y}^{(k-1)}$ respectively. Then, one can decide whether $|\mathcal{X}| \subseteq |\mathcal{Y}|$ by testing, for every $\mathcal{Z}$ in the decomposition of $\mathcal{X}$, whether $|\mathcal{Z}|$ is included in $|\mathcal{Y}^{(0)}| \cup \cdots \cup |\mathcal{Y}^{(k-1)}|$. As said in the introduction, the LDV cannot be used to tackle this test as it has been used in [10]. The reason is that the condition on full computations behind the procedure for k-valued transducers does not hold anymore: it may well exist computations in $\mathcal{Z} \times \mathcal{Y}^{(0)} \times \ldots \times \mathcal{Y}^{(k-1)}$ having no pairs of compatible projections *even if* $|\mathcal{Z}|$ *is included in* $|\mathcal{Y}^{(0)}| \cup \cdots \cup |\mathcal{Y}^{(k-1)}|$, for these transducers are not necessarily functional. One could be tempted to investigate the construction of the LDV on the *morphic images* of $\mathcal{Z}$ and $\mathcal{Y}^{(0)}, \ldots, \mathcal{Y}^{(k-1)}$, say $\mathcal{R}$ and $\mathcal{W}^{(0)}, \ldots, \mathcal{W}^{(k-1)}$ respectively. But this does not work either, for

$|\mathcal{R}| \subseteq |\mathcal{W}^{(0)}| \cup \cdots \cup |\mathcal{W}^{(k-1)}|$ does not imply $|\mathcal{Z}| \subseteq |\mathcal{Y}^{(0)}| \cup \cdots \cup |\mathcal{Y}^{(k-1)}|$. So we shall adapt Lemma 2 to establish that, if $|\mathcal{Z}| \subseteq |\mathcal{Y}^{(0)}| \cup \cdots \cup |\mathcal{Y}^{(k-1)}|$, then, for every successful computation c of $\mathcal{Z}$, there exists a compatible one d in some $\mathcal{Y}^{(i)}$ such that $\langle c,\, d\rangle$ is bounded by an expression on the size of $\mathcal{X}$ and $\mathcal{Y}$.

At first, we need to adapt the definition of full computations: let c be a computation in $\mathcal{Z} \times \mathcal{Y}^{(0)} \times \ldots \times \mathcal{Y}^{(k-1)}$ reading u; let x be the output of its projection on $\mathcal{Z}$; we say that c is full if its projection on $\mathcal{Z}$ is successful and, for every i, if u belongs to the domain $\mathcal{Y}^{(i)}$, then the projection of c on this transducer is also successful, and moreover *outputs* x *if* $x \in u|\mathcal{Y}^{(i)}|$.

Next, we shall adapt the bound $2^{\mathcal{O}(hs\ell k^5 n^{k+4})}$ stated in Lemma 2. This bound does not seem to be adequate to reproduce the pumping argument with the new definition of full computations. The reason is that the action $\circ_\mu$, which counts the number of successful computations with the same input, tells nothing about the *outputs* of the factorisations $f_1 f_2^r f_3$, and thus does not guarantee that these constructed computations are full if f is. Our idea is to replace $\circ_\mu$ by a more powerful device, the lead or delay covering of index $2^{\mathcal{O}(hs\ell k^5 n^{k+4})}$ of the transducer $\mathcal{Z} \cup \mathcal{Y}^{(0)} \cup \cdots \cup \mathcal{Y}^{(k-1)}$. Let $\mathcal{V}$ be the part of this covering restricted to the accessible states whose first component is a state of $\mathcal{Z}$. This part is a covering of $\mathcal{Z}$, and allows to retrieve, for every successful computation c of $\mathcal{Z}$, the differences (bounded by $2^{\mathcal{O}(hs\ell k^5 n^{k+4})}$) between the output of c and the outputs of the computations of $\mathcal{Y}^{(0)}, \ldots, \mathcal{Y}^{(k-1)}$ with the same input. Now, we can derive a new bound by applying the pumping argument on the product $\mathcal{V} \times \mathcal{Y}^{(0)} \times \ldots \times \mathcal{Y}^{(k-1)}$: the properties of the lead or delay covering $\mathcal{V}$ allows to show that the computations $f_1 f_2^r f_3$ are full. The number of states of $\mathcal{V}$ is a double exponential: $2^{2^{\mathcal{O}(hs\ell k^5 n^{k+4})}}$. The same expression is used in order to produce the factorisations $f_1 f_2^r f_3$, thus we have (with the same notation of Lemma 2):

Lemma 3. *Let $\mathcal{V}$ be the lead or delay covering of index $2^{\mathcal{O}(hs\ell k^5 n^{k+4})}$ of the transducer $\mathcal{Z} \cup \mathcal{Y}^{(0)} \cup \cdots \cup \mathcal{Y}^{(k-1)}$. If $|\mathcal{Z}| \subseteq |\mathcal{Y}^{(0)}| \cup \cdots \cup |\mathcal{Y}^{(k-1)}|$, then, for every successful computation c of $\mathcal{Z}$, there exists a compatible successful computation d in some $\mathcal{Y}^{(i)}$ such that $\langle c,\, d\rangle \leq 2^{2^{\mathcal{O}(hs\ell k^5 n^{k+4})}}$.*

The compatibility property stated in Lemma 3 can be verified, as the one in Lemma 2, with the construction of a new lead or delay covering (of index $2^{2^{\mathcal{O}(hs\ell k^5 n^{k+4})}}$ and for $\mathcal{V} \cup \mathcal{Y}^{(0)} \cup \cdots \cup \mathcal{Y}^{(k-1)}$) followed by an easy inspection of the differences stored in the final states of the covering. We can state our main result with a more precise expression for the complexity:

Theorem 6. *Let $(\mathcal{X}, \sigma)$ and $(\mathcal{Y}, \sigma)$ be k-valued morphic transducers with at most n states, where $\mathcal{X}$ and $\mathcal{Y}$ are transducers over $A^* \times B^*$ and σ is a morphism from B^* to C^*. Let ℓ be the maximal length of the outputs of the transitions of $\mathcal{X}$ and $\mathcal{Y}$, h the size of B, and $s = \max\{|b\sigma| \mid b \in B\}$. It is decidable in time complexity $2^{2^{2^{\mathcal{O}(hs\ell k^5 n^{k+4})}}}$ whether $|\mathcal{X}| = |\mathcal{Y}|$.* □

Acknowledgements. I am grateful to Jacques Sakarovitch for his encouragement and the anonymous referees for their useful remarks.

References

1. Weber, A.: Decomposing a k-valued transducer into k unambiguous ones. RAIRO Informatique Théorique et Applications 30(5), 379–413 (1996)
2. Sakarovitch, J., de Souza, R.: On the decomposition of k-valued rational relations. In: Albers, S., Weil, P. (eds.) Proceedings of STACS 2008, pp. 621–632 (2008), http://stacs-conf.org (to appear in Theory of Computing Systems)
3. Fischer, P.C., Rosenberg, A.L.: Multitape one-way nonwriting automata. Journal of Computer and System Sciences 2(1), 88–101 (1968)
4. Ibarra, O.: The unsolvability of the equivalence problem for ϵ-free NGSM's with unary input (output) alphabet and applications. SIAM Journal on Computing 7(4), 524–532 (1978)
5. Lisovik, L.P.: The identity problem for regular events over the direct product of free and cyclic semigroups (in Ukrainian). Dokl. Akad. Nauk Ukrainskoj RSR ser. A 6, 410–413 (1979)
6. Schützenberger, M.P.: Sur les relations rationnelles. In: Brakhage, H. (ed.) GI-Fachtagung 1975. LNCS, vol. 33, pp. 209–213. Springer, Heidelberg (1975)
7. Weber, A.: Decomposing finite-valued transducers and deciding their equivalence. SIAM Journal on Computing 22(1), 175–202 (1993)
8. Weber, A.: On the lengths of values in a finite transducer. Acta Informatica 29(6/7), 663–687 (1992)
9. Sakarovitch, J., de Souza, R.: On the decidability of bounded valuedness for transducers. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 588–600. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-85238-4_48
10. de Souza, R.: On the decidability of the equivalence for k-valued transducers. In: Ito, M., Toyama, M. (eds.) DLT 2008. LNCS, vol. 5257, pp. 252–263. Springer, Heidelberg (2008), http://dx.doi.org/10.1007/978-3-540-85780-8_20
11. Eilenberg, S.: Automata, Languages, and Machines, vol. A. Academic Press, London (1974)
12. Sakarovitch, J.: A construction on finite automata that has remained hidden. Theoretical Computer Science 204(1–2), 205–231 (1998)
13. Berstel, J.: Transductions and Context-Free Languages. B. G. Teubner (1979), http://www-igm.univ-mlv.fr/~berstel/LivreTransductions
14. Sakarovitch, J.: Éléments de théorie des automates. Vuibert (2003); English translation: Elements of Automata Theory, Cambridge University Press, Cambridge (to appear)
15. Béal, M.P., Carton, O., Prieur, C., Sakarovitch, J.: Squaring transducers: an efficient procedure for deciding functionality and sequentiality. Theoretical Computer Science 292, 45–63 (2003)
16. Carton, O.: The growth ratio of synchronous rational relations is unique. Theoretical Computer Science (376), 52–59 (2007)
17. de Souza, R.: Etude structurelle des transducteurs de norme bornée. PhD thesis, TELECOM ParisTech, Paris (2008), http://www.infres.enst.fr/~rsouza/publications/diss/these.pdf
18. Sakarovitch, J., de Souza, R.: On the decidability of bounded valuedness of transducers (manuscript), http://www.infres.enst.fr/~rsouza/DFV.pdf

Erasing in Petri Net Languages and Matrix Grammars

Georg Zetzsche

Universität Hamburg, MIN-Faculty, Department Informatik
Vogt-Kölln-Straße 30, 22527 Hamburg
georg.zetzsche@informatik.uni-hamburg.de

Abstract. It is shown that applying linear erasing to a Petri net language yields a language generated by a non-erasing matrix grammar. The proof uses Petri net controlled grammars. These are context-free grammars, where the application of productions has to comply with a firing sequence in a Petri net. Petri net controlled grammars are equivalent to arbitrary matrix grammars (without appearance checking), but a certain restriction on them (linear Petri net controlled grammars) leads to the class of languages generated by non-erasing matrix grammars.

It is also shown that in Petri net controlled grammars (with final markings and arbitrary labeling), erasing rules can be eliminated, which yields a reformulation of the problem of whether erasing rules in matrix grammars can be eliminated.

1 Introduction

A matrix grammar[1] is a context-free grammar together with a finite set of sequences of its productions. Its derivations are restricted to those that are built by concatenating these sequences.

Whether erasing rules can be eliminated, is an open question[2] in the theory of matrix grammars. In other words: is there an equivalent non-erasing matrix grammar for every matrix grammar (with erasing rules)? In order to approach this question, one can investigate language classes contained in $\mathbf{MAT}^\lambda$ (the matrix languages generated with erasing rules) with respect to whether their homomorphic images lie within $\mathbf{MAT}$ (the matrix languages generated without erasing rules). That way, one either finds techniques that may be generalized to $\mathbf{MAT}$ (proving that $\mathbf{MAT} = \mathbf{MAT}^\lambda$) or an example of a language in $\mathbf{MAT}$ that has a homomorphic image outside of $\mathbf{MAT}$ (proving that $\mathbf{MAT} \subset \mathbf{MAT}^\lambda$).

Since the Parikh images of the Petri net languages[3] and the matrix languages coincide, the former have often been useful in the theory of matrix grammars.

[1] In the following, by matrix grammars, we mean matrix grammars without appearance checking.

[2] In [1, p. 106], Theorem 2.1, the impossibility is claimed. However, the given references do not contain a proof for this.

[3] By Petri net languages, we mean those defined by final markings.

V. Diekert and D. Nowotka (Eds.): DLT 2009, LNCS 5583, pp. 490–501, 2009.

For example, the decidability of the reachability problem for Petri nets implied the decidability of the emptiness and thus the word problem for matrix grammars. Also, a deep result from Petri net theory (see [2]) is the regularity of all languages over one symbol, which implies the same for matrix languages. This has been used to prove the properness of the inclusion $\mathbf{MAT} \subset \mathbf{MAT}_{\mathrm{ac}}$ (where $\mathbf{MAT}_{\mathrm{ac}}$ denotes the languages generated by non-erasing matrix grammars with appearance checking). Furthermore, if one could show that arbitrary homomorphic images of Petri net languages are contained in **MAT**, this would imply an interesting result on multiset grammars (see section 4 for details).

For these reasons, the author investigated which homomorphic images of Petri net languages can be generated by non-erasing matrix grammars. The main result proven in this article is that for every Petri net language L (generated by a net without λ-transitions), if a homomorphism h is linear erasing on L, then $h(L)$ is contained in **MAT**.

The proof uses Petri net controlled grammars. These are context-free grammars, where the derivation is regulated by a Petri net. This is done by associating a production to every transition in the Petri net such that the firing of the transition leads to the application of the rule. This concept of Petri net controlled grammars has been introduced by Dassow and Turaev [3,4]. Their main focus was to compare the generative power depending on restrictions on the Petri nets, the accepting rule and the type of labeling. Here, however, arbitrary Petri nets with final markings and arbitrary labelings are considered.

In [3], it is shown that Petri net controlled grammars with erasing rules generate the same language class as arbitrary matrix grammars (in [5], the same result is proven, only in different terms). It is then shown here that when we restrict to so-called linear Petri net controlled grammars, their language class coincides with **MAT**. The main result of the paper is then proven by showing that for Petri net languages L (generated without λ-transitions) and a linear erasing homomorphism h on L, the language $h(L)$ is also generated by a linear Petri net controlled grammar.

In this paper, it is also shown that in Petri net controlled grammars, erasing rules can be eliminated. This answers a question open in [3,4] and yields a reformulation of the question of whether erasing rules can be eliminated in matrix grammars, see section 5 for details.

2 Definitions

A *monoid* is a set M together with an associative operation $\odot : M \times M \to M$ and a neutral element $e \in M$. For a monoid M with the operation $\odot$ and $m, m' \in M$, we write $m \sqsubseteq m'$ iff there is an $m'' \in M$ such that $m' = m \odot m''$. In this case, m is called a *prefix* of m'.

For an alphabet Σ, we will write Σ^* for the set of words over Σ. The empty word is denoted by $\lambda \in \Sigma^*$. Together with the concatenation as its operation, Σ^* is a monoid. For a symbol $x \in \Sigma$ and a word $w \in \Sigma^*$, let $|w|_x$ be the number of occurrences of x in w. By $|w|$, we will refer to the length of w. For alphabets

Σ, Γ and a language $L \subseteq \Sigma^*$, a homomorphism $h : \Sigma^* \to \Gamma^*$ is called *linear erasing on* L, iff there is a $k \in \mathbb{N}$ such that $|w| \leq k \cdot |h(w)|$ for every $w \in L$. For a language class $\mathcal{C}$, the class of languages $h(L)$, where $L \in \mathcal{C}$ and h is a linear erasing homomorphism on L, is denoted by $\mathcal{H}^{\text{lin}}(\mathcal{C})$.

Furthermore, $\Sigma^\oplus$ denotes the set of *multisets* over Σ, i.e. $\Sigma^\oplus$ is the set of mappings $\alpha : \Sigma \to \mathbb{N}$. The operation $+$ on $\Sigma^\oplus$ is defined by $(\alpha + \beta)(x) := \alpha(x) + \beta(x)$ for all $x \in \Sigma$. Together with the neutral element $\mathbf{0}$, defined by $\mathbf{0}(x) := 0$ for every $x \in \Sigma$, $\Sigma^\oplus$ is a (commutative) monoid. For a multiset $\mu \in \Sigma^\oplus$, let $\|\mu\| := \sum_{x \in \Sigma} \mu(x)$. Here, $\|\mu\|$ is called the *size* of μ. For $\alpha \sqsubseteq \beta$, let $(\beta - \alpha)(x) := \beta(x) - \alpha(x)$. The operation $\ominus$, however, is defined for all pairs $\alpha, \beta \in \Sigma^\oplus$: $(\beta \ominus \alpha)(x) := \max\{\beta(x) - \alpha(x), 0\}$. The *Parikh mapping* is the mapping $\Psi : \Sigma^* \to \Sigma^\oplus$ defined by $\Psi(w)(x) := |w|_x$ for all $w \in \Sigma^*$ and $x \in \Sigma$.

Definition 1 (Matrix grammar). *A* matrix grammar *is a quadruple* $G = (V, \Sigma, S, M)$*, where* V *is a finite alphabet of* nonterminal *symbols,* Σ *is a finite alphabet of* terminal *symbols,* $S \in V$ *is the* starting symbol *and* M *is a finite set of sequences of context-free productions over* V *and* Σ*, i.e.* $M = \{m_1, \ldots, m_n\}$*, where* $m_i = (A_1 \to w_1, \ldots, A_{k(i)} \to w_{k(i)})$ *and* $A_j \in V$*,* $w_j \in (V \cup \Sigma)^*$ *for* $1 \leq j \leq k(i)$ *and* $1 \leq i \leq n$*. The productions are also called* rules*. The elements of* M *are called* matrices*. For a rule* $A \to w$*,* A *is called its* left side *and* w *its* right side*. A rule whose right side is the empty word, is called* erasing*. A matrix grammar is called* non-erasing *if it has either no erasing rule, or* $S \to \lambda$ *is its only erasing rule and* S *does not appear on a right side.*

For $m \in M$*,* $m = (A_1 \to w_1, \ldots, A_k \to w_k)$*,* $A_j \in V$*,* $w_j \in (V \cup \Sigma)^*$*, and* $x, y \in (V \cup \Sigma)^*$*, we write* $x \Longrightarrow_m y$ *iff there are words* $u_0, \ldots, u_k \in (V \cup \Sigma)^*$*,* $\alpha_1, \ldots, \alpha_k, \beta_1, \ldots, \beta_k \in (V \cup \Sigma)^*$ *such that* $x = u_0$*,* $y = u_k$ *and* $u_{j-1} = \alpha_j A_j \beta_j$*,* $u_j = \alpha_j w_j \beta_j$ *for all* $1 \leq j \leq k$*.*

For $x, y \in (V \cup \Sigma)^*$*, let* $x \Longrightarrow_G y$ *iff there is a matrix* $m \in M$ *such that* $x \Longrightarrow_m y$*. Furthermore, let* $\Longrightarrow_G^*$ *be the reflexive transitive closure of* $\Longrightarrow_G$*. Words* $w \in (V \cup \Sigma)^*$ *with* $S \Longrightarrow_G^* w$ *are called* sentential forms*. We can now define the language generated by* G*:*

$$L(G) := \{w \in \Sigma^* \mid S \Longrightarrow_G^* w\}.$$

The class of languages generated by (non-erasing) matrix grammars is denoted by $\mathbf{MAT}^\lambda$ *(*$\mathbf{MAT}$*).*

Definition 2 (Labeled Petri net). *A* labeled Petri net *is an octuple* $N = (\Sigma, P, T, \partial_0, \partial_1, \sigma, \mu_0, F)$*, where* Σ *is a finite alphabet,* P *is a finite set of* places*, and* T *is a finite set of* transitions*.* $\partial_0, \partial_1 : T^\oplus \to P^\oplus$ *are homomorphisms, where* $\partial_0(t)$ *and* $\partial_1(t)$ *specifies the pre- or post-multiset, respectively, for every transition* t*.* $\sigma : T \to \Sigma \cup \{\lambda\}$ *is the* labeling function*,* $\mu_0 \in P^\oplus$ *is the* initial marking*, and* $F \subseteq P^\oplus$ *is a finite set of* final markings*.*

The elements of $P^\oplus$ *will be called* markings*. A transition* $t \in T$ *is called* λ-transition *iff* $\sigma(t) = \lambda$*. A labeled Petri net without* λ*-transitions is called* λ-free*. The binary relation* $\longrightarrow_t$ *on* $\Sigma^* \times P^\oplus$ *is defined by* $(w, \mu) \longrightarrow_t (w', \mu')$ *iff*

$$w' = w\sigma(t),\ \partial_0(t) \sqsubseteq \mu,\ \mu' = (\mu - \partial_0(t)) + \partial_1(t)$$

for $w \in \Sigma^*$, $x \in \Sigma$ and $\mu, \mu' \in P^{\oplus}$. A firing sequence from (w, μ) to (w', μ') in *N is a word $s \in T^*$, $s = t_1 \cdots t_n$, $t_i \in T$, such that*

$$(w_0, \mu_0) \longrightarrow_{t_1} (w_1, \mu_1) \longrightarrow_{t_2} \cdots \longrightarrow_{t_n} (w_n, \mu_n),$$

where $w_0 = w$, $\mu_0 = \mu$, $w_n = w'$, $\mu_n = \mu'$. We write $(w, \mu) \longrightarrow_N (w', \mu')$ iff there is a $t \in T$ such that $(w, \mu) \longrightarrow_t (w, \mu')$. Then the language generated by N is

$$L(N) := \{w \in \Sigma^* \mid \exists \mu \in F : (\lambda, \mu_0) \longrightarrow_N^* (w, \mu)\},$$

where $\longrightarrow_N^*$ is the reflexive transitive closure of $\longrightarrow_N$. Two labeled Petri nets N, N' are called equivalent*, iff $L(N) = L(N')$. The class of languages generated by (λ-free) labeled Petri nets is denoted by $\mathcal{L}_0^\lambda$ ($\mathcal{L}_0$).*

We will now define Petri net controlled grammars, which play a central role in the proof of our main result.

Definition 3 (Petri net controlled grammar). *A* Petri net controlled grammar *(or shortly* Petri net grammar*) is a tuple $G = (V, \Sigma, S, R, N)$, where V,Σ,S are defined as in matrix grammars, $R \subseteq V \times (V \cup \Sigma)^*$ is a set of context-free productions, $N = (R, P, T, \partial_0, \partial_1, \sigma, \mu_0, F)$ is a labeled Petri net, whose alphabet is the set R of productions. A Petri net grammar G is called non-erasing iff either all the productions in R are non-erasing or $S \to \lambda$ is the only erasing rule and S does not occur on any right side.*

The derivation relation on $(V \cup \Sigma)^* \times P^{\oplus}$ is defined by $(w, \mu) \Longrightarrow_t (w', \mu')$ iff $\partial_0(t) \sqsubseteq \mu$, $\mu' = (\mu - \partial_0(t)) + \partial_1(t)$ and one of the following holds:

- *$\sigma(t) = A \to u$ and there exist words $\alpha, \beta \in (V \cup \Sigma)^*$ such that $w = \alpha A \beta$, $w' = \alpha u \beta$,*
- *$\sigma(t) = \lambda$ and $w' = w$.*

A pair $(w, \mu) \in (V \cup \Sigma)^* \times P^{\oplus}$ will be called a configuration*, w its* sentential form *and μ its* marking*. A* firing sequence from (w, μ) to (w', μ') in *G is a word $s \in T^*$, $s = t_1 \cdots t_n$, $t_i \in T$, such that*

$$(w_0, \mu_0) \Longrightarrow_{t_1} (w_1, \mu_1) \Longrightarrow_{t_2} \cdots \Longrightarrow_{t_n} (w_n, \mu_n),$$

where $w_0 = w$, $\mu_0 = \mu$, $w_n = w'$, $\mu_n = \mu'$. We write $(w, \mu) \Longrightarrow_G (w', \mu')$ if there is a $t \in T$ such that $(w, \mu) \Longrightarrow_t (w', \mu')$. The generated language is defined as

$$L(G) := \{w \in \Sigma^* \mid \exists \mu \in F : (S, \mu_0) \Longrightarrow_G^* (w, \mu)\},$$

where $\Longrightarrow_G^*$ is the reflexive transitive closure of $\Longrightarrow_G$. The class of languages generated by (non-erasing) Petri net grammars will be denoted by $\mathbf{PN}^\lambda$ ($\mathbf{PN}$).

It is easily seen that these classes coincide with the classes $\mathbf{PN}^\lambda(\lambda, t)$ and $\mathbf{PN}(\lambda, t)$ in [3,4].

Later in this paper, we want to simulate Petri net grammars by non-erasing matrix grammars. This is done by encoding the multiset μ from a configuration

(w, μ) in the sentential form of the matrix grammar. However, the construction requires that the size of the marking is at most linear in the length of the word w of that configuration. Therefore, the notion of linear Petri net grammars will be needed.

Definition 4. *Let $G = (V, \Sigma, S, R, N)$, $N = (R, P, T, \partial_0, \partial_1, \sigma, \mu_0, F)$, be a non-erasing Petri net grammar, $w \in L(G) \setminus \{\lambda\}$ and $k \in \mathbb{N}$. A derivation*

$$(w_0, \mu_0) \Longrightarrow_G (w_1, \mu_1) \Longrightarrow_G \ldots \Longrightarrow_G (w_n, \mu_n),$$

*$w_0 = S$, $\mu_n \in F$, is called a k-*derivation of w*, iff $w_n = w$ and $\|\mu_i\| \leq k \cdot |w_i|$ for every $0 \leq i \leq n$. The grammar G is said to be k-*linear*, iff for every $w \in L(G) \setminus \{\lambda\}$ there is a k-derivation of w. G is called* linear *iff it is k-linear for some $k \in \mathbb{N}$. The class of languages generated by linear Petri net grammars will be denoted by* **LinPN**.

Note that the condition is only required for words $w \in L(G) \setminus \{\lambda\}$. This is due to the fact that we would otherwise require the final marking to be empty whenever the empty word is derived. For our purposes, it is enough that the condition is fulfilled for all but finitely many words from $L(G)$, so we can exclude the empty word.

3 Linear Petri Net Grammars and Matrix Grammars

The first step in the proof of $\mathcal{H}^{\text{lin}}(\mathcal{L}_0) \subseteq$ **MAT** is to show that linear Petri net grammars are equivalent to non-erasing matrix grammars. That is, **LinPN** = **MAT**. To simplify the construction, we need a normal form for labeled Petri nets.

Lemma 1. *For every labeled Petri net $N = (\Sigma, P, T, \partial_0, \partial_1, \sigma, \mu_0, F)$, there is an equivalent one $N' = (\Sigma, P', T', \partial'_0, \partial'_1, \sigma', \mu'_0, F')$, where*

- *$\mu'_0 \in P'$, $F' \subseteq P'$,*
- *every reachable marking in $F'^{\oplus}$ is already contained in F'.*
- *the empty marking cannot be reached in N',*
- *N' is λ-free if N is λ-free,*
- *the normal form preserves linearity, i.e. if $G = (V, \Gamma, S, R, N)$ is a linear Petri net grammar, then $G = (V, \Gamma, S, R, N')$ is too.*

The proof is not difficult and therefore omitted due to space constraints.

Theorem 1. **LinPN** = **MAT**.

Proof. We begin with the inclusion **LinPN** $\subseteq$ **MAT**. Let $G = (V, \Sigma, S, R, N)$, $N = (R, P, T, \partial_0, \partial_1, \sigma, \mu_0, F)$, be a k-linear Petri net grammar where N is in the normal form of Lemma 1. We will construct a matrix grammar G' such that $L(G) \setminus \{\lambda\} \subseteq L(G') \subseteq L(G)$, i.e. $L(G')$ and $L(G)$ coincide up to the word λ. This suffices, for if $\lambda \in L(G)$, it is easy to modify G' so that it also generates λ.

Therefore, we can assume that G does not contain the production $S \to \lambda$ (and thus no erasing rule).

In the constructed matrix grammar, the marking of the simulated net will have to be stored inside the sentential form. Therefore, it will have a set of nonterminals that can store multisets up to a certain size:

$$\bar{V} := \{(x, \mu) \mid x \in V \cup \Sigma,\ \mu \in P^{\oplus},\ \|\mu\| \leq k\}.$$

Note that $\bar{V}$ is finite. Every configuration $(w, \mu) \in (V \cup \Sigma)^* \times P^{\oplus}$ of the Petri net grammar will be represented by a sentential form $w' \in \bar{V}^*$, $w' = (x_1, \mu_1) \cdots (x_n, \mu_n)$, such that $\mu_1 + \cdots + \mu_n = \mu$ and $x_1 \cdots x_n = w$. We will need an embedding homomorphism $\iota : (V \cup \Sigma)^* \to \bar{V}^*$, that is defined by $\iota(x) := (x, \mathbf{0})$ for every $x \in V \cup \Sigma$.

We will now describe the matrices that simulate the firing of a transition and the application of the corresponding context-free rule. In order to do that, we will regard matrices as words over context-free productions and describe the matrices as a set of words. For the sake of readability, we will only give the matrices for one transition. The whole set of matrices is then obtained by constructing these for every transition.

Let $t \in T$ be a transition with pre-multiset $\nu_0 = \partial_0(t)$ and post-multiset $\nu_1 = \partial_1(t)$. The set U of context-free productions will simulate the context-free rule assigned to t. If t does not have a rule, i.e. $\sigma(t) = \lambda$, then $U := \{\lambda\}$. Now consider the case $\sigma(t) = A \to u$, $A \in V$, $u \in (V \cup \Sigma)^+$. Since $|u| \geq 1$, we can write $u = xu'$, $x \in V \cup \Sigma$, $u' \in (V \cup \Sigma)^*$. Then U simulates the application of $A \to u$:

$$U := \{(A, \mu) \to (x, \mu)\iota(u') \mid \mu \in P^{\oplus},\ \|\mu\| \leq k\}.$$

We also need rules to simulate the subtraction of the pre-multiset and the addition of the post-multiset of t. These are constructed separately for every place $p \in P$:

$$\begin{aligned} S_p &:= \{(y, \mu) \to (y, \mu - p) \mid y \in V \cup \Sigma,\ \mu \in P^{\oplus},\ \|\mu\| \leq k,\ \mu(p) \geq 1\}, \\ A_p &:= \{(y, \mu) \to (y, \mu + p) \mid y \in V \cup \Sigma,\ \mu \in P^{\oplus},\ \|\mu + p\| \leq k\}. \end{aligned}$$

The set of matrices simulating the application of t and $A \to u$ is then

$$M_t := U S_{p_1}^{\nu_0(p_1)} \cdots S_{p_n}^{\nu_0(p_n)} A_{p_1}^{\nu_1(p_1)} \cdots A_{p_n}^{\nu_1(p_n)},$$

where $P = \{p_1, \ldots, p_n\}$. Since G is k-linear, every generated word has a derivation such that in every configuration, the marking can fit into the sentential form in our simulation.

To complete the construction, we need productions that convert the multiset-carrying symbols from $\bar{V}$ to ordinary terminal symbols. Here, we make sure that a pair $(y, \mu) \in \bar{V}$ can only be converted to y when $\mu = \mathbf{0}$ or $\mu \in F$. This implies that a sentential form consisting solely of terminal symbols can only be derived when a marking from $F^{\oplus}$ was reached in the simulated net. According to the normal form, this is the case iff a marking in F was reached. The remaining productions are

$$M' := \{(y, \mathbf{0}) \to y \mid y \in \Sigma\} \cup \{(y, p) \to y \mid y \in \Sigma,\ p \in F\}.$$

The complete set of matrices is then $M := M' \cup \bigcup_{t \in T} M_t$. The start symbol is $(S, \mu_0) \in \bar{V}$. The matrix grammar is then $G' = (\bar{V}, \Sigma, (S, \mu_0), M)$.

For the direction $\mathbf{MAT} \subseteq \mathbf{LinPN}$, we only have to construct a Petri net whose set of firing sequences that end in a final marking is of the form M^*, where M is a finite set of words. This, however, can already be done by a finite automaton and is therefore easy to realize in a net. □

4 Linear Erasing in Petri Net Languages

In this section, it is shown that $\mathcal{H}^{\text{lin}}(\mathcal{L}_0) \subseteq \mathbf{LinPN} = \mathbf{MAT}$. In other words, for $L \in \mathcal{L}_0$ and a linear erasing homomorphism h on L, we have $h(L) \in \mathbf{LinPN} = \mathbf{MAT}$. The result is stronger than $\mathcal{L}_0 \subseteq \mathbf{MAT}$, since it is known that $\mathcal{L}_0$ is strictly contained in $\mathcal{H}^{\text{lin}}(\mathcal{L}_0)$ (see, for example, Theorem 4 in [6]).

The techniques used in this proof may be used later to prove that homomorphic images of larger classes are contained in $\mathbf{MAT}$. For example, it would be interesting if one could show that $\mathcal{L}_0^\lambda \subseteq \mathbf{MAT}$, for this would imply that the Parikh images of $\mathbf{MAT}$ and $\mathbf{MAT}^\lambda$ coincide (it is known that $\Psi(\mathcal{L}_0^\lambda) = \Psi(\mathbf{MAT}^\lambda)$, see [2]) and therefore arbitrary multiset grammars are equivalent to monotone multiset grammars (see [7] for multiset grammars).

In order to verify the correctness of the construction, we will need a lemma. It will be necessary to traverse the symbols of a word in an order that makes sure that a certain resource is never used up. We will regard the set of integers as a monoid with the addition as its operation. The consumption and production of the resource are then given by a homomorphism $\varphi : \Sigma^* \to \mathbb{Z}$. We are given a word w such that $\varphi(w) \geq 0$, i.e. after traversing the symbols of w in the order given by w, there are still resources in the end. We, however, want to make sure that they are never used up at any given time. The following lemma shows that this is possible by decomposing $w = uv$ and traversing first v and then u.

Lemma 2. *Let $\varphi : \Sigma^* \to \mathbb{Z}$ a monoid homomorphism and $w \in \Sigma^*$ a word with $\varphi(w) \geq 0$. Then w admits a decomposition $w = uv$ such that for every prefix z of vu, we have $\varphi(z) \geq 0$.*

Proof. Let $w = w_1 \cdots w_n$, $w_1, \ldots, w_n \in \Sigma$ and choose $j \in \{1, \ldots, n\}$ such that $\varphi(w_1 \cdots w_j)$ is minimal. Then let $u = w_1 \cdots w_j$, $v = w_{j+1} \cdots w_n$. For a prefix z of vu, we distinguish two cases. If z is a prefix of v, say $z = w_{j+1} \cdots w_k$. Then

$$\varphi(z) = \varphi(uz) - \varphi(u) = \varphi(w_1 \cdots w_k) - \varphi(w_1 \cdots w_j) \geq 0$$

since $\varphi(w_1 \cdots w_j)$ was minimal. If $z = vz'$ where z' is a prefix of u, say $z' = w_1 \cdots w_k$. Then $\varphi(z') = \varphi(w_1 \cdots w_k) \geq \varphi(w_1 \cdots w_j) = \varphi(u)$ because of the choice of j. This implies

$$\varphi(z) = \varphi(vz') = \varphi(v) + \varphi(z') \geq \varphi(v) + \varphi(u) = \varphi(uv) = \varphi(w) \geq 0. \qquad \square$$

In order to simplify the construction in the theorem, we need a further lemma to state a closure property of **MAT**. For an alphabet Σ and $x \in \Sigma$, let $\delta_x : \Sigma^* \to \Sigma^*$ be the homomorphism satisfying $\delta_x(x) = \lambda$ and $\delta_x(y) = y$ for every $y \in \Sigma$, $y \neq x$. A homomorphism $h : \Sigma^* \to \Sigma^*$ is called *single erasing on* $L \subseteq \Sigma^*$ iff there is an $x \in \Sigma$ such that $h = \delta_x$ and $|w|_x \leq 1$ for every $w \in L$. A language class $\mathcal{C}$ is called *closed against single erasing* iff $h(L) \in \mathcal{C}$ for every $L \in \mathcal{C}$ and every single erasing homomorphism h on L.

Lemma 3. **MAT** *is closed against single erasing.*

The proof can easily be done using the closure of **MAT** against inverse homomorphism, non-erasing homomorphism, and union.

Theorem 2. $\mathcal{H}^{\mathrm{lin}}(\mathcal{L}_0) \subseteq$ **LinPN** $=$ **MAT**.

Proof. Let $N = (\Sigma, P, T, \partial_0, \partial_1, \sigma, \mu_0, F)$ be a labeled Petri net and h be a linear erasing homomorphism on $L = L(N)$. Since $\mathcal{L}_0$ is closed against non-erasing homomorphisms, we can assume that $h : \Sigma^* \to \Sigma^*$ and there is an $x \in \Sigma$ such that $h(x) = \lambda$ and $h(y) = y$ for every $y \in \Sigma$, $y \neq x$. The fact that h is linear erasing on L implies that $|w| \leq k \cdot |h(w)|$ for every $w \in L$.

Let $a, b \notin \Sigma$ be new symbols, $\Sigma' := \Sigma \cup \{a, b\}$. We will construct a linear Petri net grammar G such that δ_a and δ_b are single erasing on $L(G)$ and $\delta_a(\delta_b(L(G))) = L$, which implies that $L \in$ **MAT**.

In order to guarantee that our constructed Petri net grammar will be linear, we have to make sure that every word has a derivation where the size of the marking is linear in the length of the sentential form in every step. Therefore, we cannot just use the net N as the controlling net in the Petri net grammar. It may create large markings without increasing the length of the sentential form (note that, when N produces an x, the grammar would not enhance the sentential form).

However, since we are constructing a Petri net grammar, a firing sequence of N does not necessarily have to be simulated in the order it occurs in N. We will see that it is possible to decompose any firing sequence $s \in T^*$ of N in $s = uv$ such that, when simulating them in the order vu, we always have a long enough sentential form so that the marking is linear in size.

The set of places will be $P' := P \cup \bar{P} \cup \hat{P}$, where $\bar{P} := \{\bar{p} \mid p \in P\}$, $\hat{P} = \{\hat{p} \mid p \in P\}$ are sets of new places. We will need the embedding homomorphisms $\bar{\iota} : P^{\oplus} \to \bar{P}^{\oplus}$, $\hat{\iota} : P^{\oplus} \to \hat{P}^{\oplus}$, defined by $p \mapsto \bar{p}$ and $p \mapsto \hat{p}$, respectively. Every reachable marking in the grammar will be of the form $\alpha + \beta + \gamma$, where $\alpha \in P^{\oplus}$, $\beta \in \bar{P}^{\oplus}$, $\gamma \in \hat{P}^{\oplus}$. During the simulation, the sentential forms will be of the form $w_1 A w_2 B$, where w_1,w_2 are the parts generated by u and v, respectively. The simulation of u and v will be mostly independent of each other. The part α will be the marking used to simulate u. Analogously, γ will be used to simulate v. In order to make sure that the marking from which v starts is a prefix of the marking in which u ends, the former is stored in β. In the end, α and β are synchronously reduced until β is $\mathbf{0}$ to guarantee the prefix relation. The set of final markings consists of exactly those where β is $\mathbf{0}$ and the sum of α and γ corresponds to a final marking in N. We have four types of transitions:

$$\begin{aligned}T' :=&\{Z_t^{(j)} \mid t \in T,\ j \in \{0,1\}\} \cup \{r_p \mid p \in P\} \cup \{s_{i,j} \mid i,j \in \{0,1\}\}\\ &\cup \{Z_{t,\kappa}^{(j)} \mid \kappa \sqsubseteq \partial_0(t),\ j \in \{0,1\}\}.\end{aligned}$$

Here, $Z_{t,\kappa}^{(j)}$, $Z_t^{(j)}$, r_p and $s_{i,j}$ are new transitions. The transitions $Z_t^{(j)}$ are used to simulate u. The parameter $j \in \{0,1\}$ is used to decide whether this is the last symbol added in u.

$$\partial_i'(Z_t^{(j)}) := \partial_i(t), \quad \sigma'(Z_t^{(j)}) := \begin{cases} A \to \sigma(t)A^j a^{1-j} & \text{if } \sigma(t) \neq x\\ \lambda & \text{if } \sigma(t) = x\end{cases}$$

for $i = 0,1$. The transitions $Z_{t,\kappa}^{(j)}$ simulate v. They also guess the marking which v starts from. Therefore, to simulate t, $Z_{t,\kappa}^{(j)}$ (where $\kappa \sqsubseteq \partial_0(t)$) guesses that $\partial_0(t) - \kappa$ is subtracted from the current marking and κ is added to the initial marking. The parameter $j \in \{0,1\}$ defines whether the generated symbol is the last one:

$$\partial_0'(Z_{t,\kappa}^{(j)}) := \hat\iota(\partial_0(t) - \kappa), \quad \partial_1'(Z_{t,\kappa}^{(j)}) := \bar\iota(\kappa) + \hat\iota(\partial_1(t)),$$

$$\sigma'(Z_{t,\kappa}^{(j)}) = \begin{cases} B \to \sigma(t)B^j b^{1-j} & \text{if } \sigma(t) \neq x\\ \lambda & \text{if } \sigma(t) = x\end{cases}$$

Now the transitions r_p are used to reduce the marking reached by u and the initial marking of v synchronously: $\partial_0'(r_p) := p + \bar p$, $\partial_1'(r_p) := \mathbf{0}$. The transitions $s_{i,j}$ create the sentential form $A^i a^{1-i} B^j b^{1-j}$, $i,j \in \{0,1\}$ depending on whether the part u or v of the simulated firing sequence is empty ($i = 1$ iff $|v| > 0$ and $j = 1$ iff $|u| > 0$).

$$\partial_0'(s_{i,j}) := \mathbf{0}, \quad \partial_1'(s_{i,j}) := \mathbf{0}, \quad \sigma'(s_{i,j}) := S \to A^i a^{1-i} B^j b^{1-j}.$$

The constructed Petri net grammar is then defined as $G := (V, \Sigma', S, R, N')$, $N' := (R, P', T', \partial_0', \partial_1', \sigma', \mu_0, F')$, where $V := \{S, A, B\}$, $R := \sigma'(T')$ and $F' := \{\mu + \hat\iota(\mu') \mid \mu, \mu' \in P^{\oplus},\ \mu + \mu' \in F\}$.

It is easily seen that δ_b and δ_a are single erasing on $L(G)$ or $\delta_b(L(G))$, respectively. Hence, it remains to be shown that $\delta_a(\delta_b(L(G))) = L(N)$, and that G is indeed a linear Petri net grammar. Let

$$M := \max\{\|\partial_0(t)\| + \|\partial_1(t)\| \mid t \in T\}, \qquad \ell := k \cdot M.$$

First, we will show that $L(N) \subseteq \delta_a(\delta_b(L(G)))$ and G is ℓ-linear, i.e. for every $w \in L(N)$ there is an ℓ-derivation in G of a word w' such that $\delta_a(\delta_b(w')) = w$. Therefore, let $s \in T^*$ be a firing sequence in N with $\sigma(s) = w$. Let the homomorphism $\varphi : T^* \to \mathbb{Z}$ be defined by $\varphi(t) := \ell - M$ if $\sigma(t) \neq x$ and $\varphi(t) := -M$ if $\sigma(t) = x$. Then, since $|w| \leq k \cdot |h(w)| = k(|w| - |w|_x)$, we have

$$\begin{aligned}\varphi(s) &= \ell \cdot (|w| - |w|_x) - M \cdot |w|\\ &= k \cdot M(|w| - |w|_x) - M \cdot |w|\\ &= M \cdot (k(|w| - |w|_x) - |w|) \geq 0.\end{aligned}$$

According to Lemma 2, there is a decomposition $s = uv$ such that $\varphi(z) \geq 0$ for every prefix z of vu. Let $u = u_1 \cdots u_n$, $v = v_1 \cdots v_m$, $u_i, v_j \in T$. Furthermore, let $P = \{p_1, \ldots, p_f\}$ and $q, q' \in \{0, 1\}$, where $q := 1$ iff $|v| > 0$ and $q' := 1$ iff $|u| > 0$. We claim that there are $\kappa, \kappa_i \in P^\oplus$, $1 \leq i \leq m$, such that

$$s' := s_{q,q'} \underbrace{Z^{(1)}_{v_1,\kappa_1} \cdots Z^{(1)}_{v_{m-1},\kappa_{m-1}} Z^{(0)}_{v_m,\kappa_m} Z^{(1)}_{u_1} \cdots Z^{(1)}_{u_{n-1}} Z^{(0)}_{u_n}}_{=:s''} \prod_{i=1}^{f} r_{p_i}^{\kappa(p_i)},$$

is a firing sequence in G which corresponds to an ℓ-derivation of w. It is easily seen that s' generates a word w' such that $\delta_a(\delta_b(w')) = w$.

The multisets $\kappa, \kappa_i \in P^\oplus$ are obtained as follows. κ_i is always the part of the pre-multiset of v_i that is not present in the current marking in $\hat{P}^\oplus$. In order to make the firing of $Z^{(j)}_{v_i,\kappa_i}$ possible, it has as pre-multiset only $\partial_0(v_i) - \kappa_i$ and adds $\bar{\iota}(\kappa_i)$ to the $\bar{P}^\oplus$-part of the marking. This makes sure that this part of the marking will be produced later while simulating u. Let $\hat{\iota}(\gamma_i)$ be the $\hat{P}^\oplus$-part of the marking after firing $Z^{(j)}_{v_i,\kappa_i}$. Now we can define κ_i and give a formula for γ_i inductively:

$$\kappa_1 := \partial_0(v_1), \qquad \kappa_i := \partial_0(v_i) \ominus \gamma_{i-1}.$$

$$\gamma_0 = \mathbf{0}, \qquad \gamma_i = (\gamma_{i-1} - (\partial_0(v_i) - \kappa_i)) \ + \ \partial_1(v_i).$$

κ is the $\bar{P}^\oplus$-part of the marking after the simulation of v. Therefore, $\kappa := \kappa_1 + \cdots + \kappa_m$.

Now we show that s' corresponds to an ℓ-derivation. Since $s_{q,q'}$ produces one symbol in the sentential form and does not change the marking, the condition for the ℓ-derivation is fulfilled after $s_{q,q'}$. Furthermore, since the transitions r_p do not increase the marking size, it suffices to show that the condition is satisfied during the firing of s''. Let

$$\bar{T} := \{Z^{(j)}_{v_i,\kappa_i} \mid 1 \leq i \leq m,\ j \in \{0,1\}\} \cup \{Z^{(j)}_{u_i} \mid 1 \leq i \leq n,\ j \in \{0,1\}\}$$

and $\rho : \bar{T}^* \to \mathbb{Z}$ be the homomorphism defined by $\rho(t) := \ell \cdot g - \|\partial_1'(t)\|$, where g is the number of symbols t adds to the sentential form. If we can show that $\rho(z) \geq 0$ for every prefix z of s'', then s' indeed corresponds to an ℓ-derivation. We will need the homomorphism $\tau : \bar{T}^* \to T^*$, $Z^{(j)}_{v_i,\kappa_i} \mapsto v_i$, $Z^{(j)}_{u_i} \mapsto u_i$. It is clear from the definition of φ that $\rho(t) \geq \varphi(\tau(t))$ for every $t \in \bar{T}$. Therefore, $\rho(z) \geq \varphi(\tau(z)) \geq 0$ for $z \sqsubseteq s''$, since $\tau(z)$ is then a prefix of vu. Hence, s' corresponds to an ℓ-derivation.

The inclusion $\delta_a(\delta_b(L(G))) \subseteq L(N)$ can now be done analogously by noting that every derivation in G can be rearranged so that it corresponds to a firing sequence in N in the same way as s' corresponds to s. □

It is known that there is a context-free language that is not contained in $\mathcal{L}_0^\lambda$ (see [6] and see [8] for the decidability of the reachability problem). This yields:

Corollary 1. $\mathcal{H}^{\mathrm{lin}}(\mathcal{L}_0) \subset \mathbf{MAT}$.

5 Petri Net Grammars with and without Erasing Rules

In this section, it is shown that $\mathbf{PN} = \mathbf{PN}^\lambda$. This result sheds new light on the open question of whether erasing rules can be eliminated in matrix grammars. For on the one hand, we have $\mathbf{PN} = \mathbf{MAT}^\lambda$. On the other hand, it is well-known (see, for example, [9]) that non-erasing matrix grammars are equivalent to regularly controlled non-erasing context-free grammars. Therefore, the question can be reformulated: *Does it make a difference whether context-free grammars without erasing rules are controlled by Petri nets or by finite automata?*

Theorem 3. $\mathbf{PN} = \mathbf{PN}^\lambda$.

Proof. Since $\mathbf{PN} \subseteq \mathbf{PN}^\lambda$ is clear from the definition, it suffices to show $\mathbf{PN}^\lambda \subseteq \mathbf{PN}$. Let $G = (V, \Sigma, S, R, N)$, $N = (R, P, T, \partial_0, \partial_1, \sigma, \mu_0)$, be a (possibly erasing) Petri net grammar. We construct a grammar G' with $L(G') = L(G) \setminus \{\lambda\}$. This does not mean a loss of generality, since in the case $\lambda \in L(G)$, it is easy to modify G' such that the start symbol can be rewritten to λ and that the start symbol does not occur on the right side of a production.

The simulation will work as follows. Every configuration (w, μ) of G will be represented by a configuration (w', μ') such that w has a decomposition $w = u_1 v_1 \cdots u_n v_n$, $u_1, \ldots, u_n \in (V \cup \Sigma)^*$, $v_1, \ldots, v_n \in V^*$, with $w' = u_1 \cdots u_n$ and $\mu' = \mu + \iota(\Psi(v_1 \cdots v_n))$. Here, $\iota : V^\oplus \to Q^\oplus$ is the embedding homomorphism $\iota : x \mapsto q_x$ into the new set of places $Q := \{q_x \mid x \in V\}$. In other words, w will be split into two parts, one of which is the new sentential form and one of which will be added (as a multiset) to the marking of the Petri net. The part that is kept as a multiset is the part of w that will later be rewritten to λ. Note that since these symbols are erased anyway, their order and position in the sentential form is irrelevant and therefore they can be stored safely in a multiset.

For every transition $t \in T$ with $\sigma(t) \neq \lambda$, we will add a new set of transitions. Here, the firing of such a new transition will simulate the firing of t. The transitions $t \in T$ with $\sigma(t) = \lambda$ will be kept unchanged.

Let $t \in T$, $\sigma(t) = A \to w$, $A \in V$, $w \in (V \cup \Sigma)^*$. To construct the new transitions, we need the finite set

$$\begin{aligned} X_t := \{(u, \nu) \in (V \cup \Sigma)^* \times V^\oplus \mid\ & w = u_1 v_1 \cdots u_n v_n, \\ & u_1, \ldots, u_n \in (V \cup \Sigma)^*,\ v_1, \ldots, v_n \in V^*, \\ & u = u_1 \cdots u_n,\ \nu = \Psi(v_1 \cdots v_n)\}. \end{aligned}$$

Now, for every $(u, \nu) \in X_t$, we add a new transition $s_{t,u,\nu}$. It will simulate t, apply the production $A \to u$, and add $\iota(\nu)$ to the marking of the net. If $u = \lambda$, the rule $A \to \lambda$ will be simulated by removing $\iota(A)$ from the marking and assigning no production to $s_{t,u,\nu}$. Therefore, we distinguish two cases.

- If $|u| = 0$, let $\sigma'(s_{t,u,\nu}) := \lambda$ and $\partial_0'(s_{t,u,\nu}) := \partial_0(t) + \iota(A)$.
- If $|u| \geq 1$, let $\sigma'(s_{t,u,\nu}) := A \to u$ and $\partial_0'(s_{t,u,\nu}) := \partial_0(t)$.

In either case, in addition to the post-multiset of t, $s_{t,u,\nu}$ will deposit $\iota(\nu)$: $\partial_1'(s_{t,u,\nu}) := \partial_1(t) + \iota(\nu)$. The set of transitions in the new grammar is then

$$T' := \{t \in T \mid \sigma(t) = \lambda\} \cup \{s_{t,u,\nu} \mid t \in T,\ \sigma(t) \neq \lambda,\ (u, \nu) \in X_t\}.$$

$$\begin{tikzcd}
 & \mathbf{MAT}^{\lambda} = \mathbf{PN}^{\lambda} = \mathbf{PN} \\
\mathcal{L}_0^{\lambda} \arrow[ru] & \mathbf{MAT} = \mathbf{LinPN} \arrow[u, dash] \\
\mathcal{H}^{\mathrm{lin}}(\mathcal{L}_0) \arrow[u, dash] \arrow[ru] & \\
\mathcal{L}_0 \arrow[u] &
\end{tikzcd}$$

Fig. 1. Relations between language classes

On $\{t \in T \mid \sigma(t) = \lambda\}$, the functions σ', ∂_0' and ∂_1' coincide with $\sigma, \partial_0, \partial_1$, respectively. As mentioned above, the set of places is extended by the new places in Q: $P' = P \cup Q$. The new grammar is then $G' = (V, \Sigma, S, R', N')$, $N' = (R', P', T', \partial_0', \partial_1', \sigma', \mu_0, F)$. □

We conclude by summarizing the relations between the language classes in Figure 1. The lines (arrows) denote (proper) inclusions of the lower language classes into the upper language classes. Those that are not directly connected are not necessarily incomparable.

References

1. Dassow, J., Păun, G., Salomaa, A.: Grammars with controlled derivations. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, vol. 2. Springer, Heidelberg (1997)
2. Hauschildt, D., Jantzen, M.: Petri net algorithms in the theory of matrixgrammars. Acta Informatica 31, 719–728 (1994)
3. Dassow, J., Turaev, S.: Arbitrary Petri net controlled grammars. In: Bel-Enguix, G., Jiménez-López, M.D. (eds.) Proc. 2nd International Workshop ForLing 2008, Tarragona, Spain, pp. 27–39 (2008)
4. Dassow, J., Turaev, S.: Grammars controlled by special Petri nets. In: Dediu, A.-H., Ionescu, A.-M., Martín-Vide, C. (eds.) LATA 2009. LNCS, vol. 5457, pp. 326–337. Springer, Heidelberg (2009)
5. Höppner, M.: Eine Charakterisierung der Szilardsprachen und ihre Verwendung als Steuersprachen. In: Siefkes, D. (ed.) GI 1974. LNCS, vol. 26, pp. 113–121. Springer, Heidelberg (1975)
6. Greibach, S.A.: Remarks on blind and partially blind multicounter machines. Theoretical Computer Science, 311–324 (1978)
7. Kudlek, M., Martín-Vide, C., Păun, G.: Toward a formal macroset theory. In: Calude, C.S., Pun, G., Rozenberg, G., Salomaa, A. (eds.) Multiset Processing. LNCS, vol. 2235, pp. 123–133. Springer, Heidelberg (2001)
8. Mayr, E.W.: An algorithm for the general Petri net reachability problem. SIAM J. Comput. 13(3), 441–460 (1984)
9. Dassow, J., Păun, G.: Regulated rewriting in formal language theory. Springer, Heidelberg (1989)

Author Index